Interactive Machine Learning: Robotics

Brad Hayes

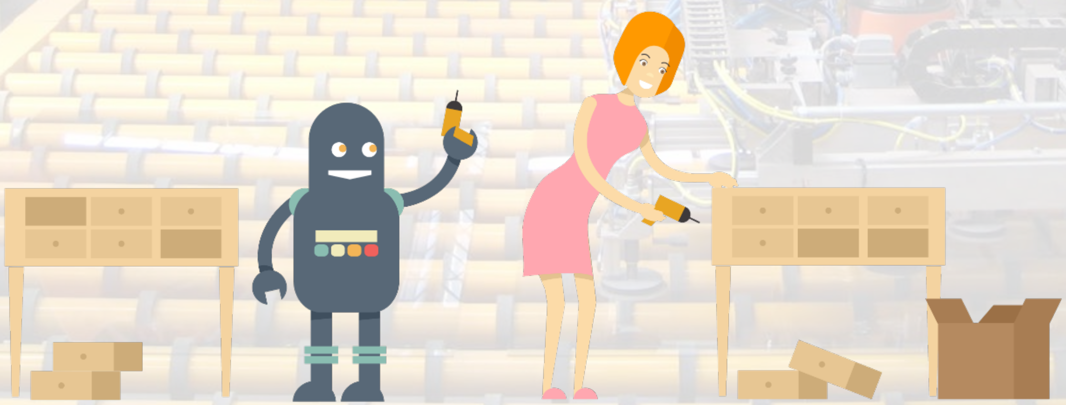
University of Colorado Boulder

Ece Kamar

Microsoft Research

Matthew E. Taylor

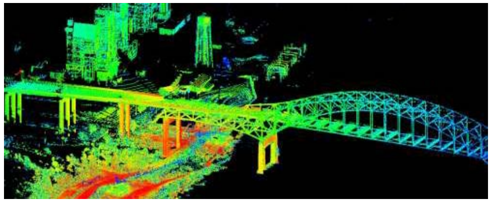
Washington State University

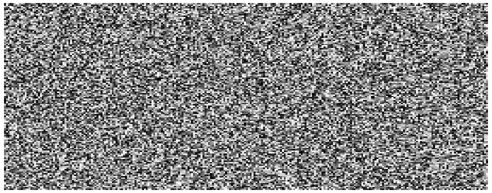


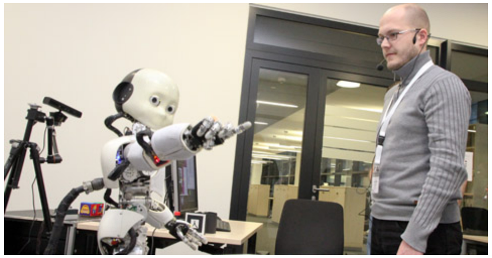
InteractiveML.net

Problems with Robots

- Robots operate in a world they cannot properly model
  - All models are wrong, some models are useful
- The data that a robot's algorithms depend upon are noisy, non-iid, and occasionally non-stationary
- Human inputs help compensate for this
  - Humans are also non-stationary data sources
  - Humans also bias the samples they provide
  - Humans don't share preferences or strategies







Tutorial Goals for this section

- Gain intuition for how interactive machine learning is used in robotics
- Build familiarity with the terms and techniques used in the field
- Communicate important ideas for making robots useful in practice

---

- Build a deep understanding of statistical methods at the core of leading learning from demonstration methods
- Provide implementation-level detail for these techniques
- Walkthrough correctness or convergence proofs

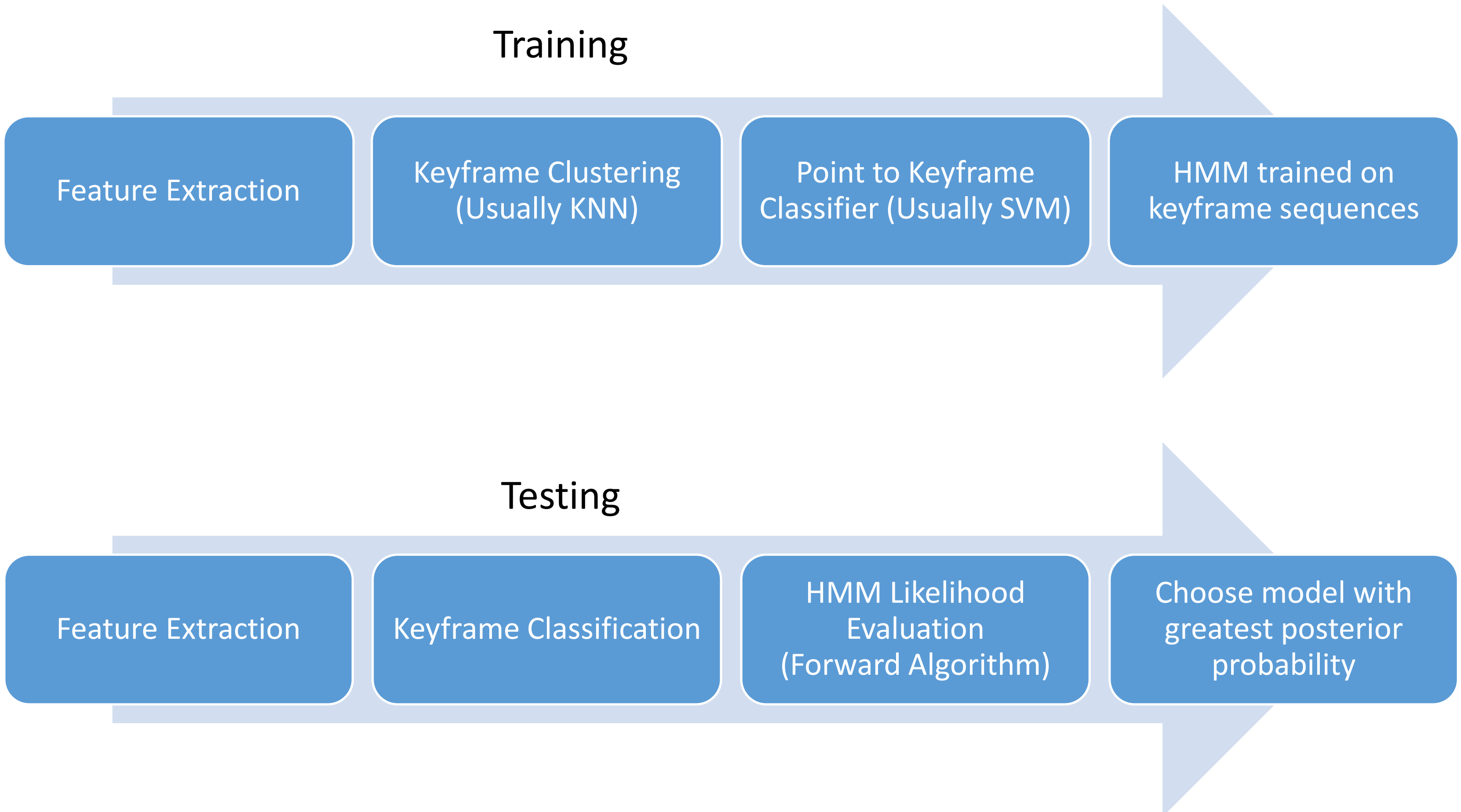
Algorithmic Human-Robot Interaction

- **Acquiring Skills and Tasks from Demonstration**
    - Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective
    - Learning and Generalization of Complex Tasks from Unstructured Demonstrations
    - Autonomously Constructing Hierarchical Task Networks for Planning and Human-Robot Collaboration
    - Towards Robot Adaptability in New Situations

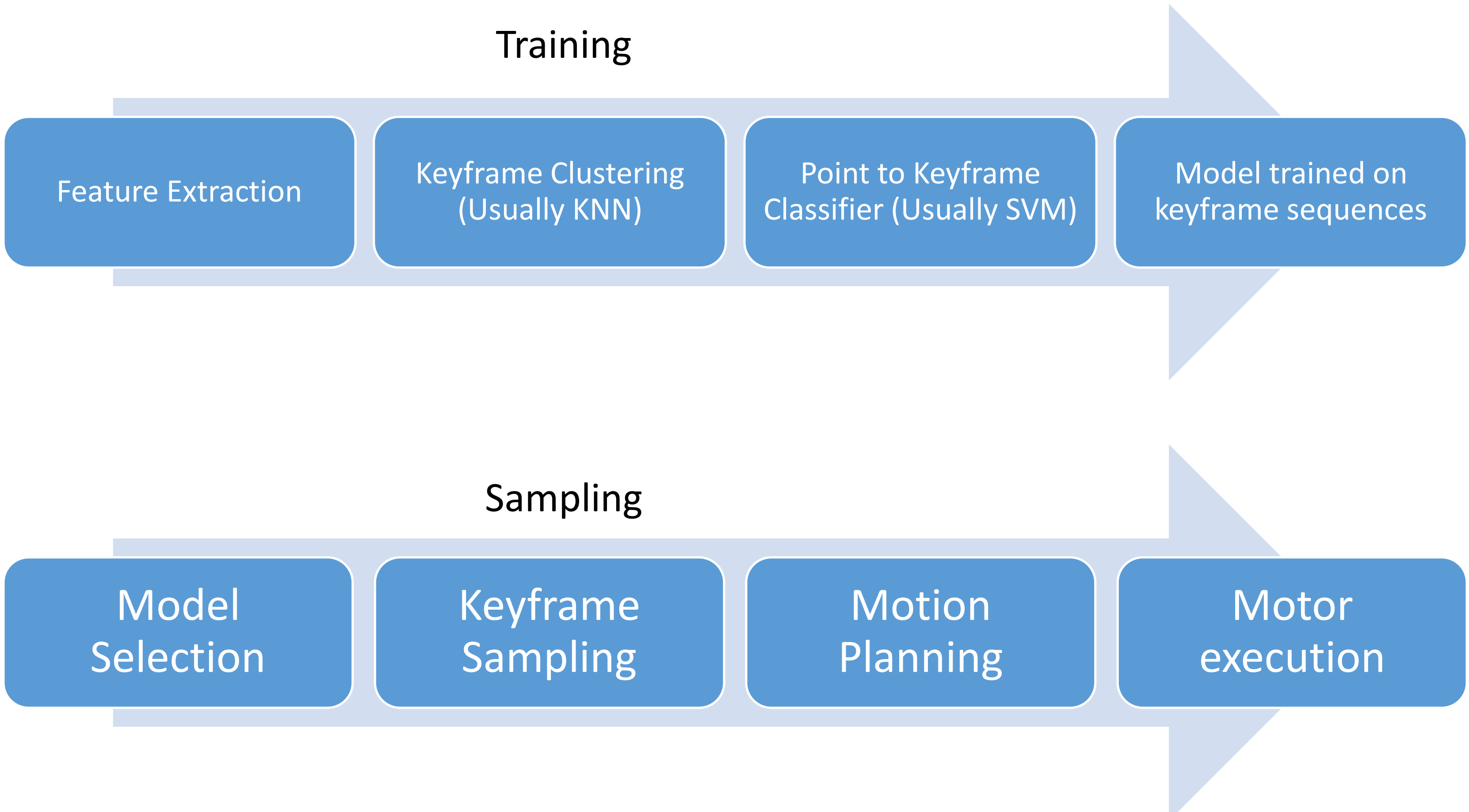
- **Cooperative Task Execution**
    - Interpretable Activity Recognition
    - Cooperative Inverse Reinforcement Learning
    - Game-Theoretic Modeling of Human Adaptation in Human-Robot Collaboration
    - Effective Robot Teammate Behaviors for Supporting Sequential Manipulation Tasks
    - Improving Robot Controller Transparency Through Autonomous Policy Explanation

- **Interaction Design:**
    - Designing Interactions for Robot Active Learners

Activity Recognition Workflow



Activity Generation Workflow



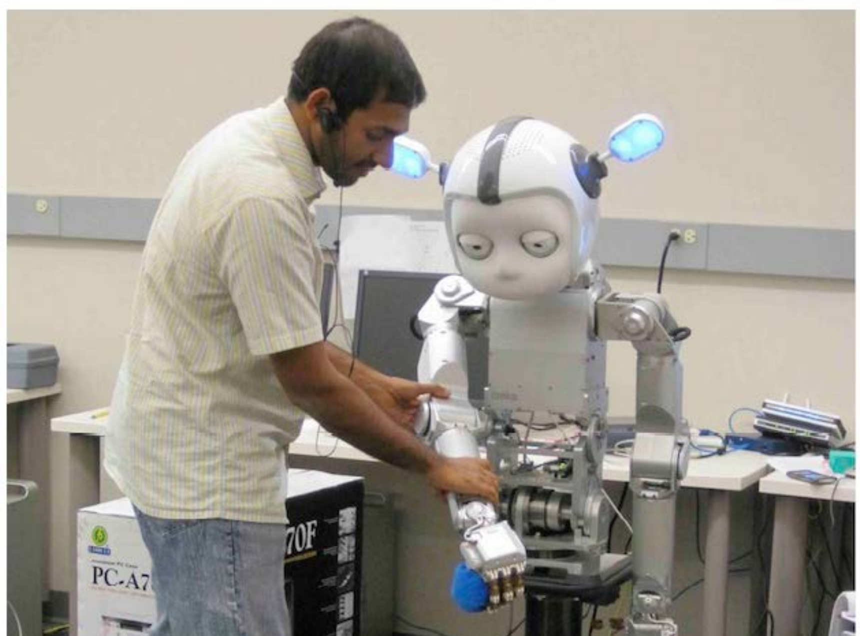
Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective

[HRI 2012]

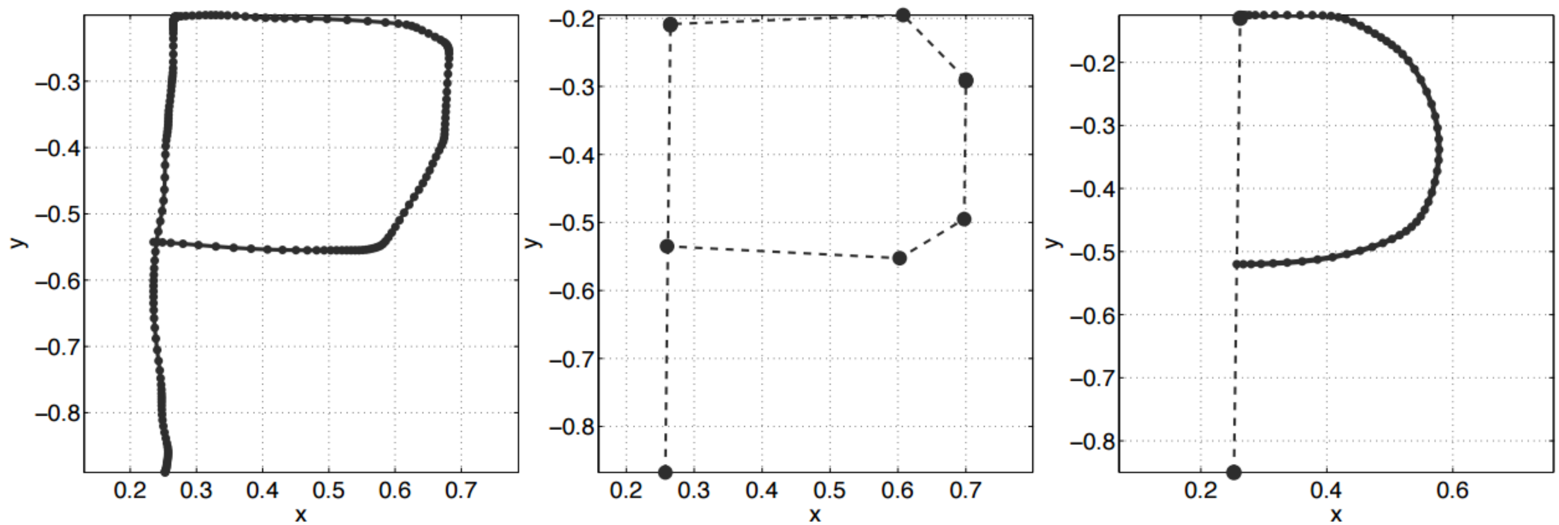
Baris Akgun, Maya Cakmak, Jae Wook Yoo, Andrea L. Thomaz

Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective

- Multiple methods exist for skill learning on a robot
- Kinesthetic teaching removes the correspondence problem
- When is it appropriate to perform trajectory-based learning?
- When is it appropriate to perform keyframe-based learning?



Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective



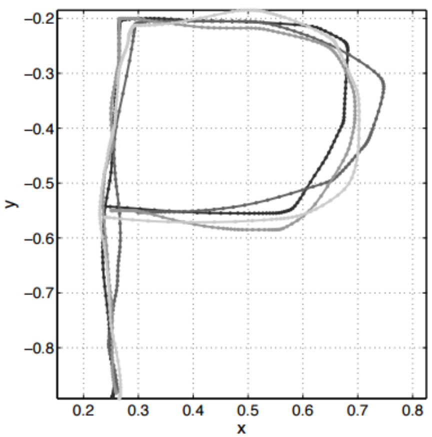
Trajectory Demonstration

Keyframe Demonstration

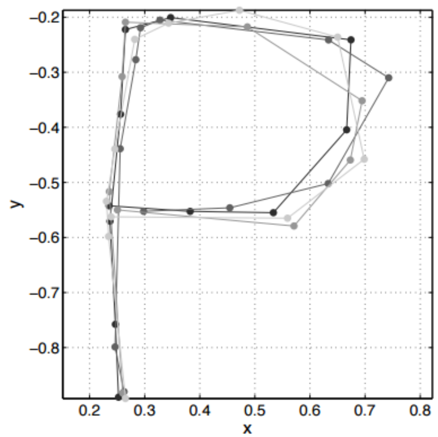
Hybrid Demonstration

Sample demonstrations of the letter P in 2D

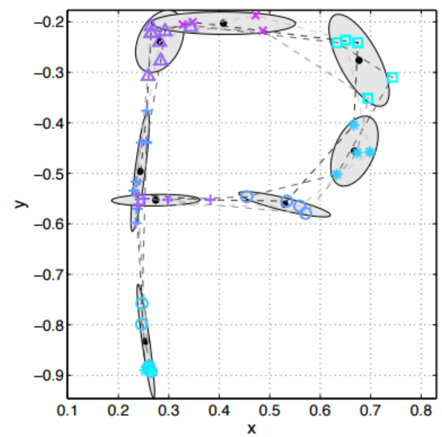
Trajectory Conversion



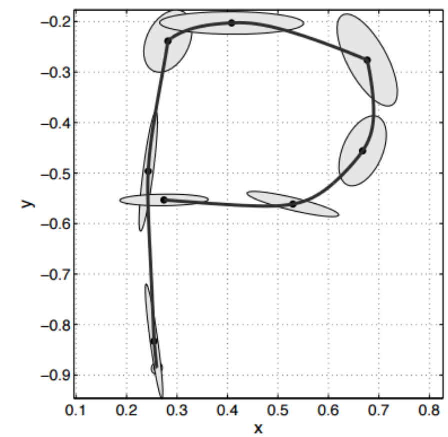
Continuous trajectories in 2D



Data converted to keyframes



Clustering of keyframes and the sequential pose distributions



Learned model trajectory

Trajectory Conversion: Forward-Inverse Relaxation Model

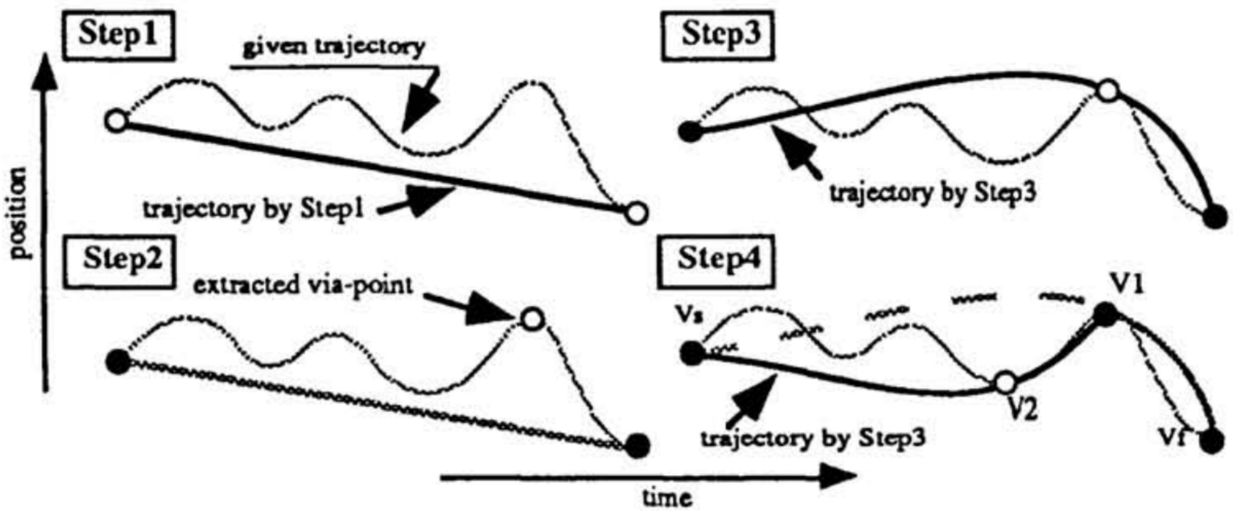
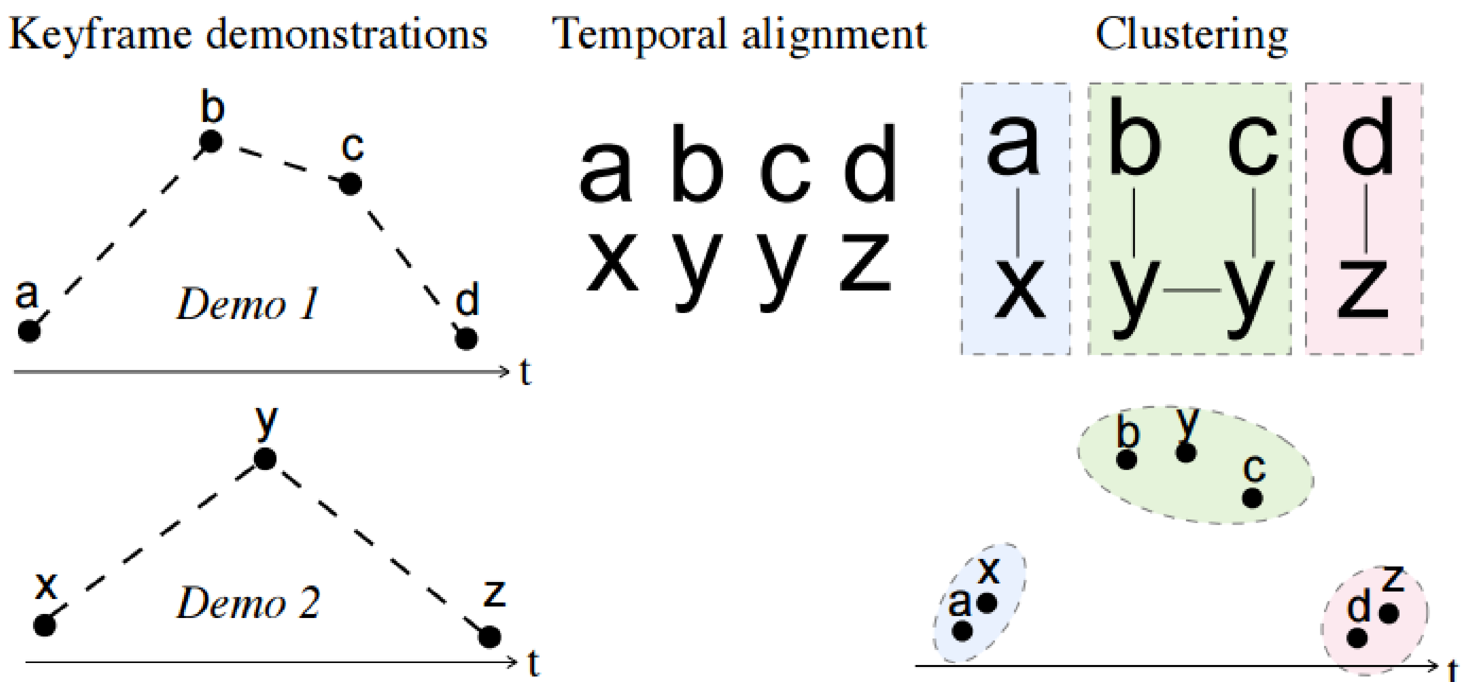


Figure 3: An algorithm for extracting via-points.

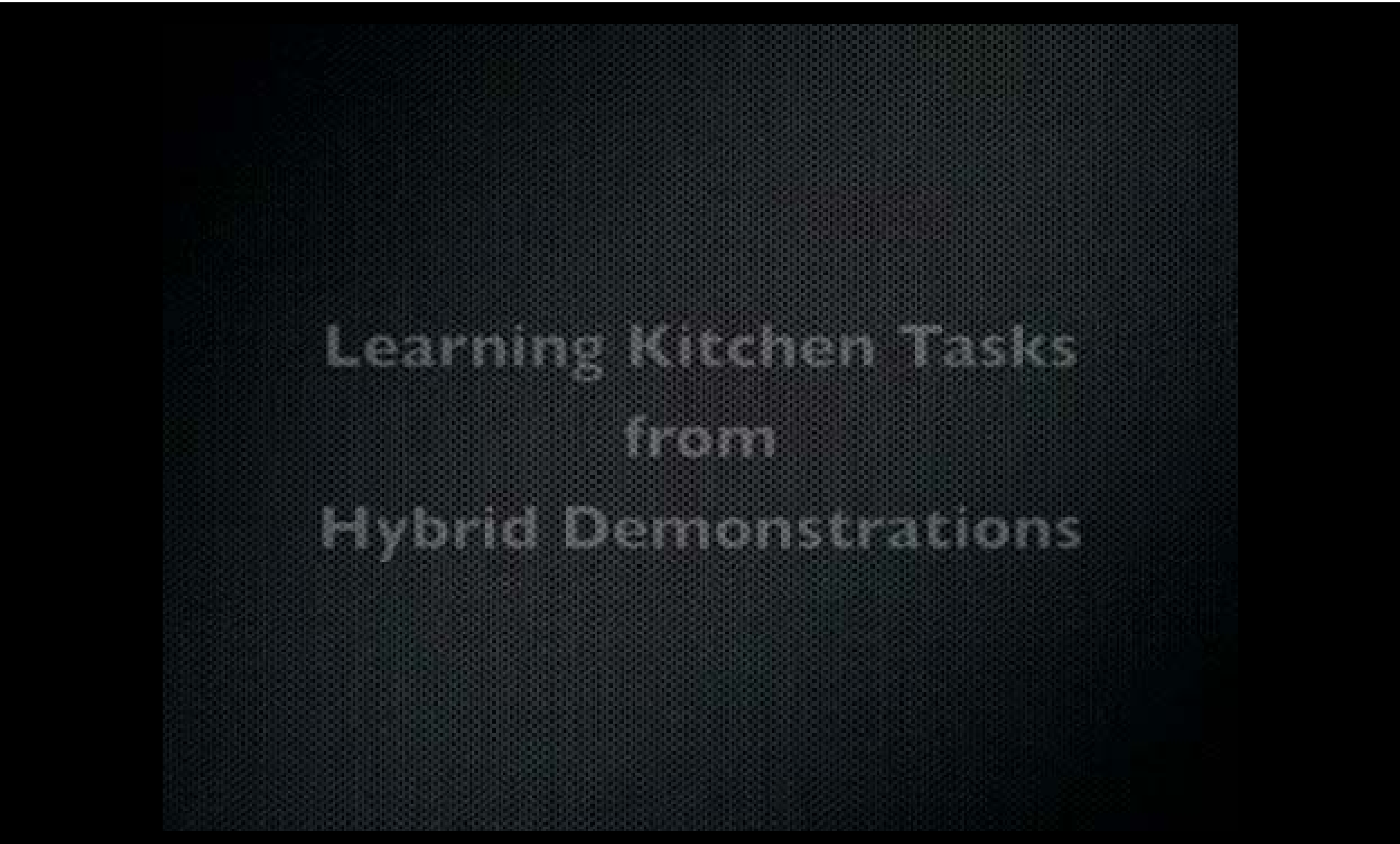
- Fifth order splines used between positions to minimize jerk, using position, velocity, and acceleration per keyframe to compute the spline unknowns.
- Keyframes assume zero velocity/acceleration per point
- Trajectory demonstrations use the means from cluster centers.

A Computational Model for Cursive Handwriting Based on the Minimization Principle – Wada et al.

Aligning Multiple Demonstrations



Implementation on PR2



Non-monolithic Task Representations

Can we learn and generalize multi-step tasks?

- Supports “life-long learning”
- Avoids dependency on isolated skill learning
  - Expensive to require human attention and demonstration
- Automatic segmentation allows for better skill transfer

Can we impart more complicated feature spaces into our skill representations without sacrificing usability?

Skill Learning Wishlist

Recognize repeated instances of skills and generalize them to new settings.

Segment data without a priori knowledge of task structure.

Identify broad, general classes of skills
(eg., manipulations, gestures, goal-based actions.)

Skill policies should have a flexible encoding such that they can be improved over time.

Learning and Generalization of Complex Tasks from Unstructured Demonstrations

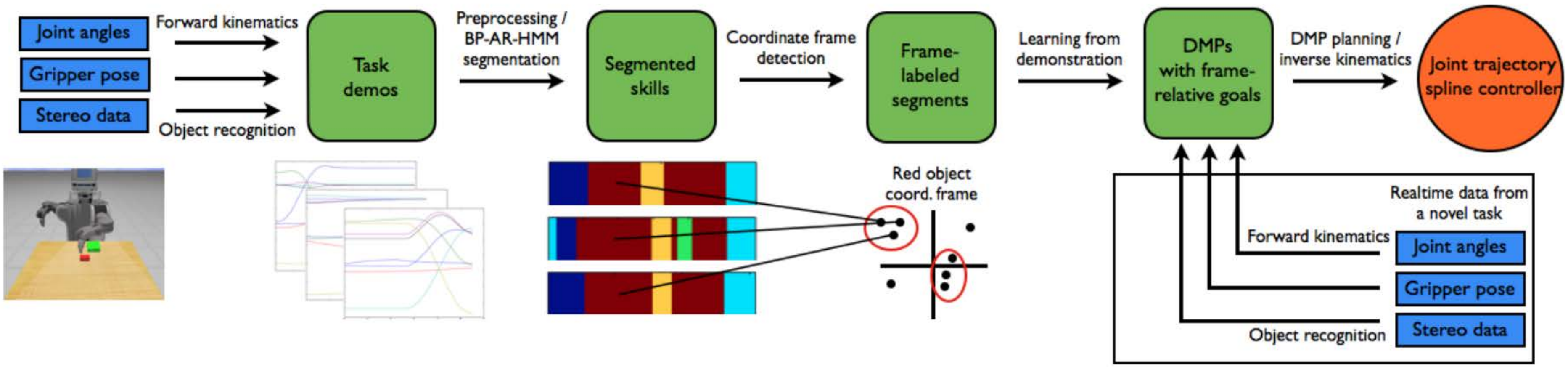
[IROS 2015 / IJRR]

Scott Niekum, Sarah Osentoski, George Konidaris, Andrew G. Barto

Learning and Generalization of Complex Tasks from Unstructured Demonstrations

- Model-free skill segmentation
  - Using Bayesian nonparametric techniques
- Rapid policy learning
  - Learning from Demonstration accelerates skill acquisition
- Activity recognition without task priors
  - Using a Beta-Process Autoregressive Hidden Markov Model
- Flexible skill encoding
  - Dynamic Movement Primitives

Task Learning Pipeline



- Task and skill representation created simultaneously from a continuous demonstration
- Recognizes re-used skills

Preprocessing/Segmentation

- Segmentation is performed using a BP-AR-HMM [1]

$$B \mid B_0 \sim \mathrm{BP}(1, B_0)$$

Draw a set of global weights for each segment

$$X_i \mid B \sim \mathrm{BeP}(B)$$

Draw Bernoulli Process params for relevant segment selection

$$\pi_j^{(i)} \mid \boldsymbol{f}_i, \gamma, \kappa \sim \mathrm{Dir}([\gamma, \ldots, \gamma + \kappa, \gamma, \ldots] \otimes \boldsymbol{f}_i)$$

Construct transition vector for segment-segment transitions

$$z_t^{(i)} \sim \pi_{z_{t-1}^{(i)}}^{(i)}$$

Select a segment for each time step based on prev. segment

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

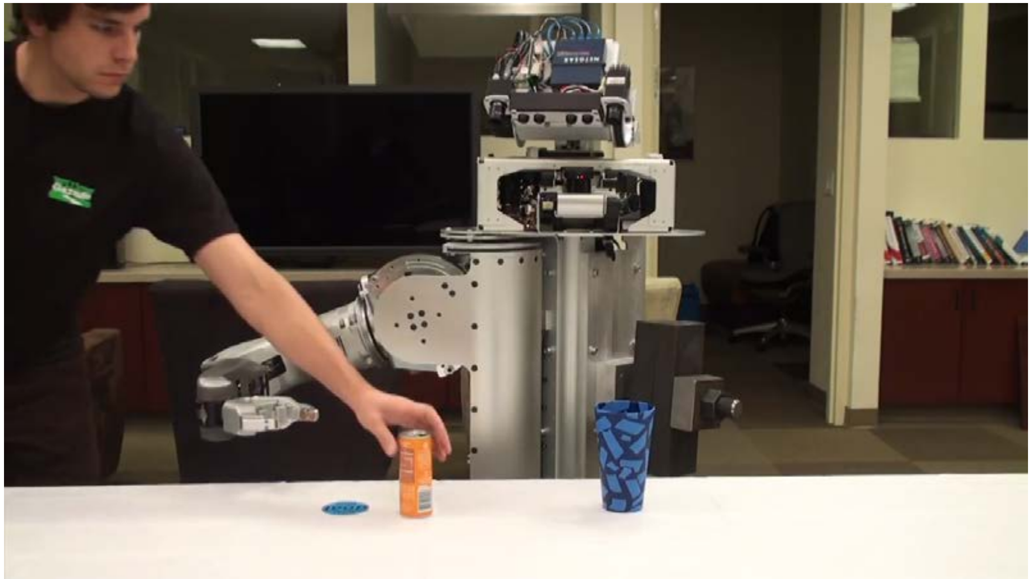
Observation computed as summed linear transforms of previous observations + noise

[1] Joint Modeling of Multiple Related Time Series via the Beta Process – Fox et al.

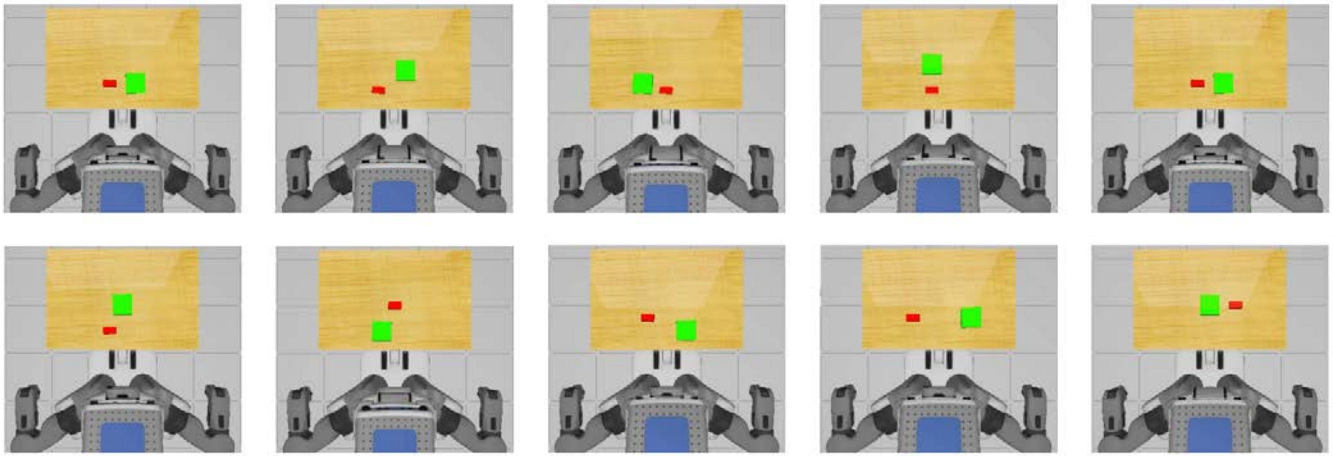
Skill Learning

- Skills are modeled as Dynamic Movement Primitives
  - Linear point attractor modulated by a nonlinear (learned) function
  - Uses end effector positions + quaternions for gripper rotation

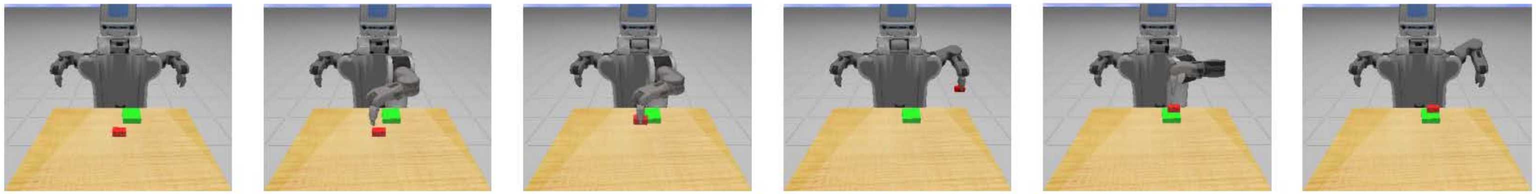
$$f_{\text{target}}(s) = \frac{-K(g - x(s)) + D\dot{x}(s) + \tau\,\ddot{x}(s)}{g - x_0}$$



Evaluation: Testing Segmentation



- Trained on demonstrations of the top task
- Tested on demonstrations of the bottom task



(a) Starting pose (b) Reaches toward red block (red block frame) (c) Picks up red block (red block frame) (d) Returns to home position (torso frame) (e) Places red block on green block (green block frame) (f) Returns to home position (torso frame)

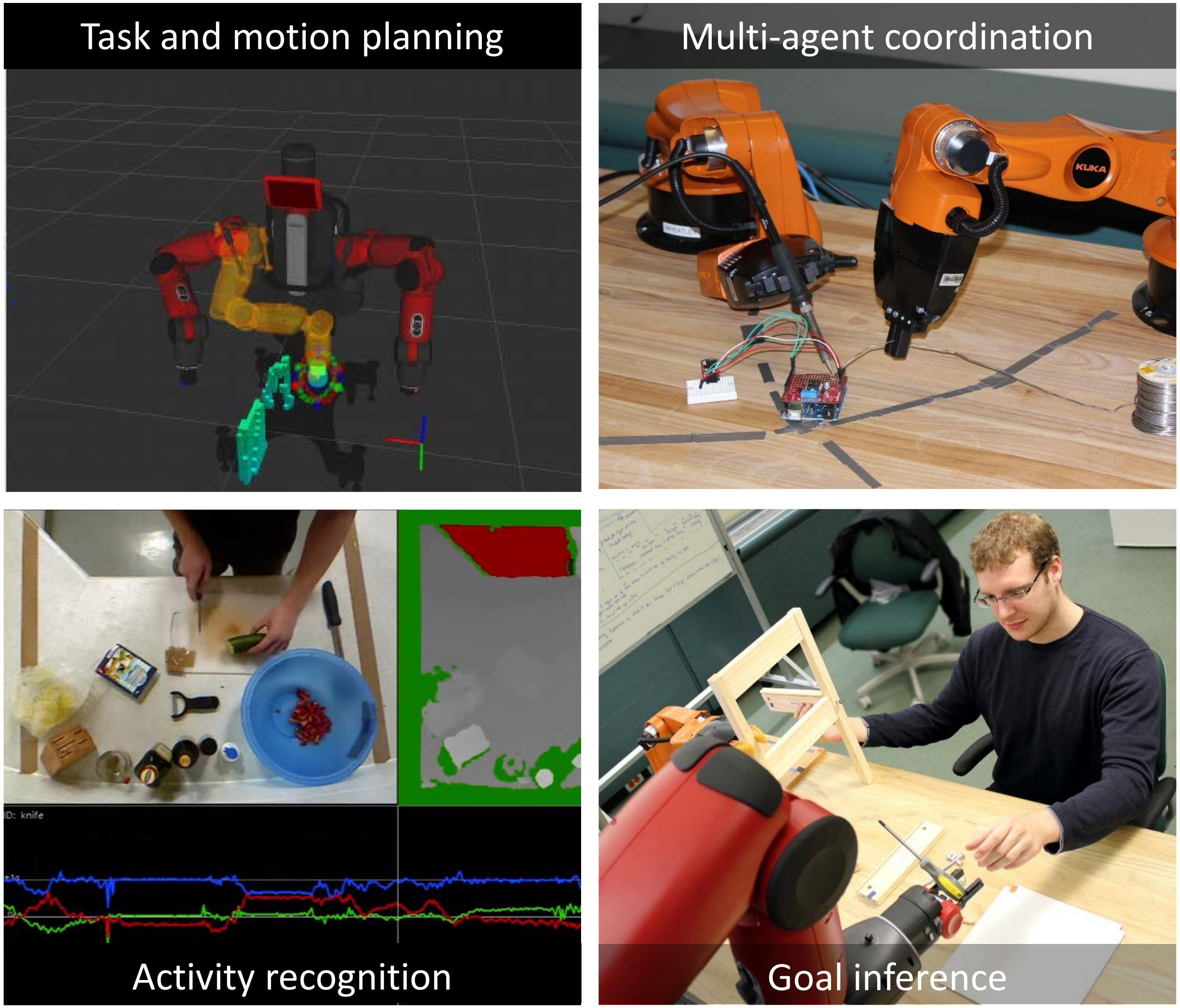
- Autonomously segmented skills and associated frames

Failure Modes

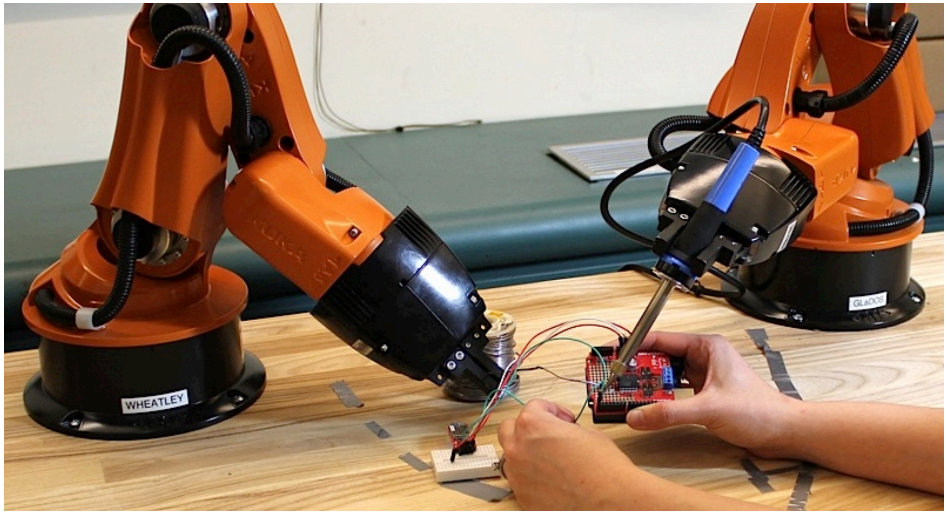
- Symbolic Failure
  - Occurs when objects in the task description cannot be resolved (e.g., are missing from the environment)
  - Remedied through suggestion of substitutions
  - Possible corrections can be accepted or rejected due to pragmatic or preferential reasons.
    - Allows propositions for which the robot does not have an object model
- Execution Failure
  - Occurs when symbolic substitutions are accepted without a model sufficient for interaction
  - Occurs when outside the known policy region of a skill

Application: Interactive Corrections

Abstraction is essential for solving complex problems



Not all robots operate in isolation



Autonomously Constructing Hierarchical Task Networks for Planning and Human-Robot Collaboration

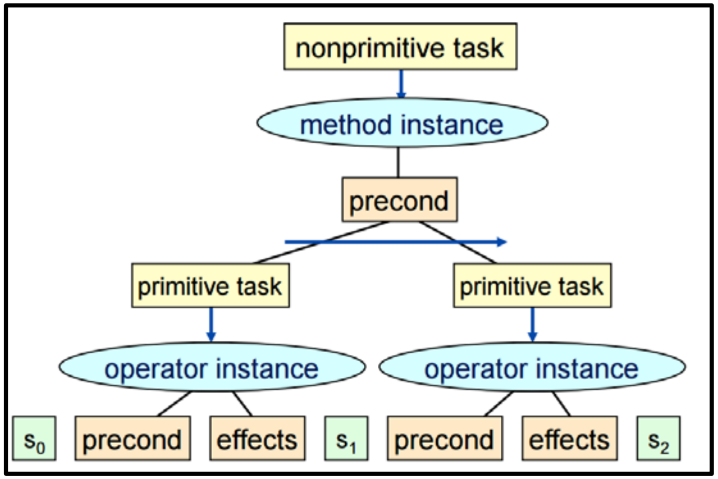
[ICRA 15]

Bradley Hayes and Brian Scassellati

Hierarchical Task Networks

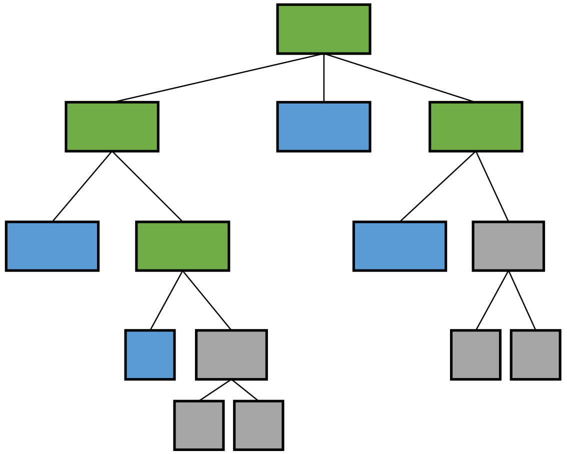
Benefits

- Defines macro actions as compositions of primitive operators
- Provides a detailed problem factorization
- Operators are defined as **(task, preconditions, effects)**
- Facilitates look-ahead for increased execution flexibility (least-commitment planning)



Challenges

- Precise specifications of preconditions and effects can be difficult to specify
- Typically defined manually



Constructing Task Networks from Demonstrations

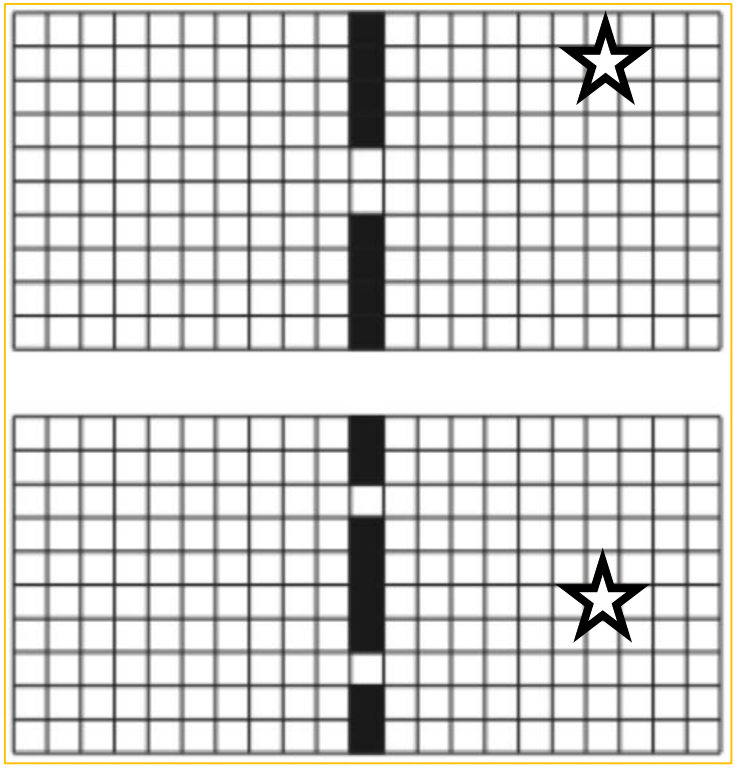
1. Extract task subgoals using min-cut

2. Convert task graph to subgoal graph

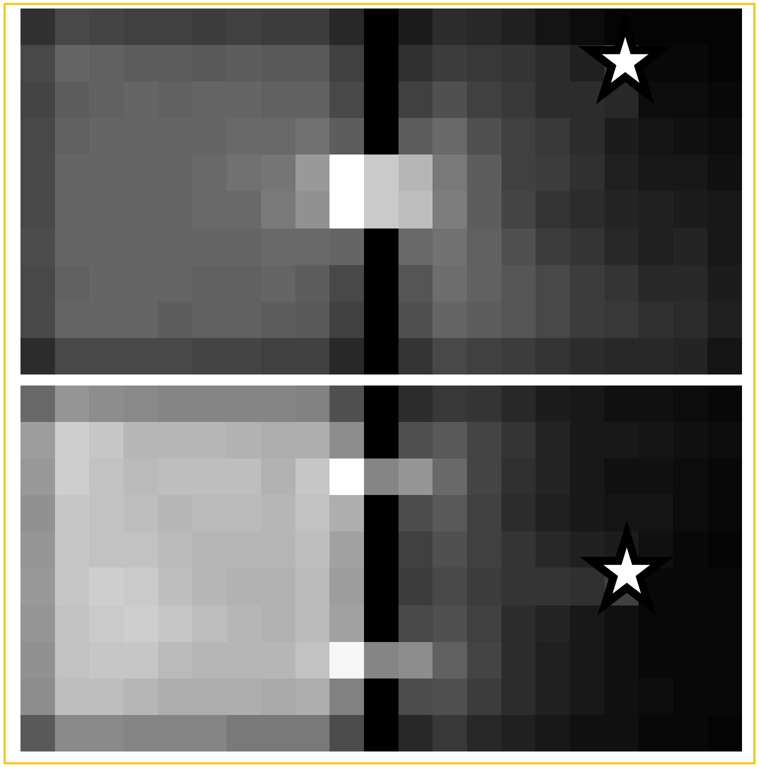
3. Apply a series of contraction operators to the subgoal graph

4. Create macro actions out of totally and partially ordered sub-plans at each iteration of contraction

Extracting Sub-Goals: Intuition – Bottleneck Recognition



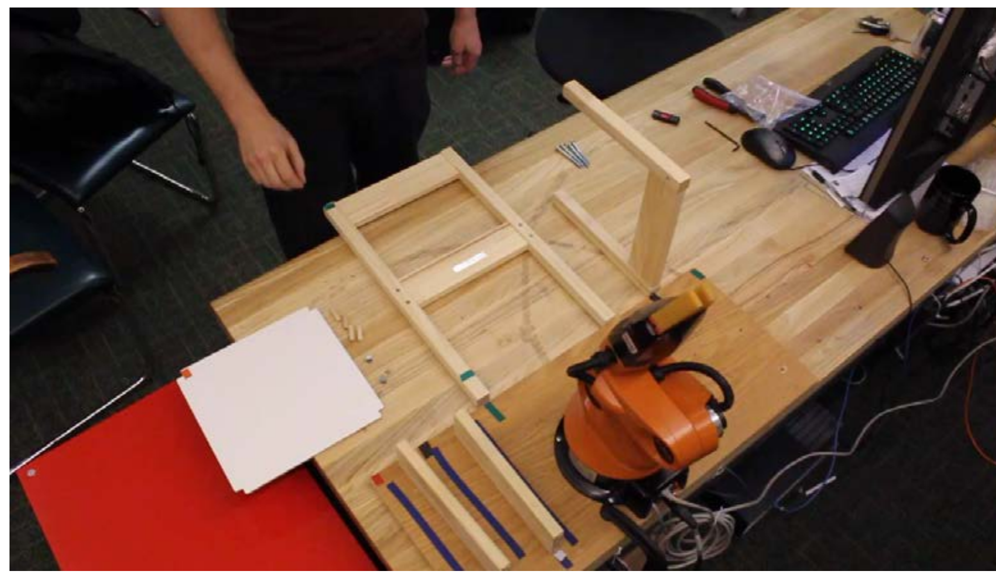
Problem Domain



State Frequency Map

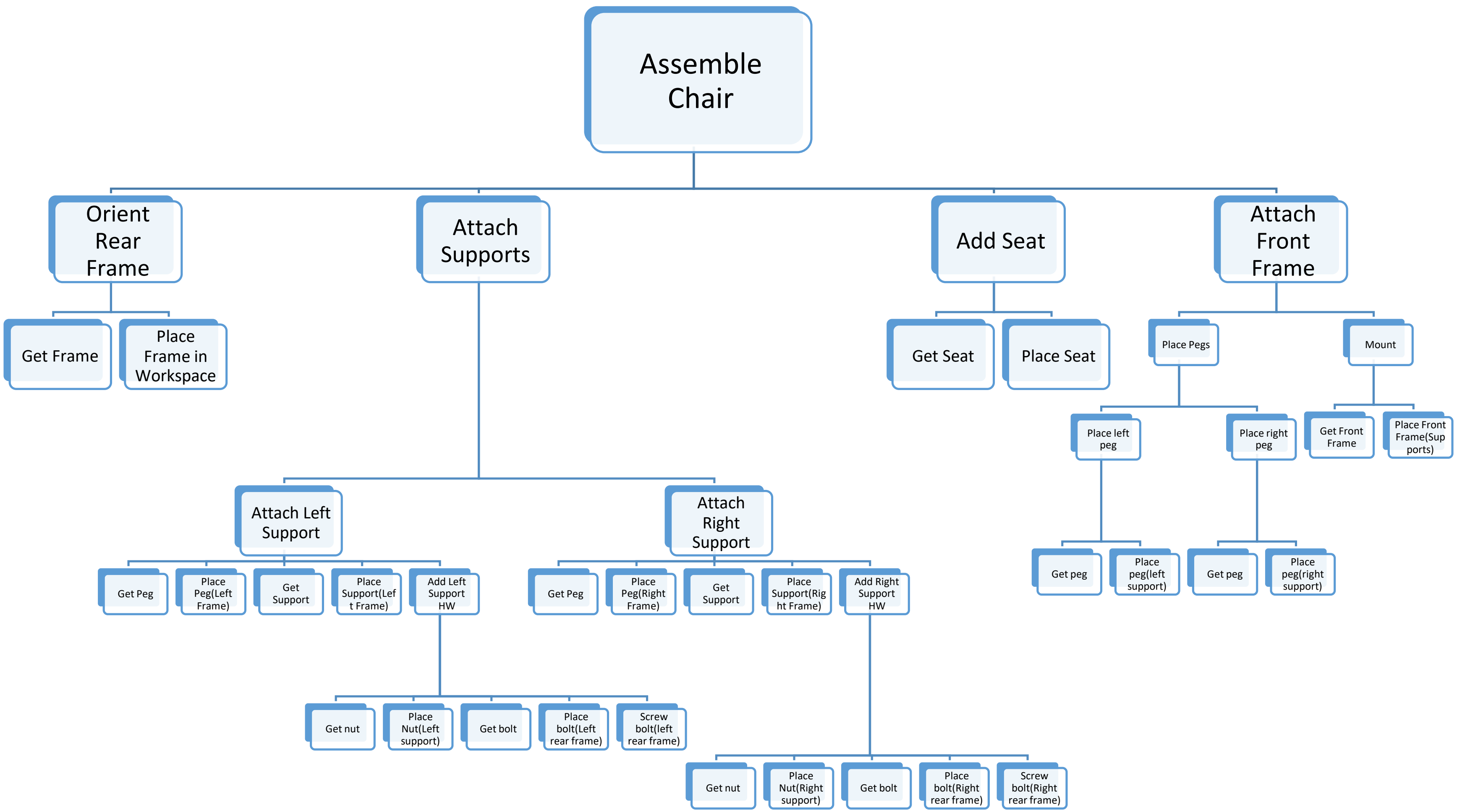
[Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning. Menache et al. 2002]

Application Domain - IKEA furniture



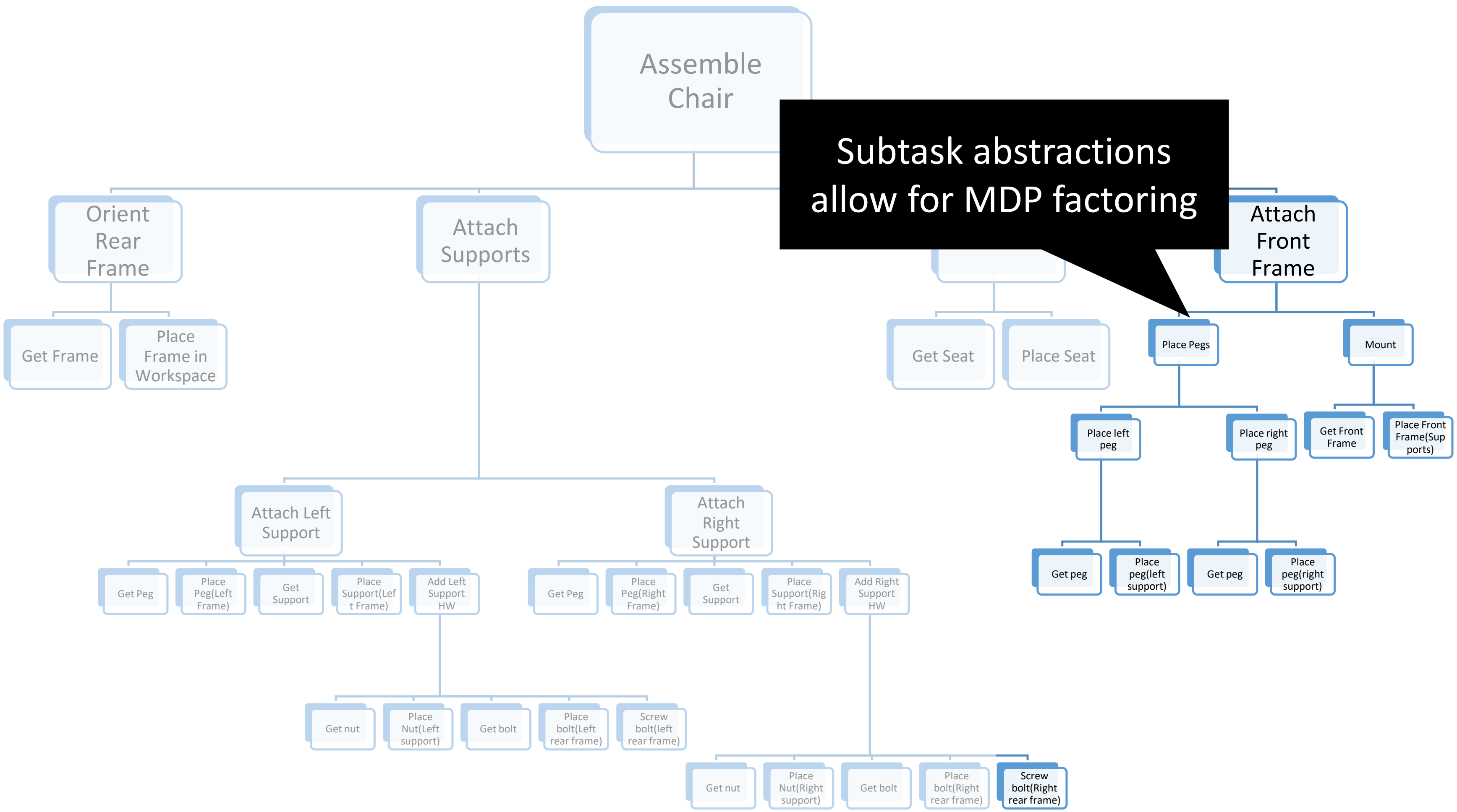
Hierarchical Task Structure

IKEA Chair

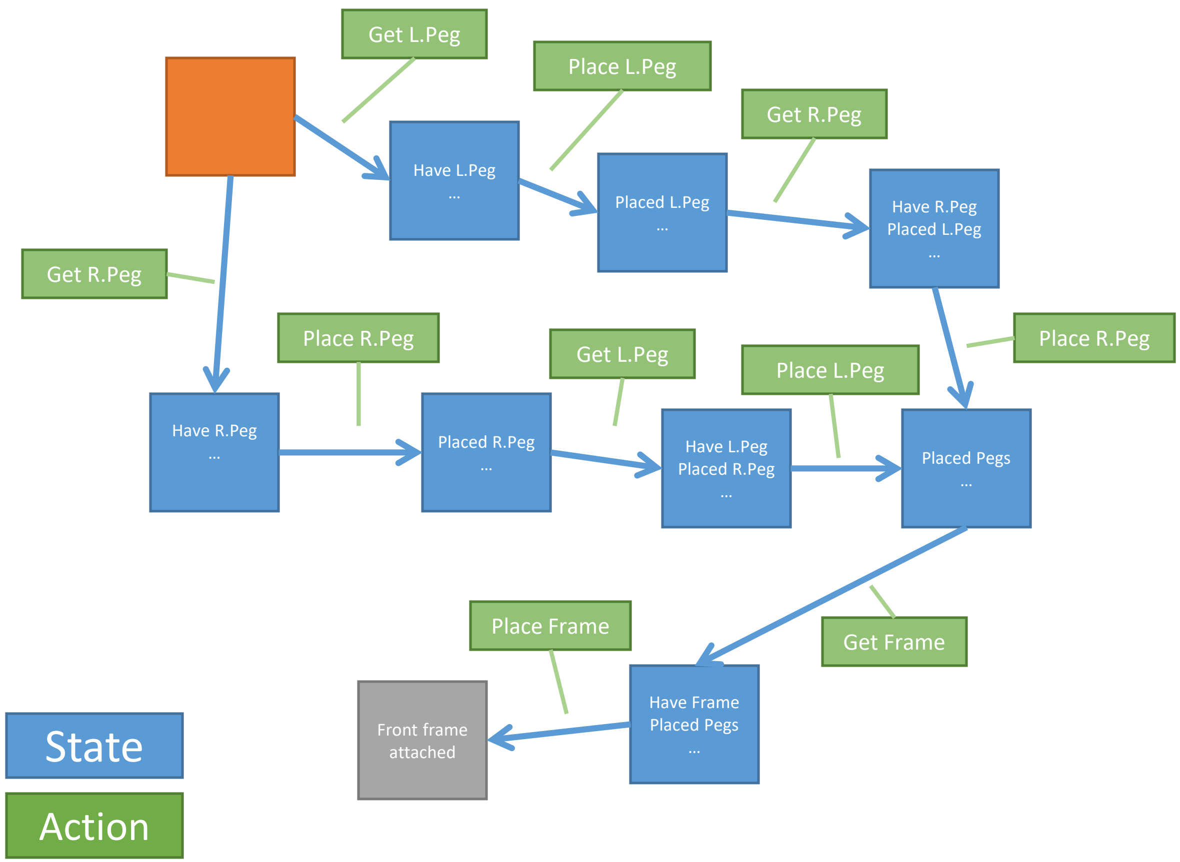


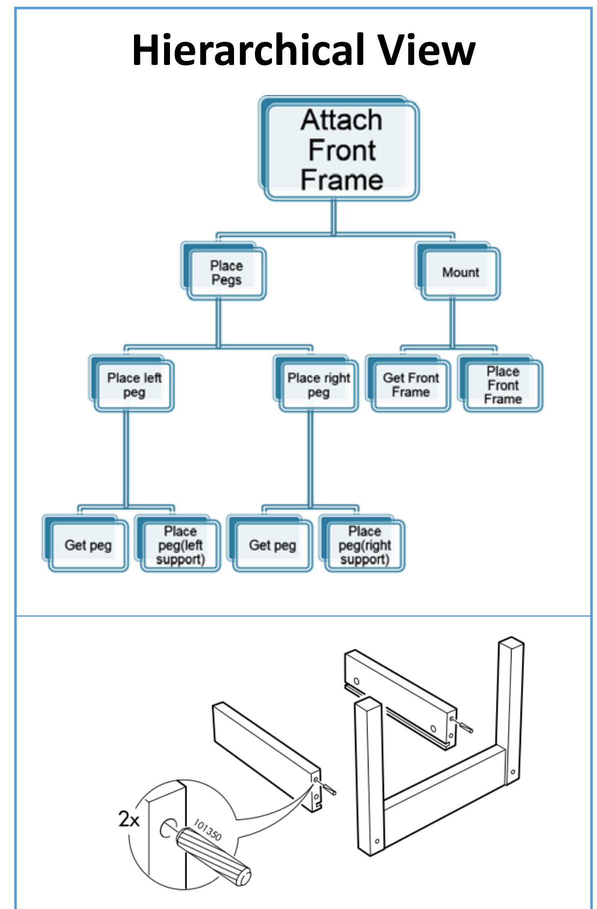
Hierarchical Task Structure

IKEA Chair

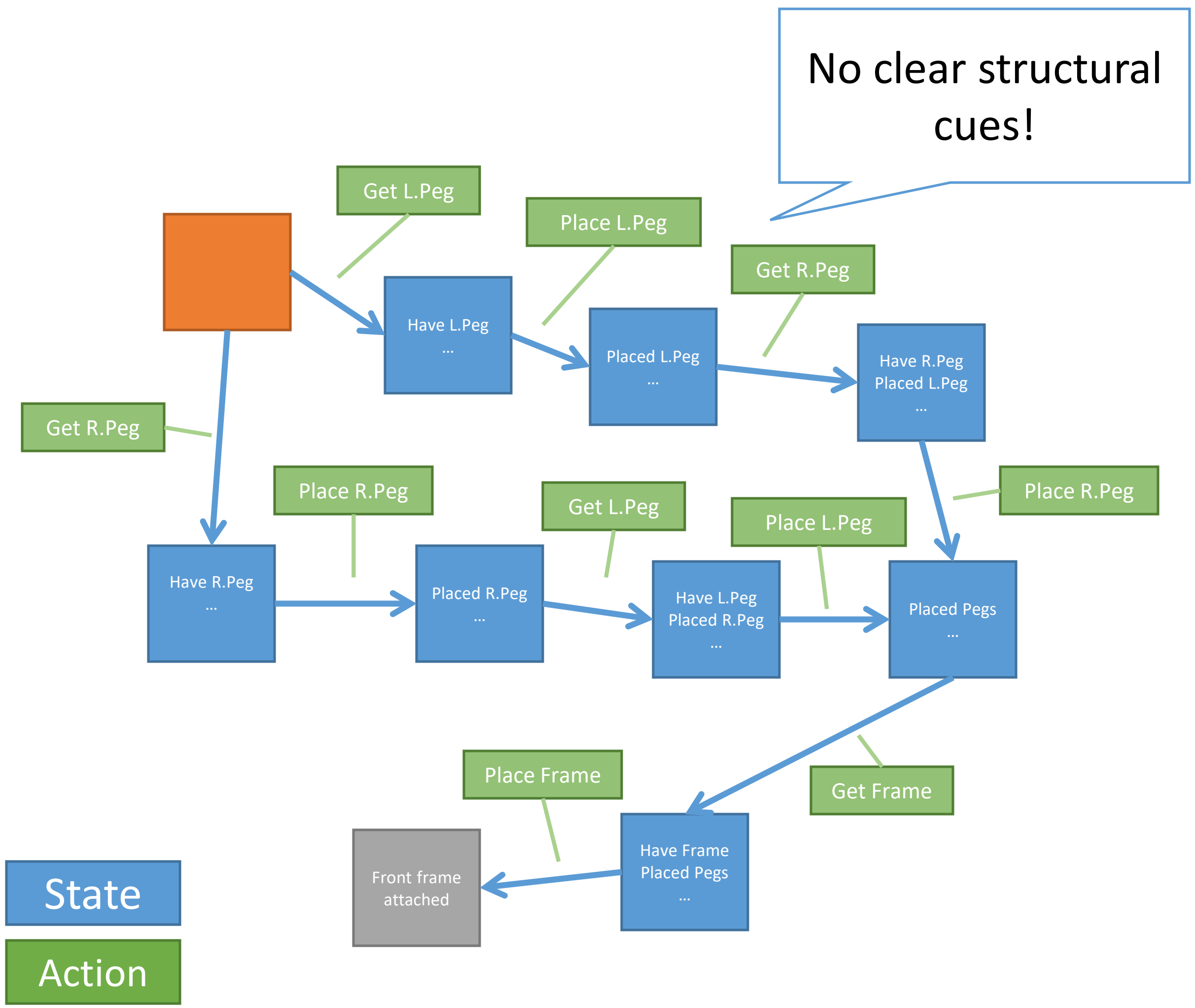


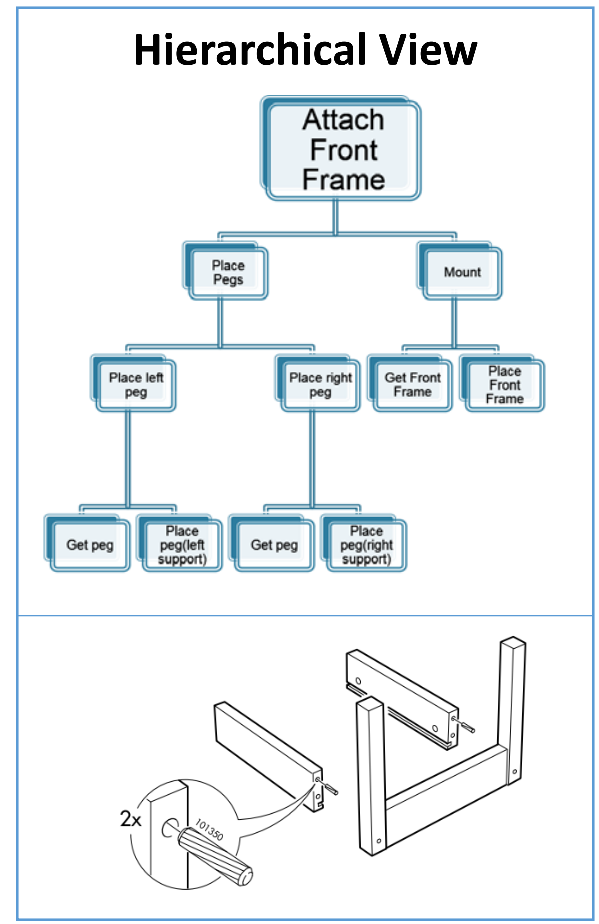
SMDP of “Attach Front Frame” Subtask



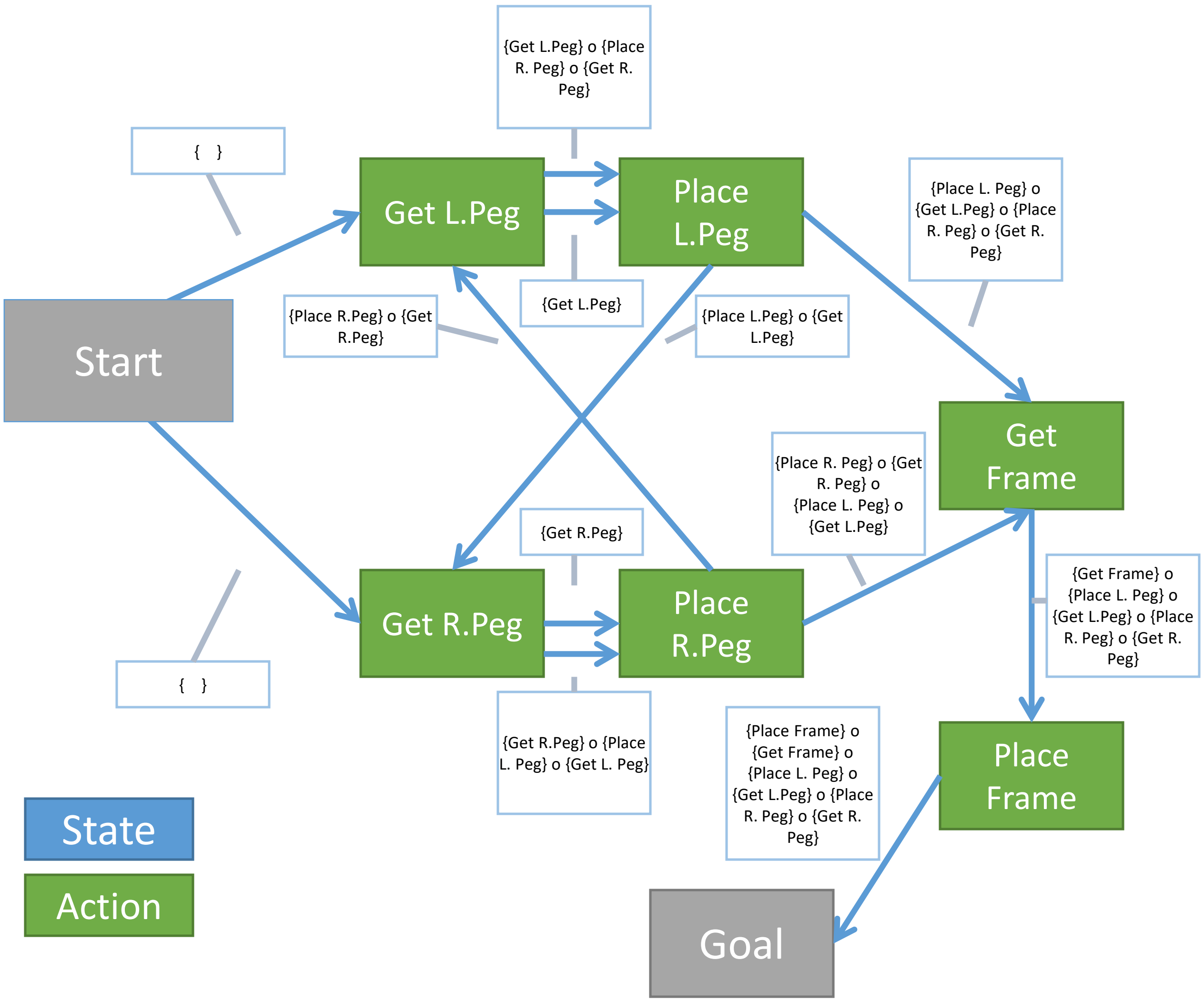


SMDP of “Attach Front Frame” Subtask





SMDP-Conjugate of “Attach Front Frame” Subtask

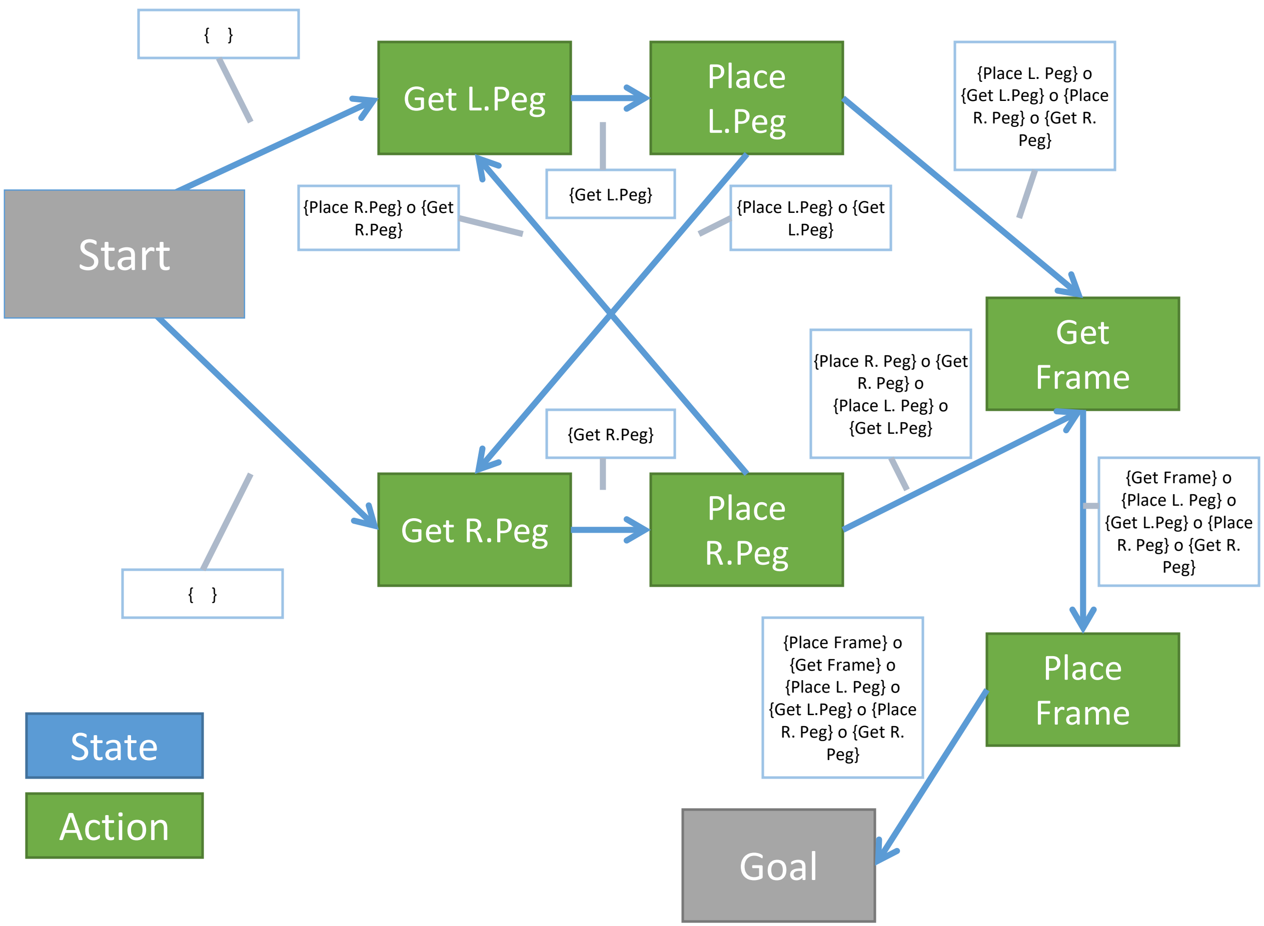


Conjugate Graph Transform Algorithm

Input: SMDP-like Graph $G = \{V, E\}$
Output: Conjugate Task Graph C

$C \leftarrow$ empty Conjugate Task Graph {W,F};
origin_vertex $\leftarrow$ new empty vertex;
terminal_vertex $\leftarrow$ new empty vertex;
Add origin_vertex and terminal_vertex to W;
foreach *unique* $a \in \{action(e) | \forall e \in E\}$ **do**
  Add vertex $\{$action $= a\}$ to W;
foreach *edge* $e \in E$ **do**
  if *source*$(e) \in initiation\_states(G)$ **then**
    Add {from: origin_vertex, to: $[v \in W \ni action(v) = action(e)]$, prerequisites: $\emptyset\}$ to F;
  if *dest*$(e) \in termination\_states(G)$ **then**
    Add {from: $[v \in W \ni action(v) = action(e)]$, to: terminal_vertex, prerequisites: environment_state(dest(e))} to F;
  else
    foreach *edge* $f \in outbound\_edges(dest(e))$ **do**
      Add {from: [vertex $v \in W \ni action(v) = action(e)]$, to: $[u \in W \ni action(u) = action(f)]$, prerequisites: environment_state(to(e))};
foreach $a, b \in F \ni a \neq b$ **do**
  if *from(a) = from(b) and to(a) = to(b) and prerequisites(a)* $\subseteq$ *prerequisites(b)* **then** delete b;

Building Hierarchical Structure



Building a constraint-based hierarchy

Goal:
Exploit existing structure to find logical groupings of sub-tasks

```
Task Graph to HTN Transform
Input: SMDP-like Graph G
Output: HTN H
H ← Conjugate-Graph-Transform(G)
while |H_vertices| > 1 do
    h_size ← |H_vertices|
    foreach maximal clique c = {V, E} ∈ H do
        Compact {V ∈ c} into single metanode m
    foreach maximal chain c = {V, E} ∈ H do
        Compact {V ∈ c} into single metanode m
    if h_size == |H_vertices| then break;
```

Step 0: Start Algorithm

Building Hierarchical Structure

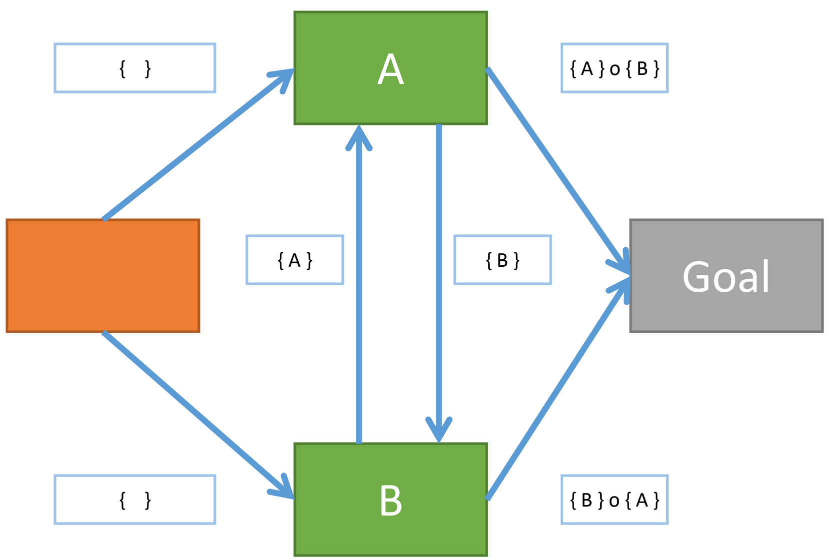
Task Graph to HTN Transform

Input: SMDP-like Graph G
Output: HTN H

$H \leftarrow$ Conjugate-Graph-Transform(G)
while $|H_{vertices}| > 1$ **do**
  $h\_size \leftarrow |H_{vertices}|$
  foreach *maximal clique* $c = \{V, E\} \in H$ **do**
    Compact $\{V \in c\}$ into single metanode m
  foreach *maximal chain* $c = \{V, E\} \in H$ **do**
    Compact $\{V \in c\}$ into single metanode m
  if $h\_size == |H_{vertices}|$ **then** break;

Goal

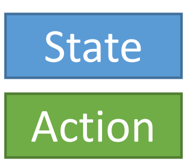
Cliques



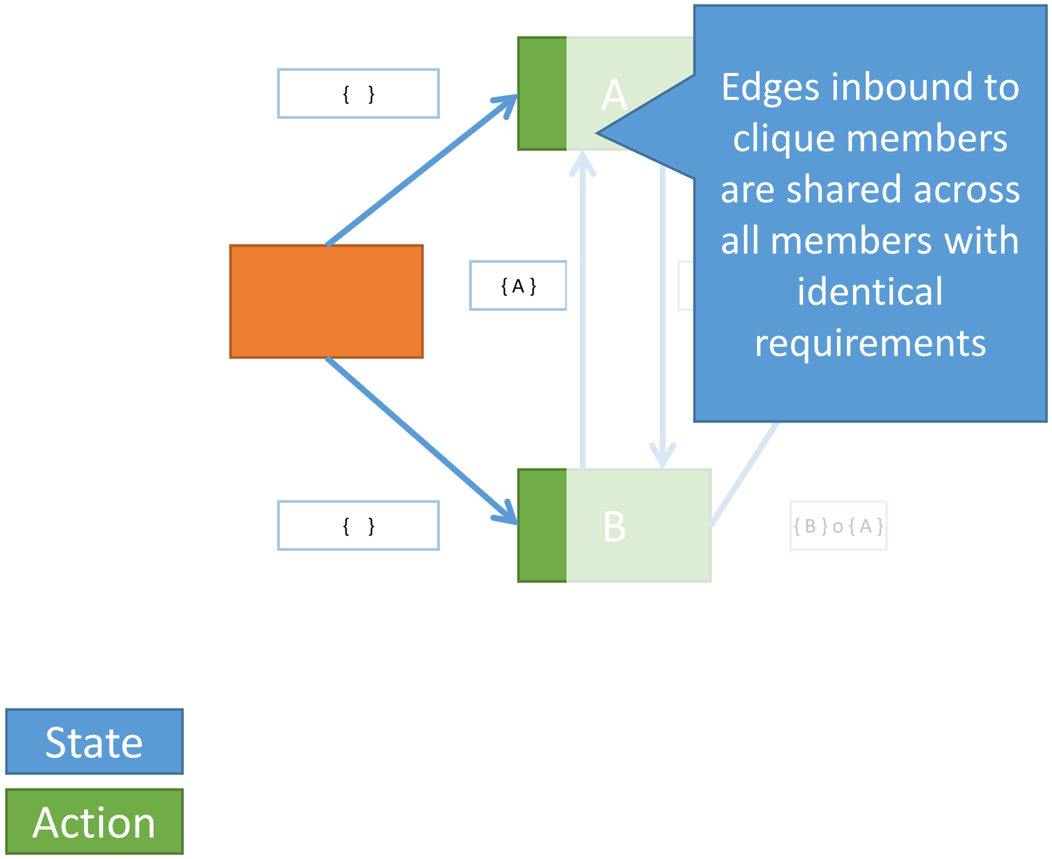
Any edges inbound to a clique member must have identical inbound edges to *all* clique members.

Any edge outbound from a clique member to an external vertex must have identical outbound edges from *all* clique members to the same target.

All internal nodes must be connected without internal ordering constraints (only source vertex's postconditions can be on the edge requirements).



Cliques

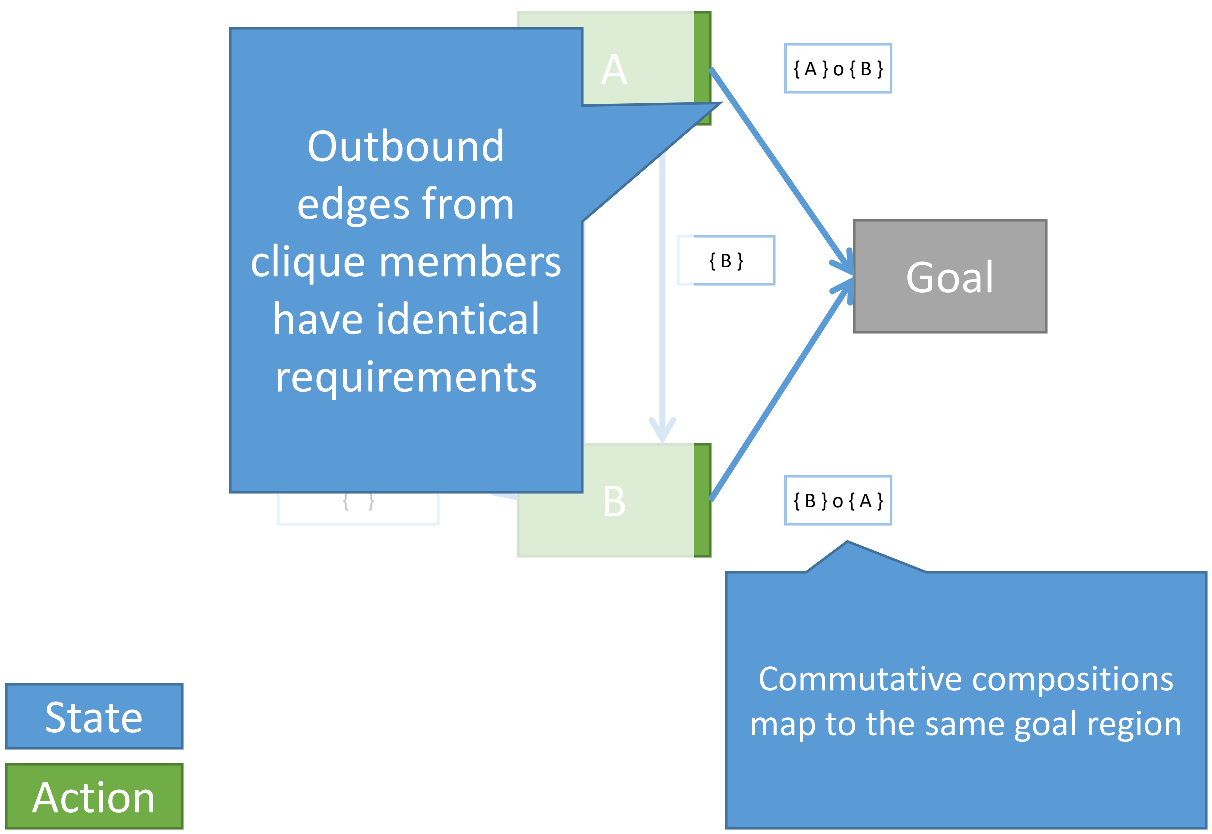


Any edges inbound to a clique member must have identical inbound edges to *all* clique members.

Any edge outbound from a clique member to an external vertex must have identical outbound edges from *all* clique members to the same target.

All internal nodes must be connected without internal ordering constraints (only source vertex's postconditions can be on the edge requirements).

Cliques

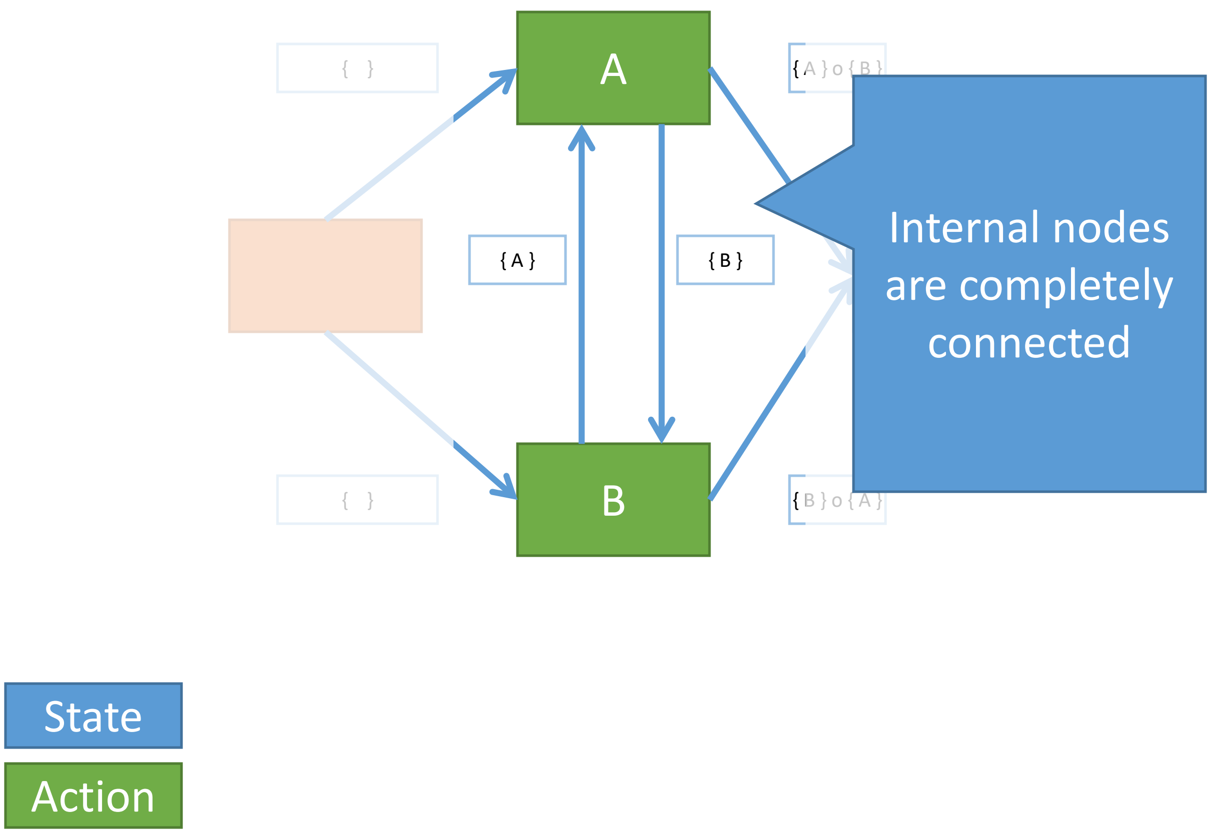


Any edges inbound to a clique member must have identical inbound edges to *all* clique members.

Any edge outbound from a clique member to an external vertex must have identical outbound edges from *all* clique members to the same target.

All internal nodes must be connected without internal ordering constraints (only source vertex's postconditions can be on the edge requirements).

Cliques

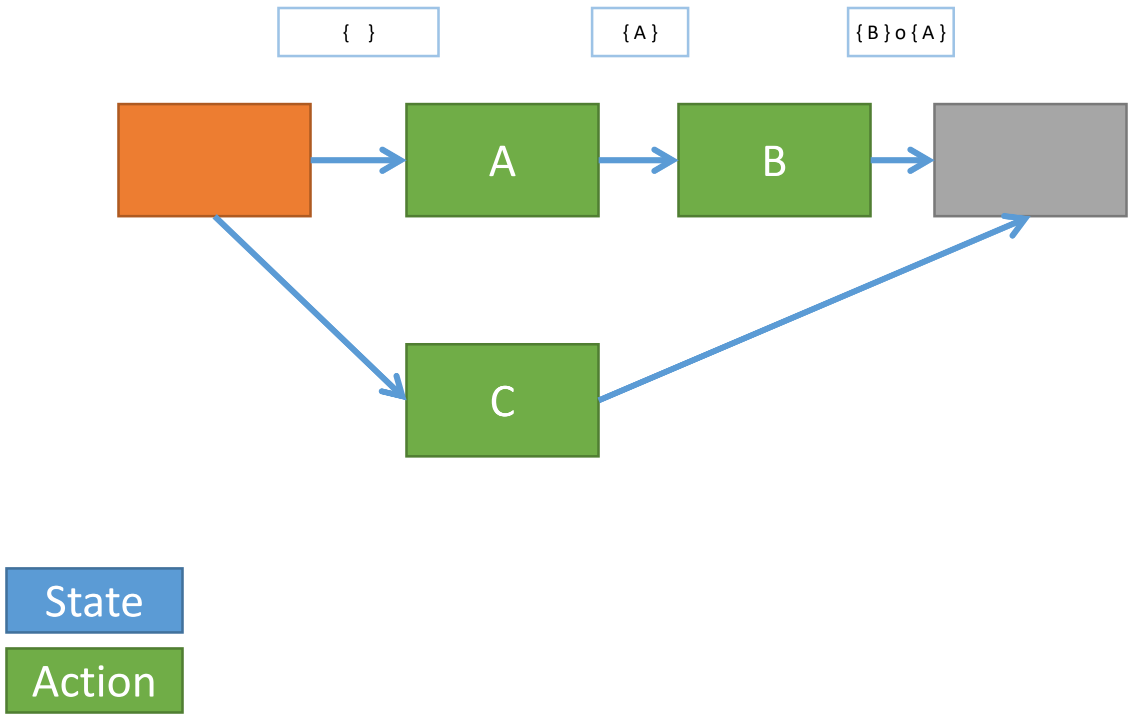


Any edges inbound to a clique member must have identical inbound edges to *all* clique members.

Any edge outbound from a clique member to an external vertex must have identical outbound edges from *all* clique members to the same target.

All internal nodes must be connected without internal ordering constraints (only source vertex's postconditions can be on the edge requirements).

Chains

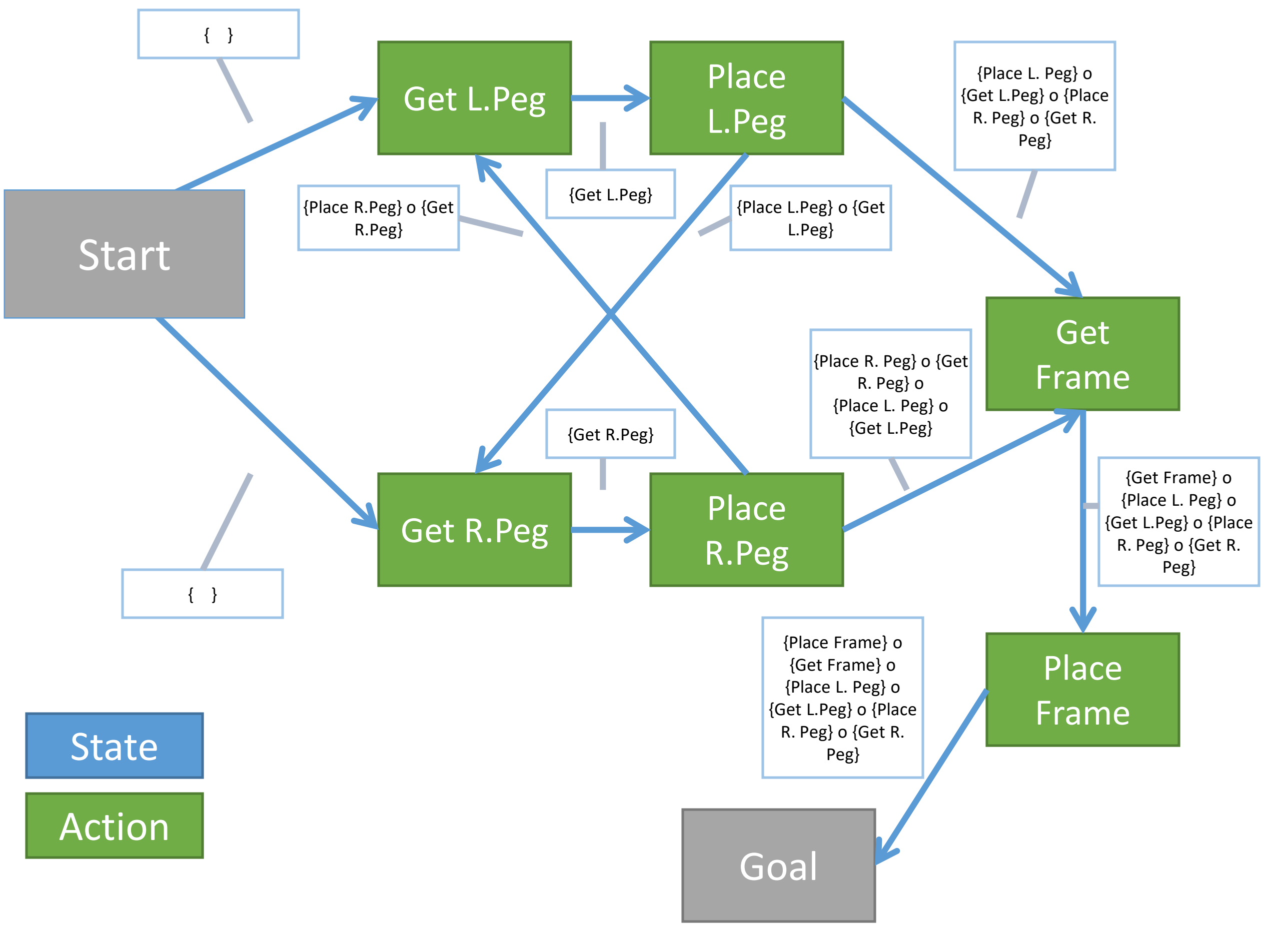


Any edges inbound to a chain must only connect to the chain's starting vertex.

All internal nodes must have in and out degree 1.

Any edges outbound from the chain must only originate from the chain's terminating vertex.

Building Hierarchical Structure



Building a constraint-based hierarchy

Goal:
Exploit existing structure to find logical groupings of task steps (sub-tasks)

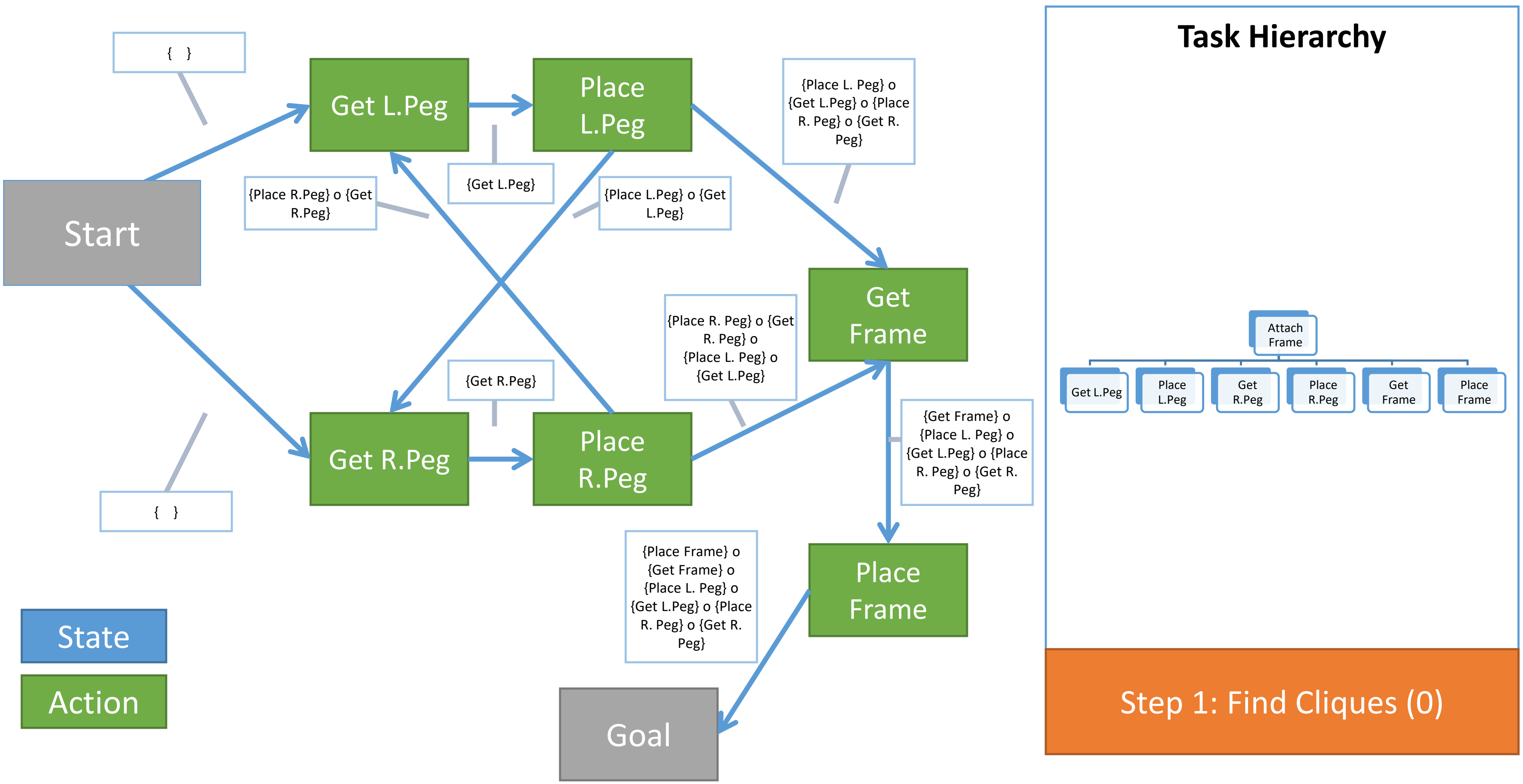
Task Graph to HTN Transform

Input: SMDP-like Graph G
Output: HTN H

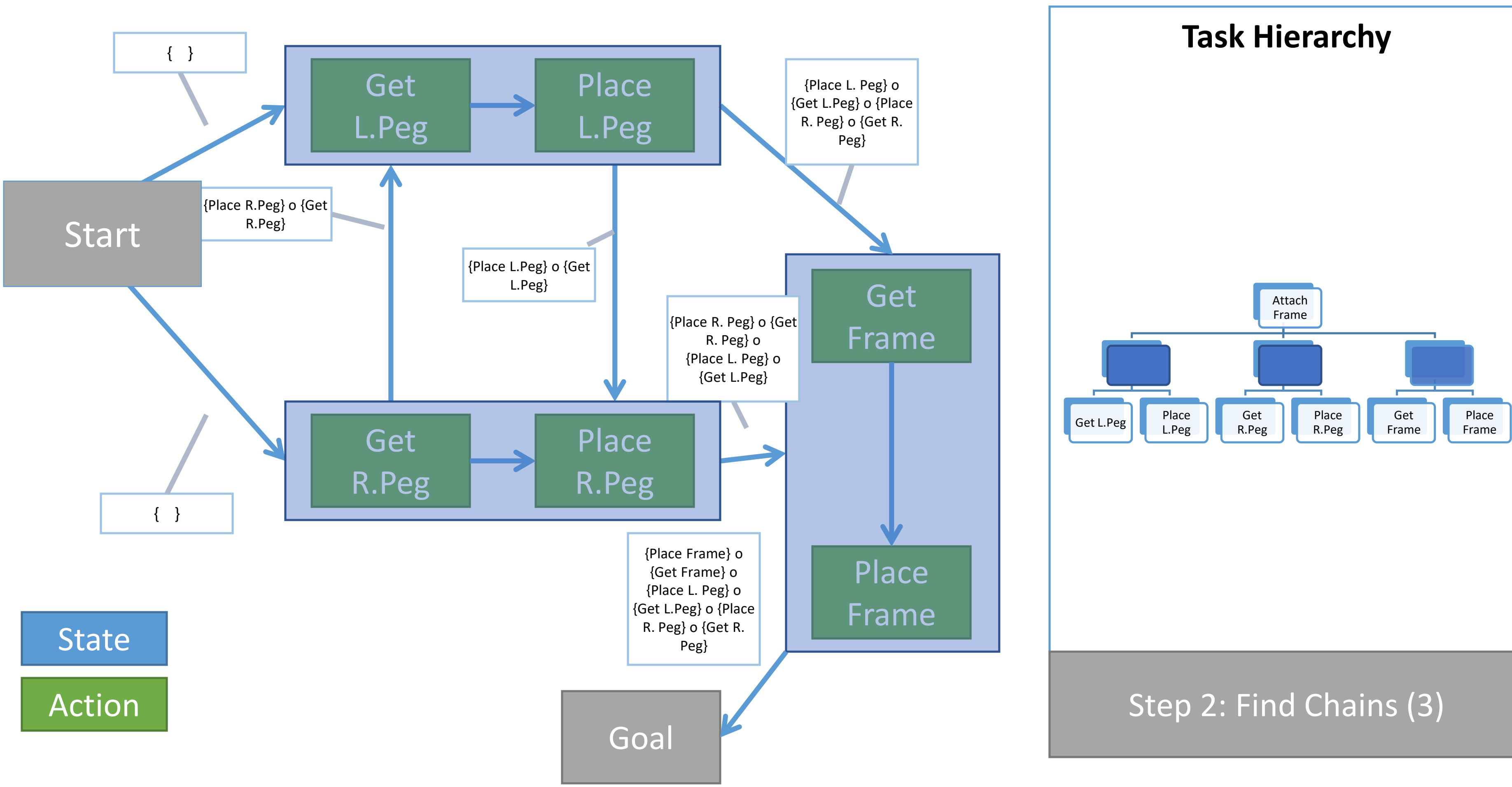
$H \leftarrow$ Conjugate-Graph-Transform(G)
while $|H_{vertices}| > 1$ **do**
  $h\_size \leftarrow |H_{vertices}|$
  foreach *maximal clique* $c = \{V, E\} \in H$ **do**
    Compact $\{V \in c\}$ into single metanode m
  foreach *maximal chain* $c = \{V, E\} \in H$ **do**
    Compact $\{V \in c\}$ into single metanode m
  if $h\_size == |H_{vertices}|$ **then** break;

Step 0: Start Algorithm

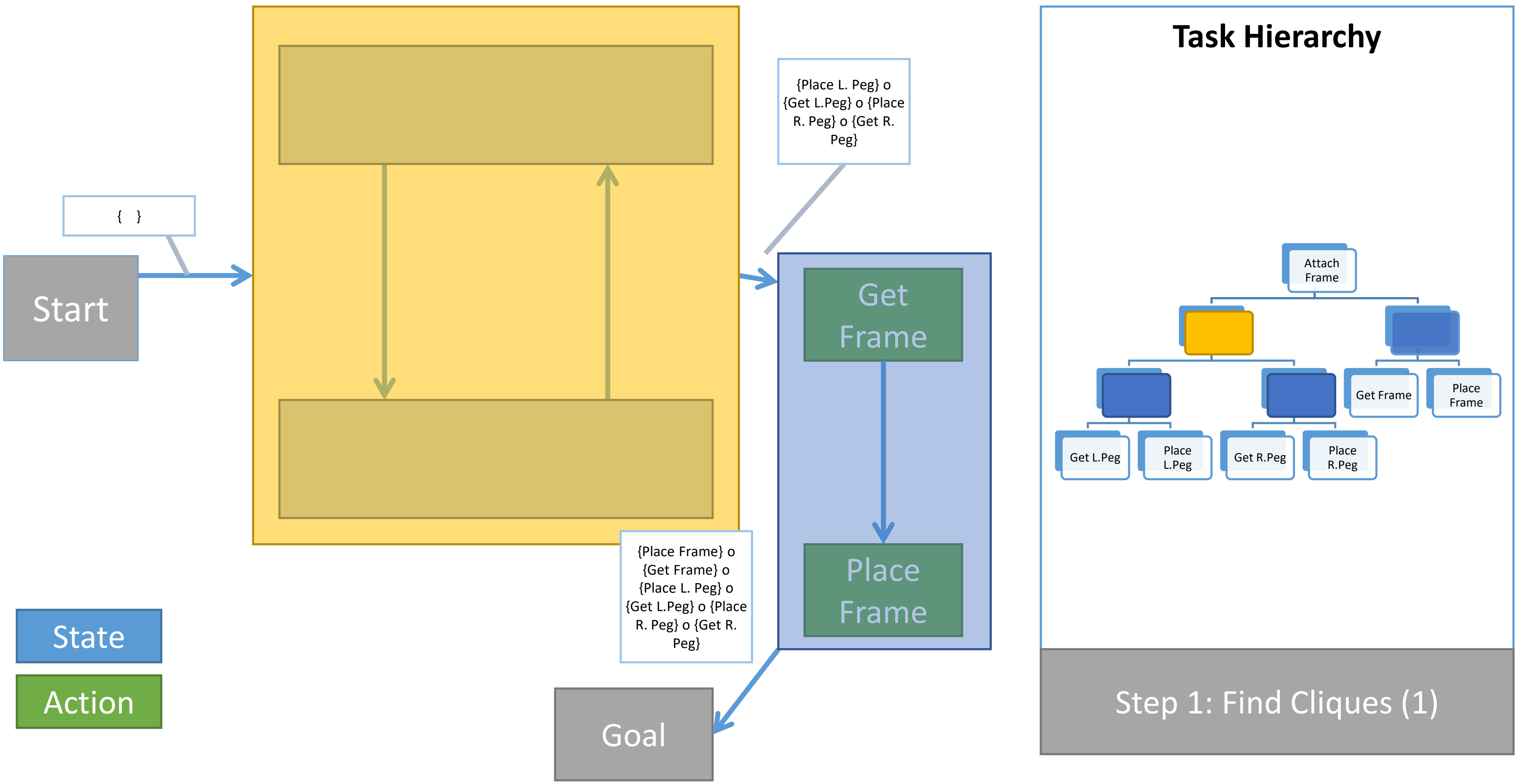
Building Hierarchical Structure



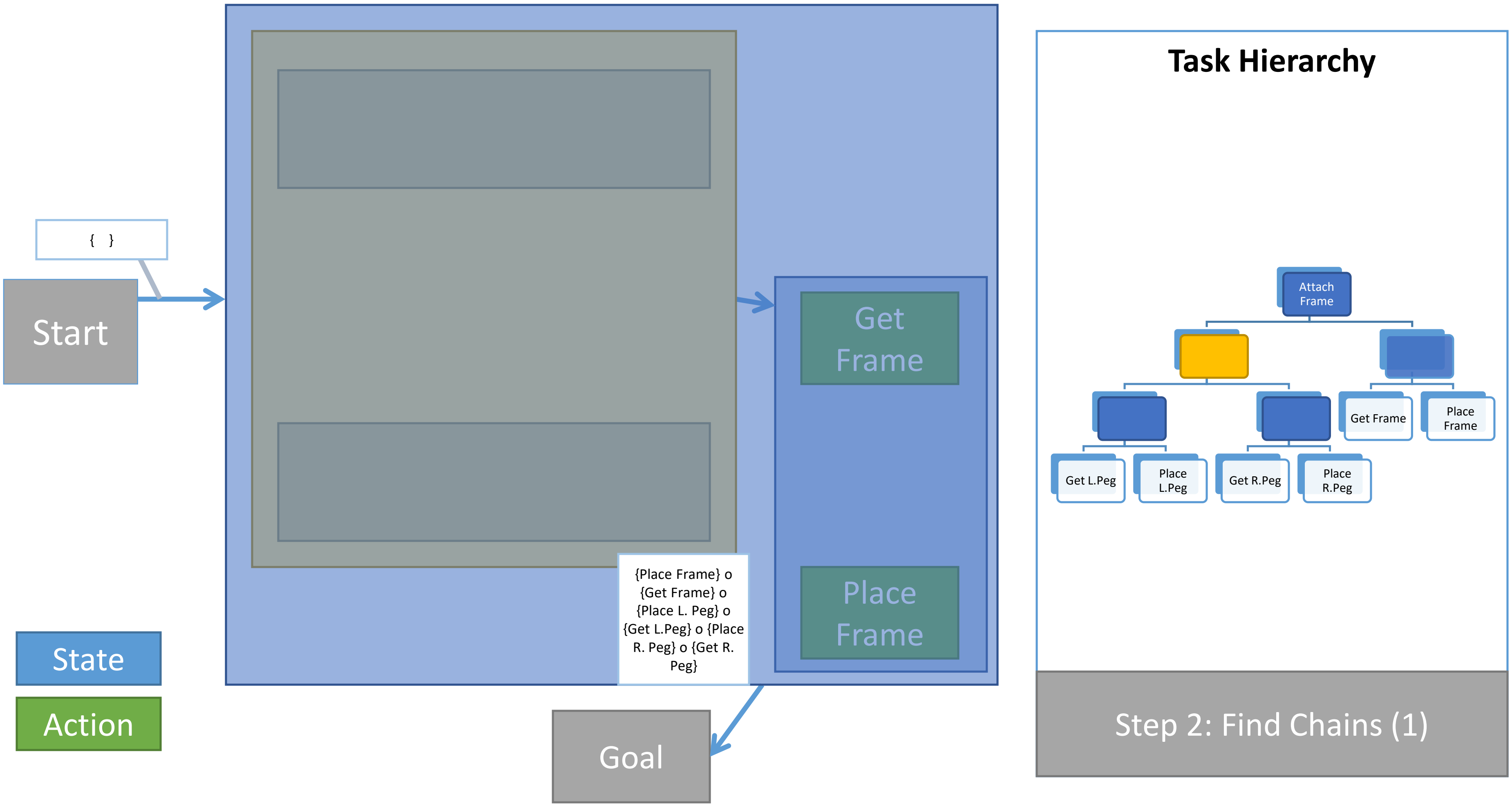
Building Hierarchical Structure



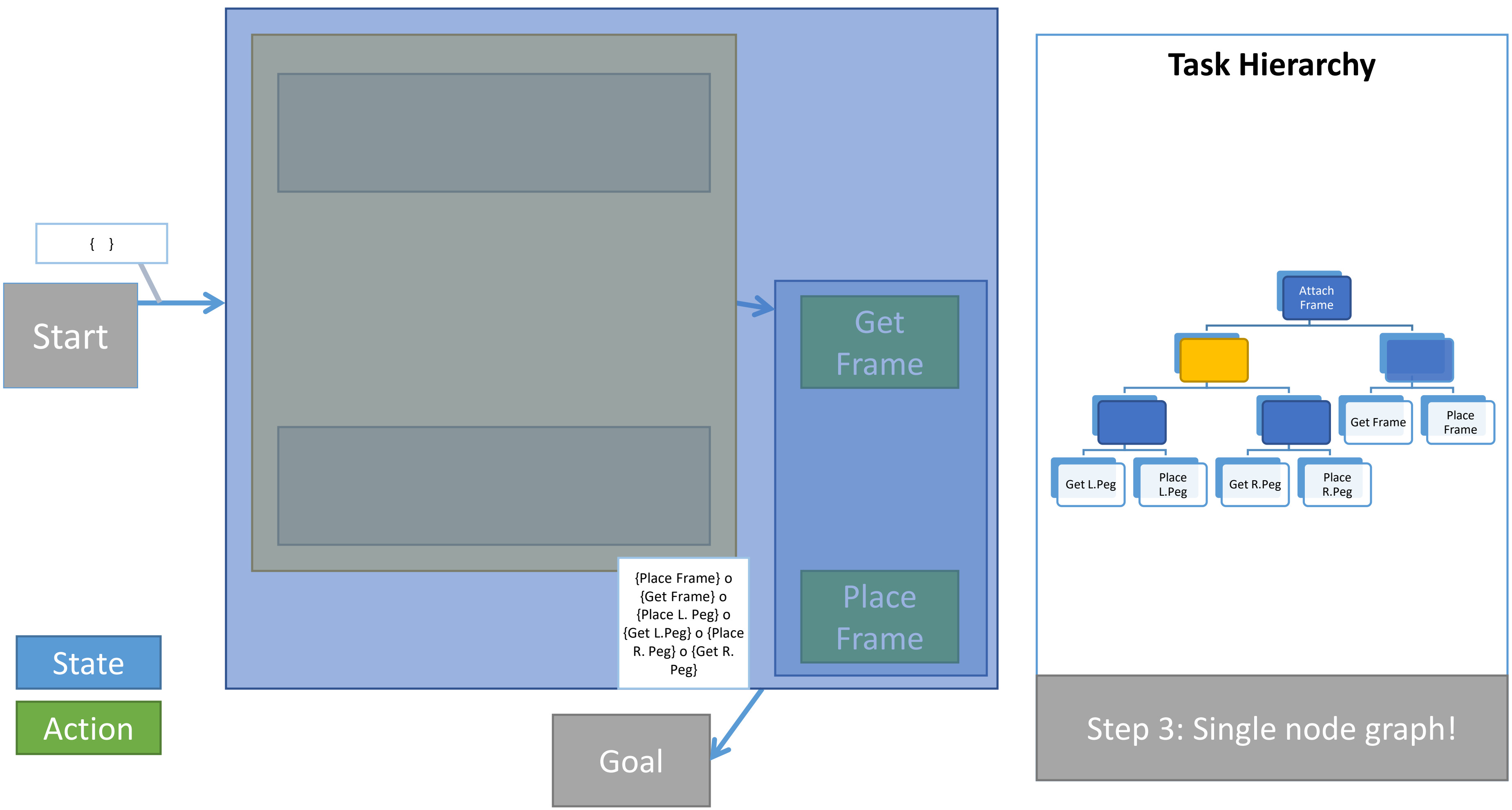
Building Hierarchical Structure



Building Hierarchical Structure



Building Hierarchical Structure



Context-sensitive Supportive Behavior Policies

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

[ICRA 17]

Bradley Hayes and Julie Shah

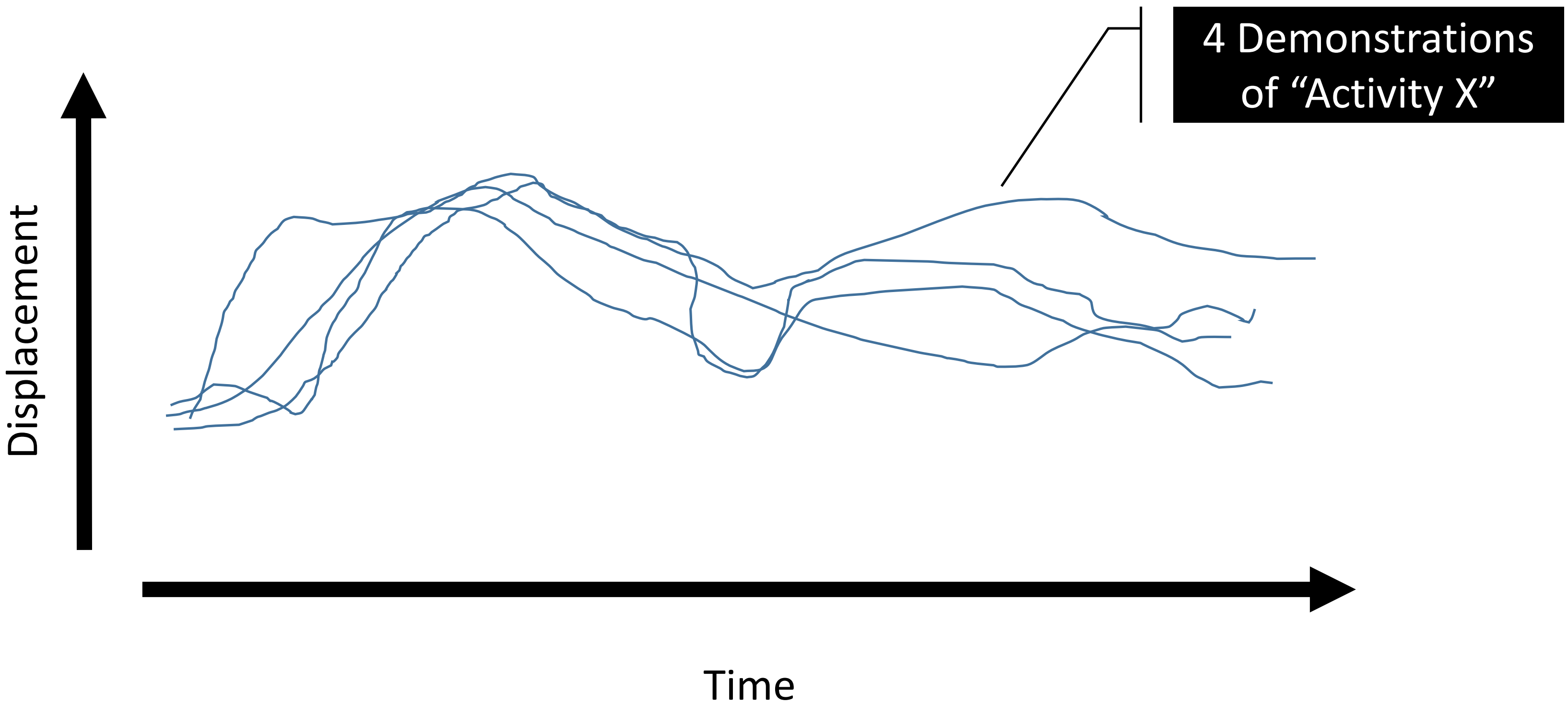
Collaborative robots need to recognize human activities

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

Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

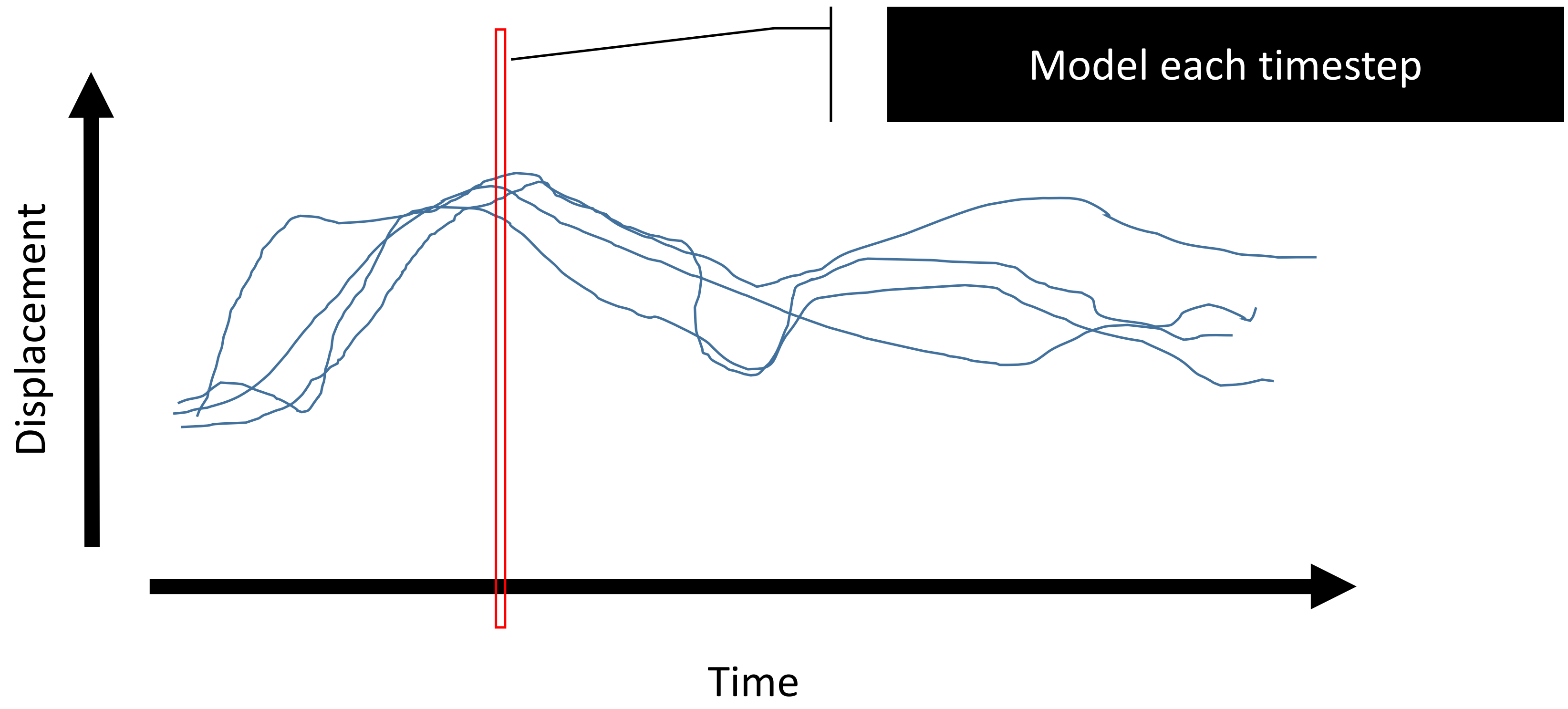
(Perez D'Arpino ICRA15)



Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

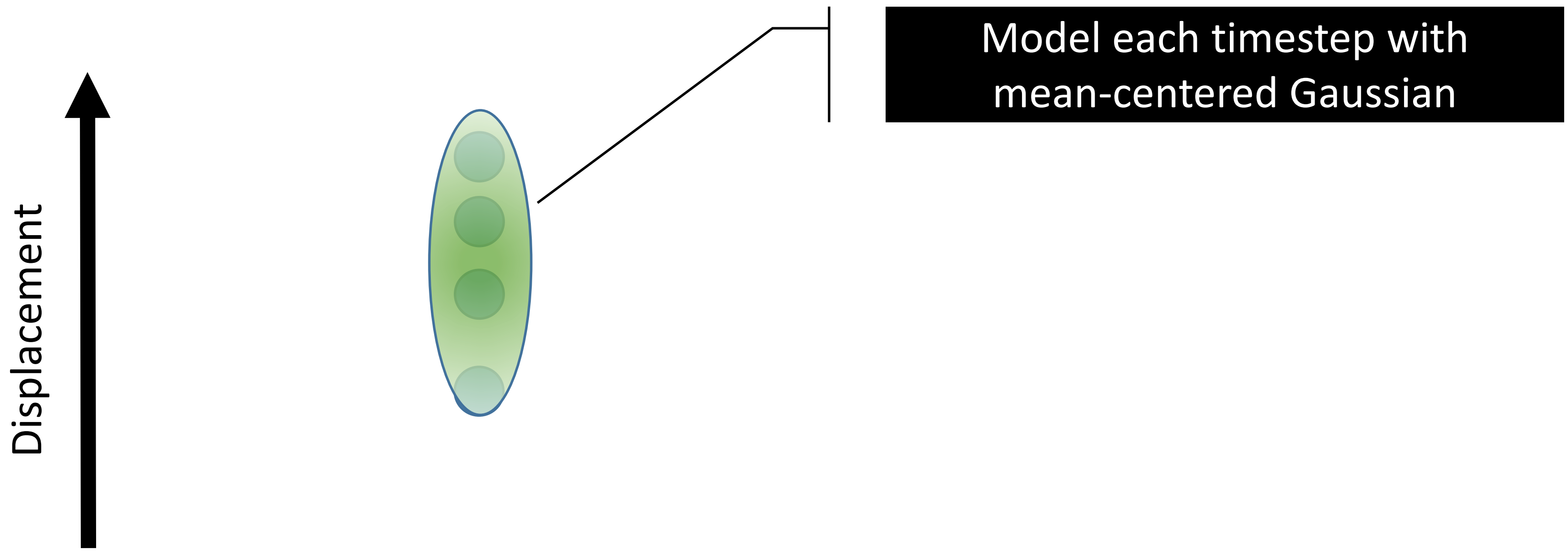
(Perez D'Arpino ICRA15)



Related Work

Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

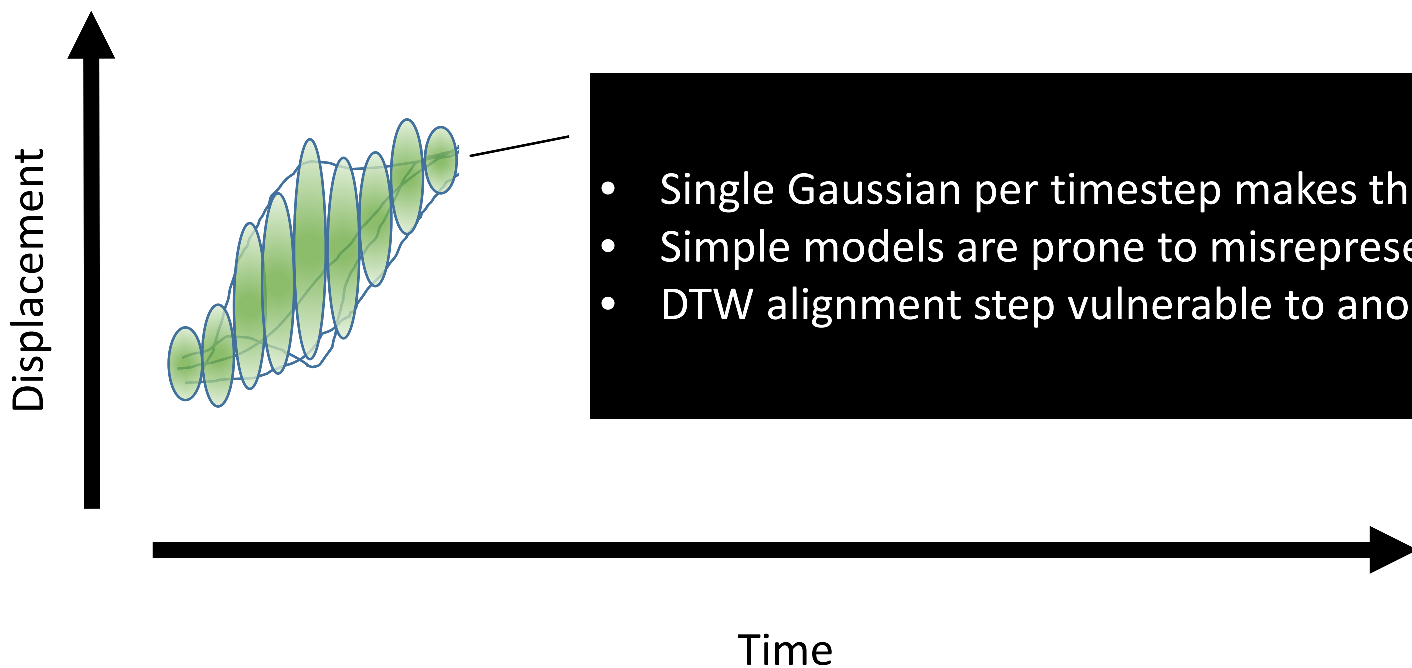
(Perez D'Arpino ICRA15)



Related Work

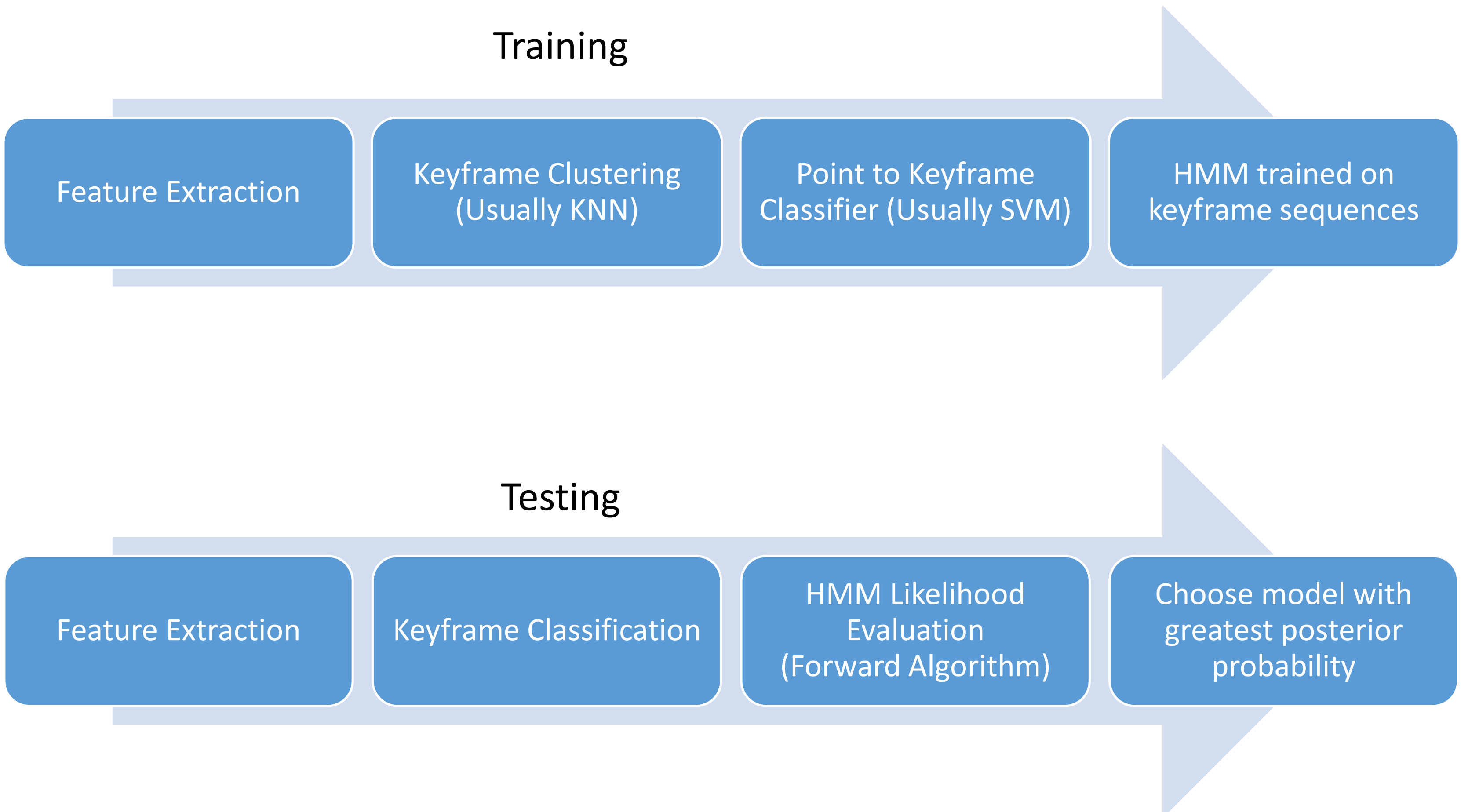
Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification

(Perez D'Arpino ICRA15)



- Single Gaussian per timestep makes this fast
- Simple models are prone to misrepresenting data
- DTW alignment step vulnerable to anomalies

Common Activity Classifier Pipeline



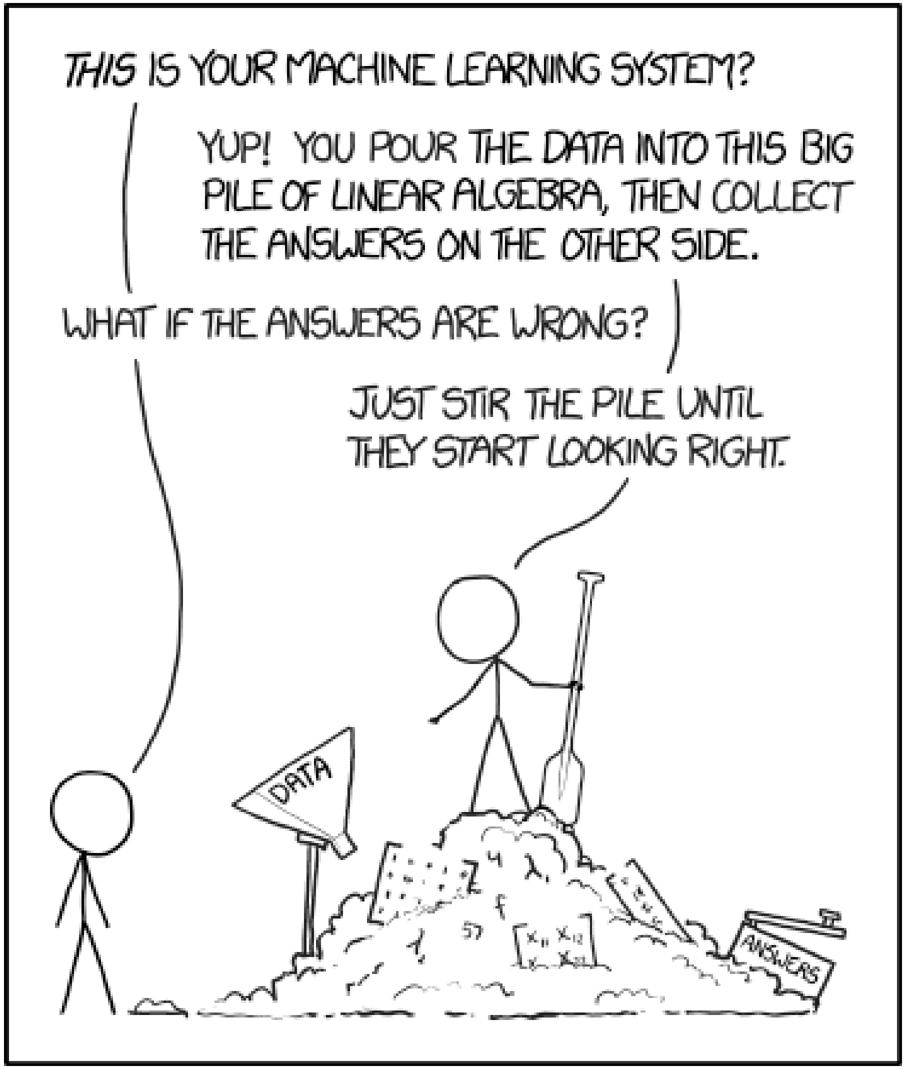
- P. Koniusz, A. Cherian, and F. Porikli, “Tensor representations via kernel linearization for action recognition from 3d skeletons.”
- Gori, J. Aggarwal, L. Matthies, and M. Ryoo, “Multitype activity recognition in robot-centric scenarios,”
- E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante, “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.”

New Activity Recognition Approaches?

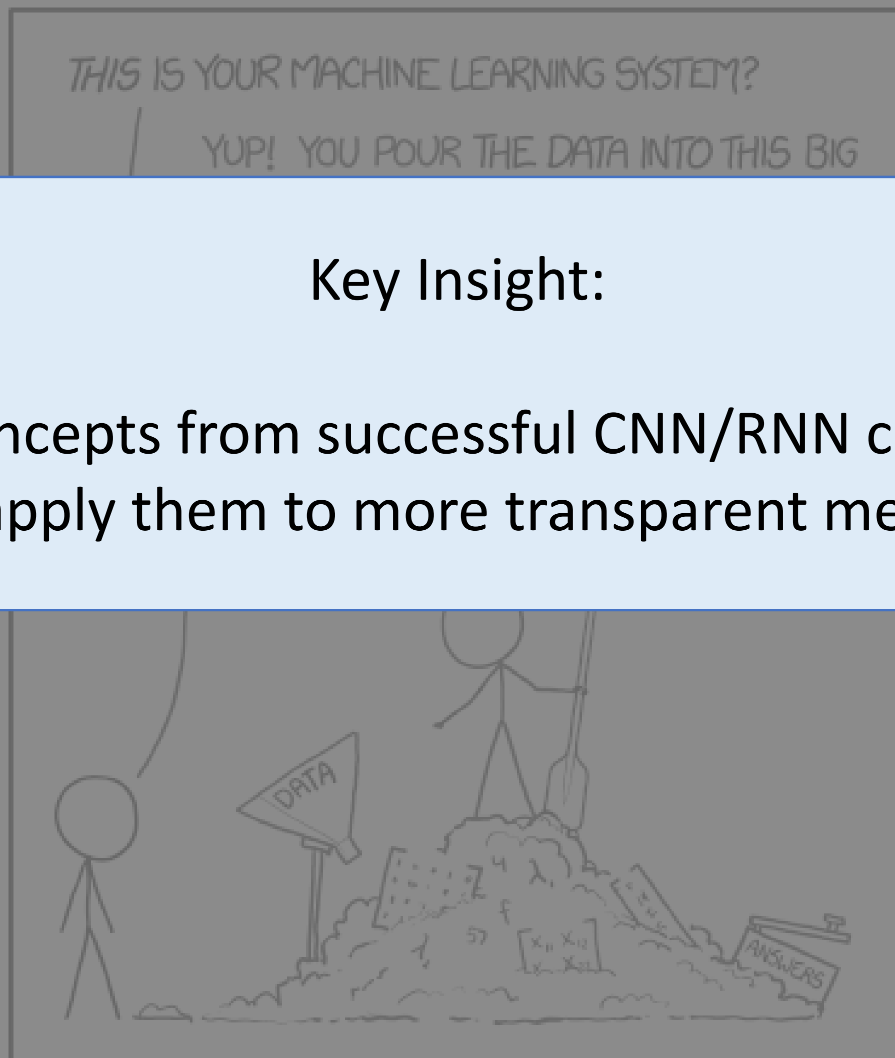
End-to-end Network

End-to-end Network

In real deployments, humans need to be able to understand robot decisions



In real deployments, humans need to be able to understand robot decisions

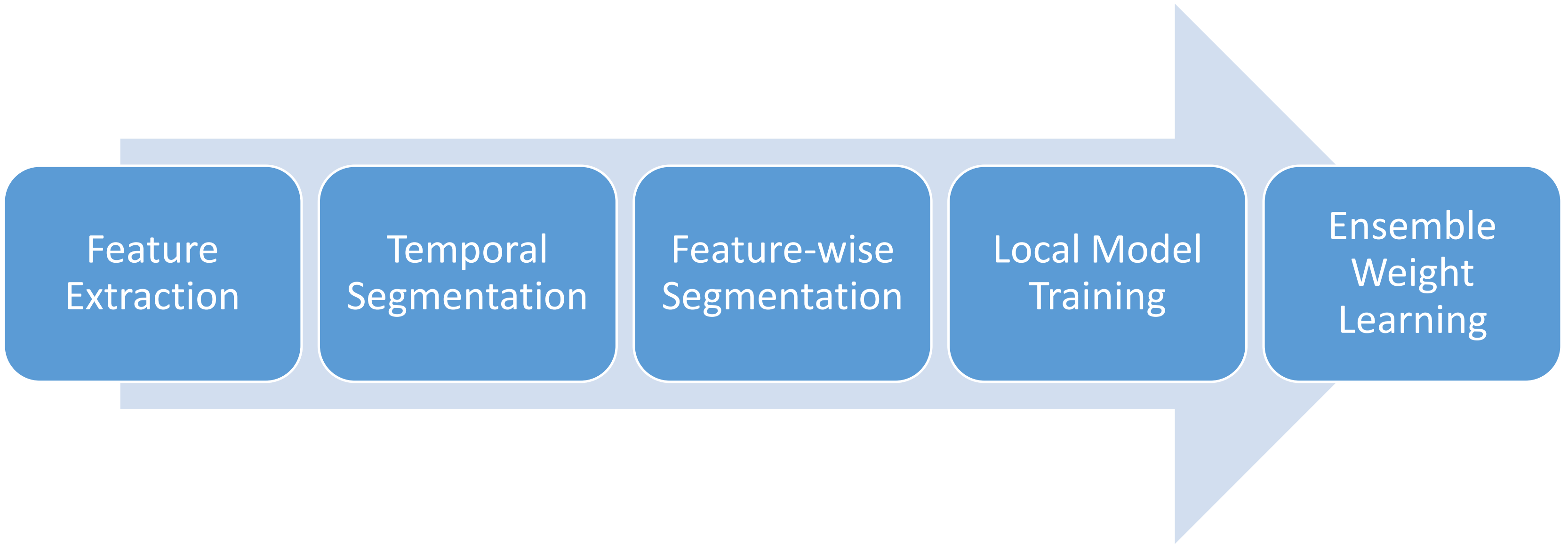


Key Insight:

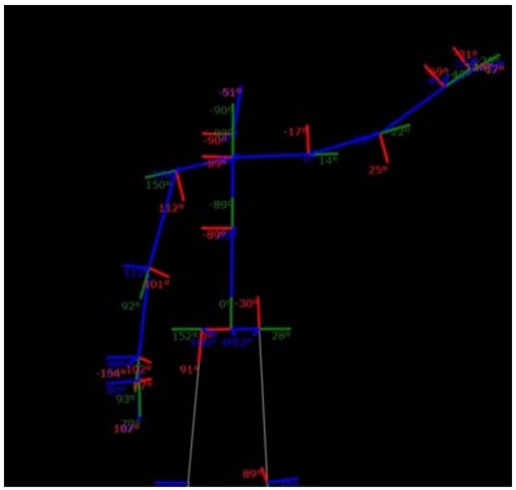
Take concepts from successful CNN/RNN classifiers and apply them to more transparent methods

Rapid Activity Prediction Through Object-oriented Regression (RAPTOR)

A highly parallel ensemble classifier that is resilient to temporal variations

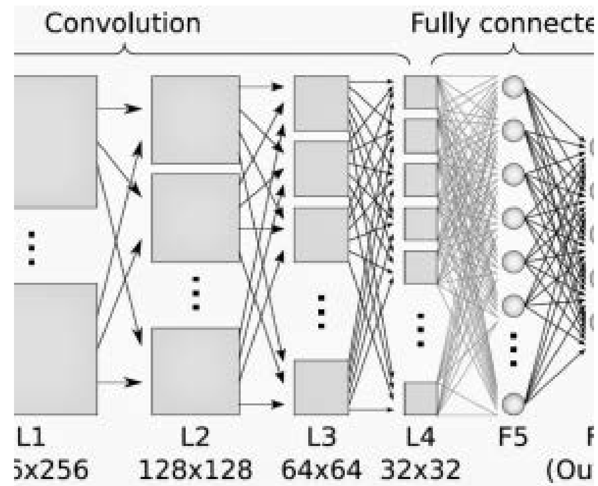


Activity Model Training Pipeline



Kinect Skeletal Joints

VICON Markers



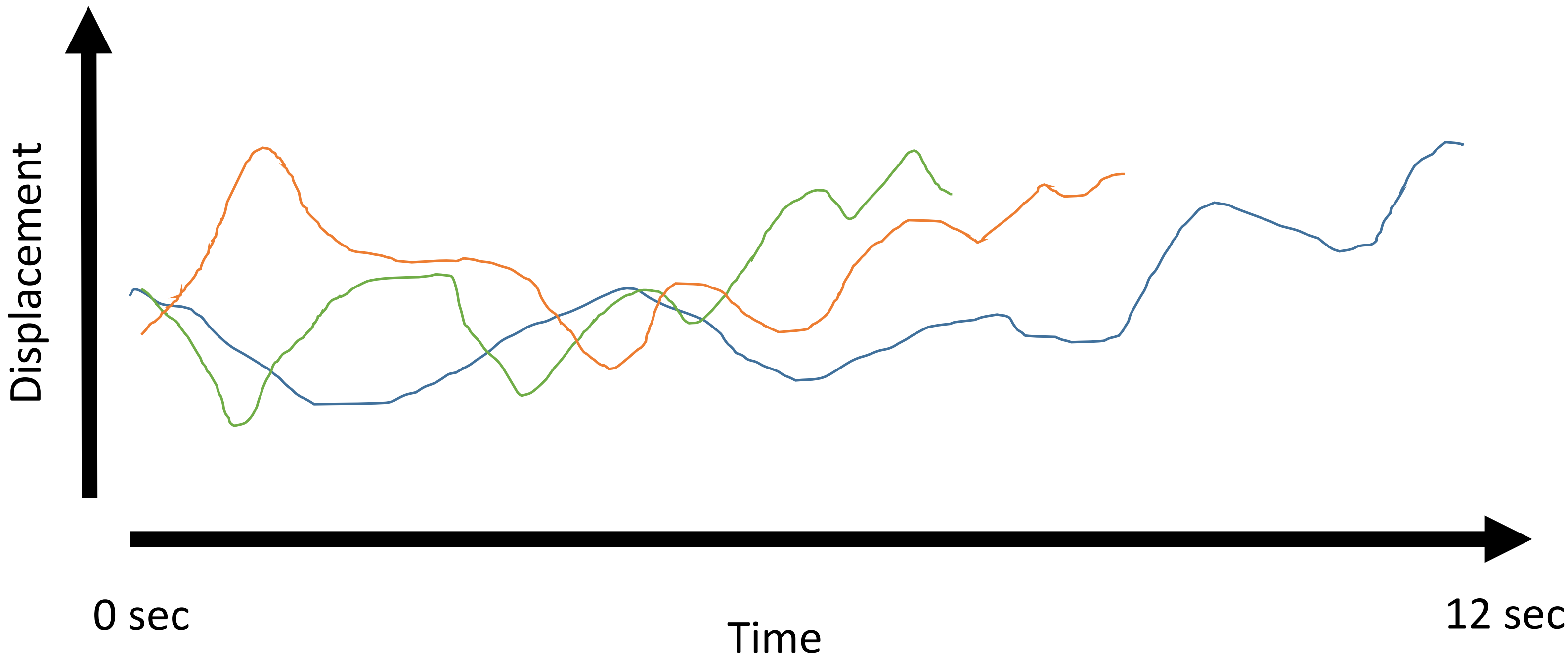
Learned Feature Extractor

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

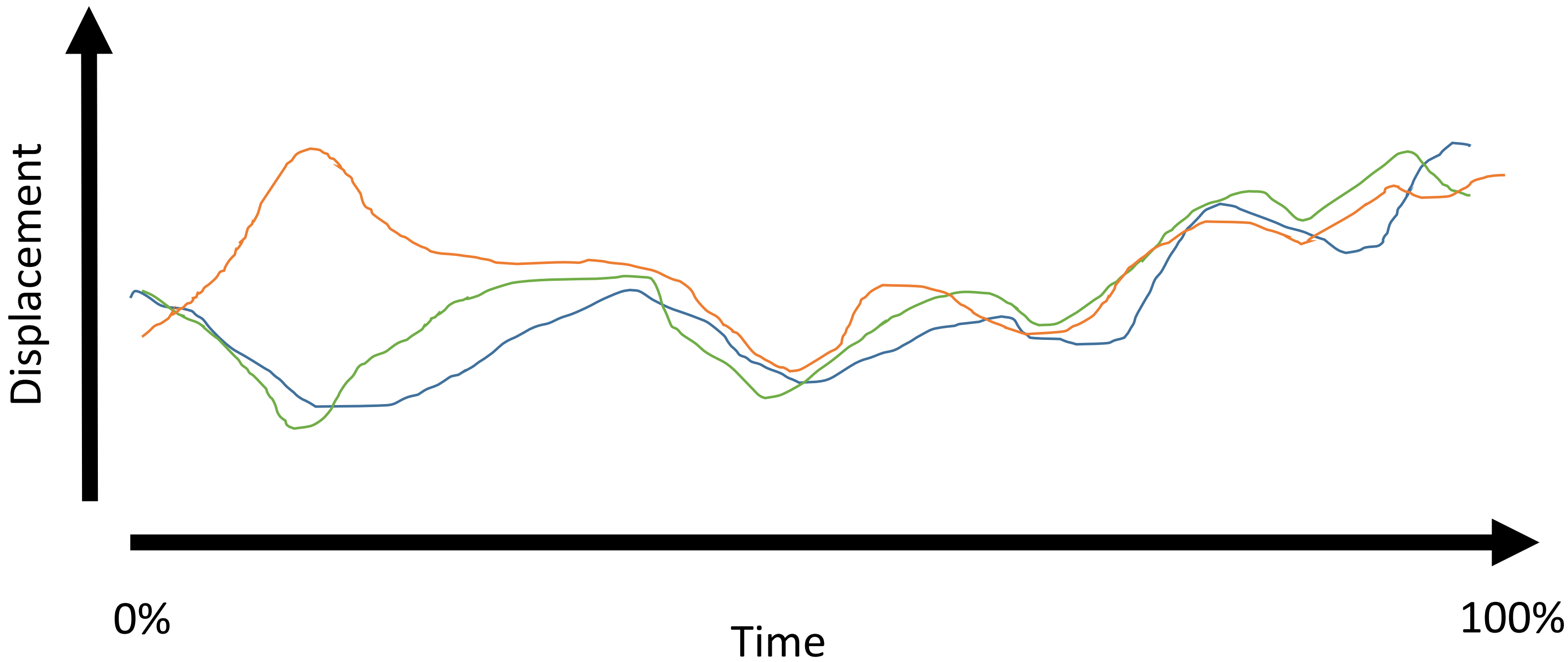
[Timestep x Feature] Matrix

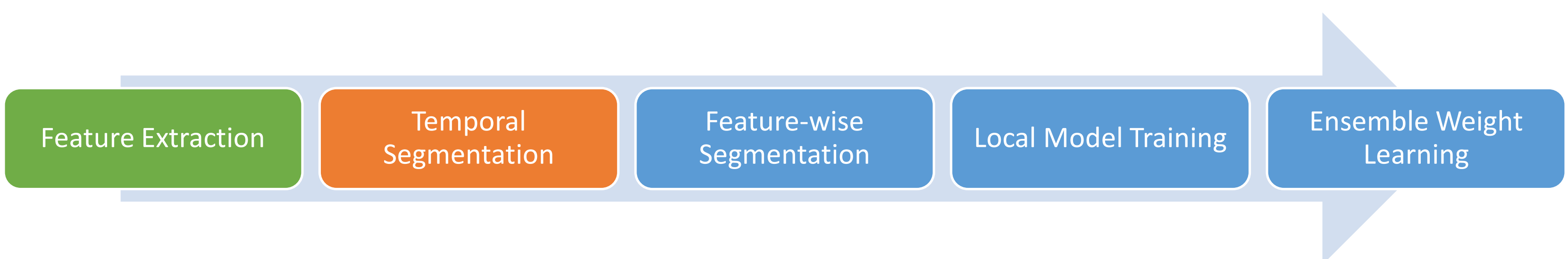
Feature Extraction → Temporal Segmentation → Feature-wise Segmentation → Local Model Training → Ensemble Weight Learning

Activity Model Training Pipeline

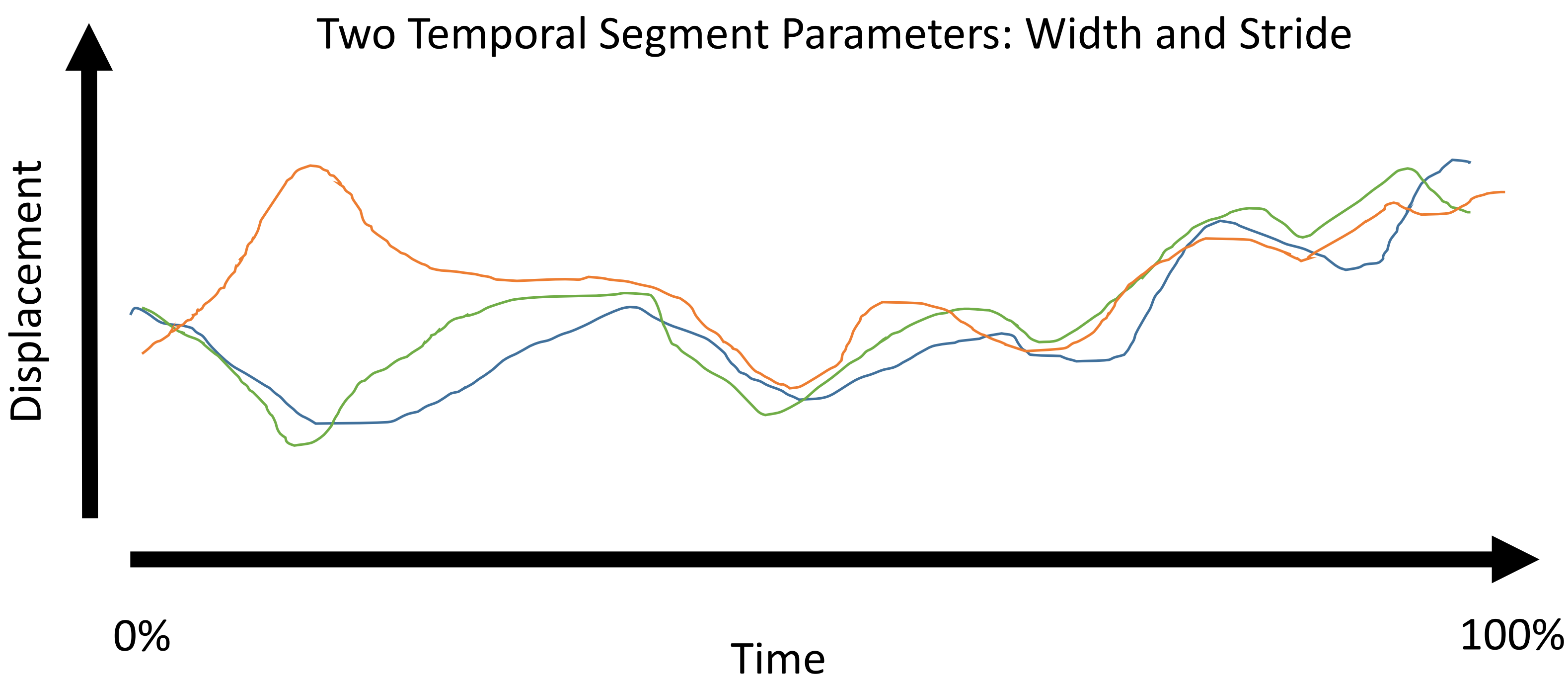


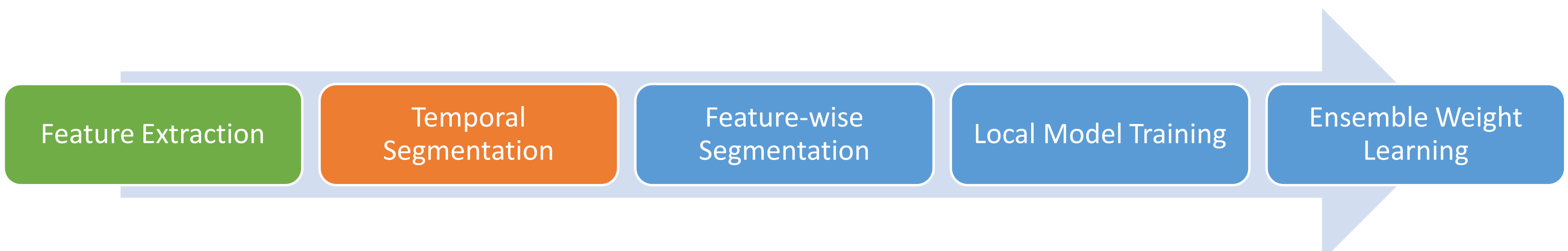
Activity Model Training Pipeline



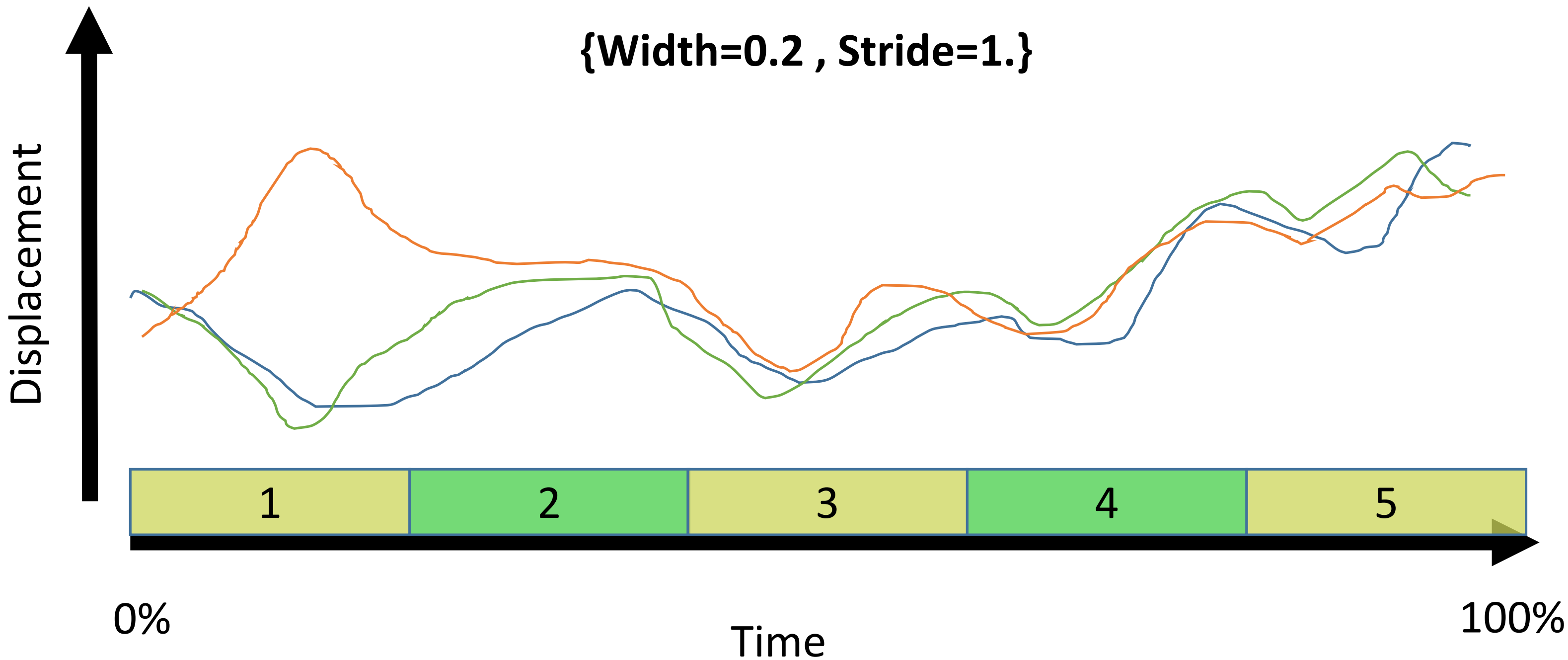


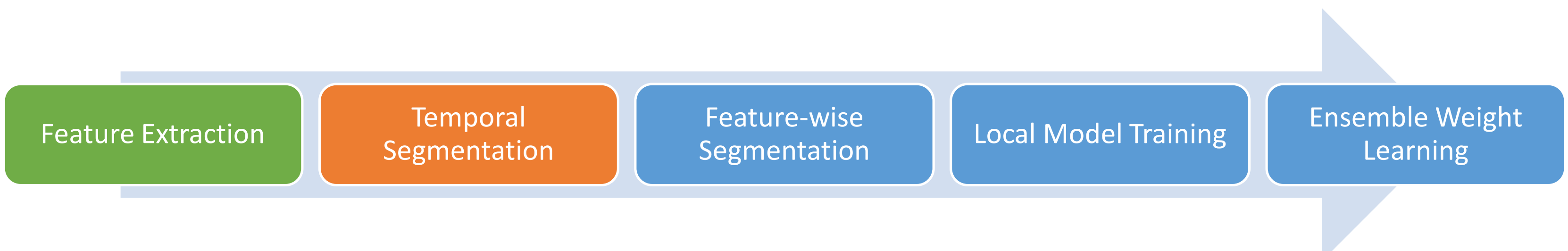
Activity Model Training Pipeline



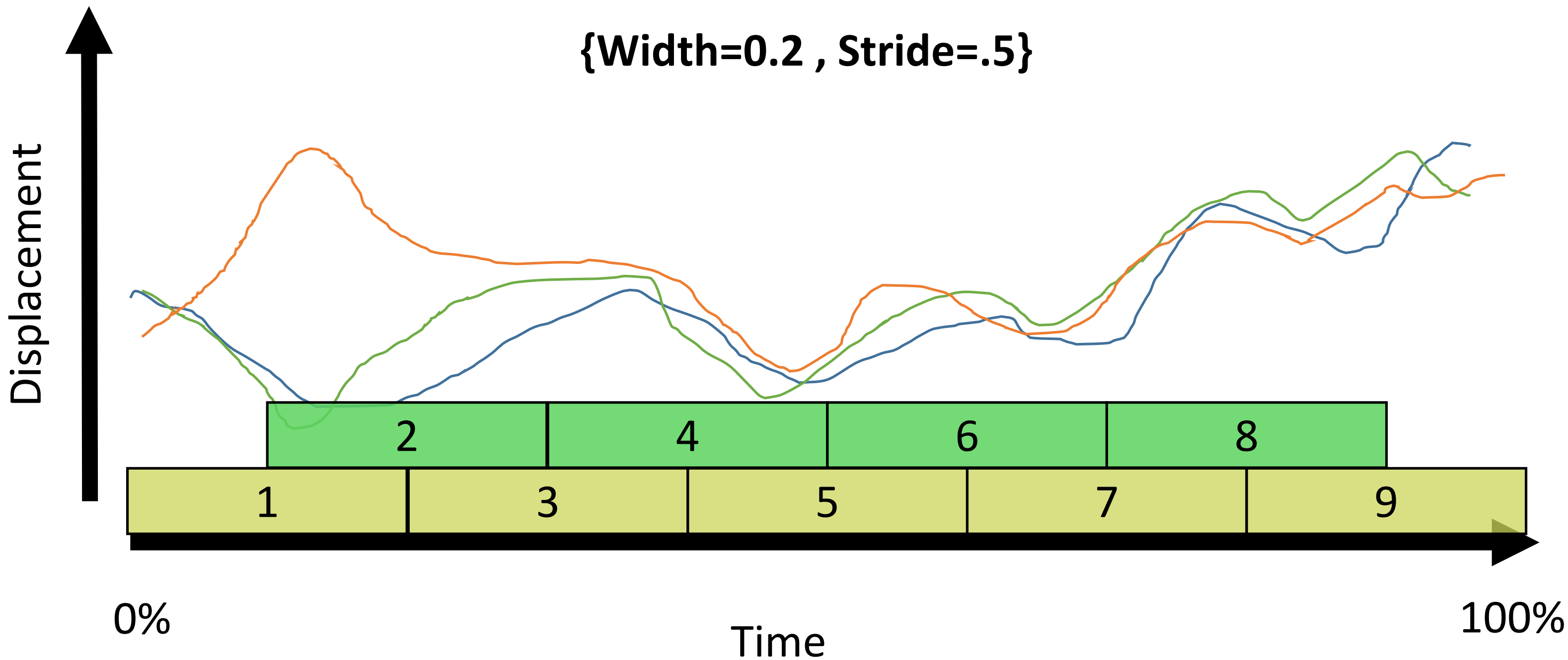


Activity Model Training Pipeline

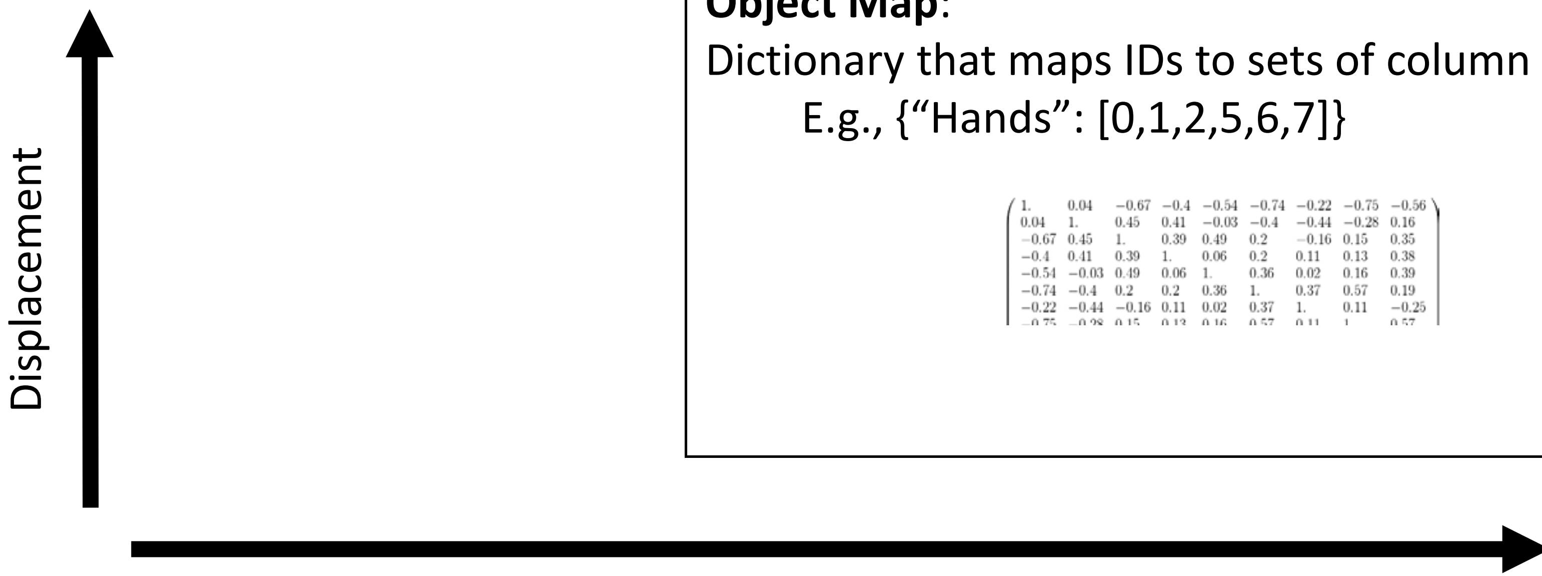




Activity Model Training Pipeline



Activity Model Training Pipeline

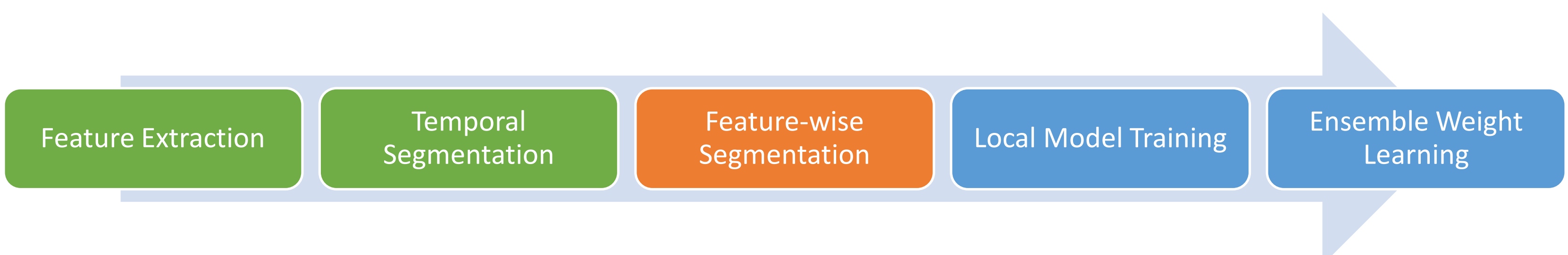


Object Map:

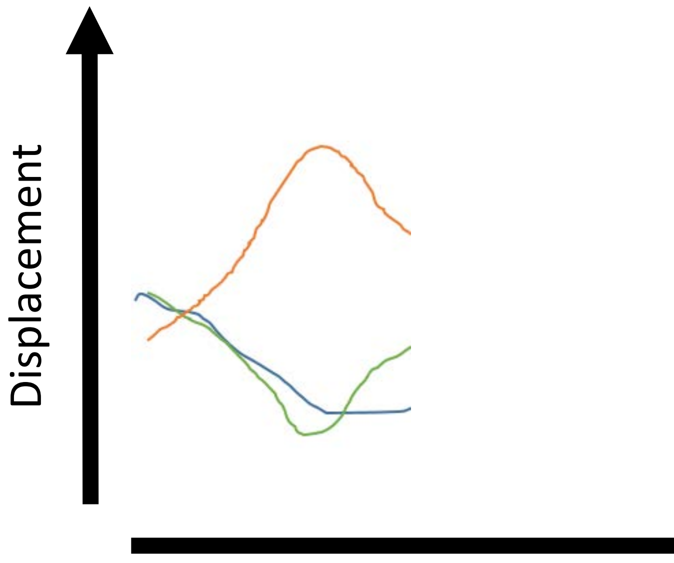
Dictionary that maps IDs to sets of column indices

E.g., {“Hands”: [0,1,2,5,6,7]}

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

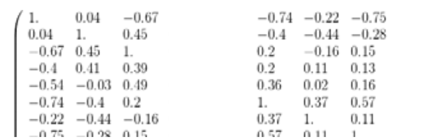


Activity Model Training Pipeline

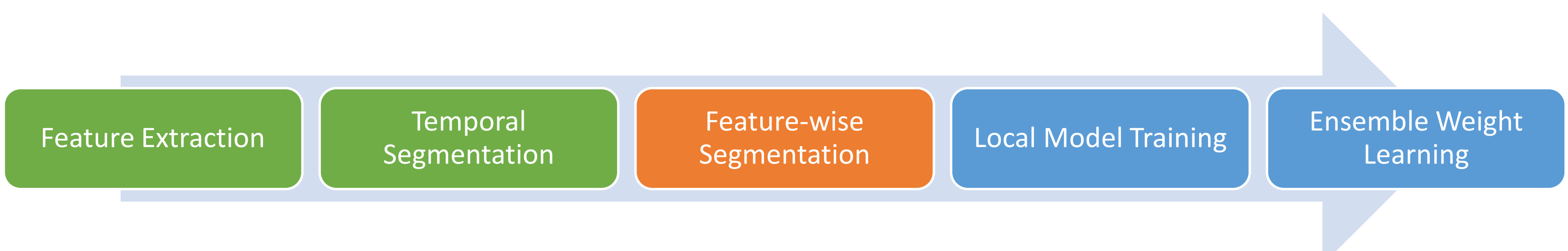


Within each temporal segment:

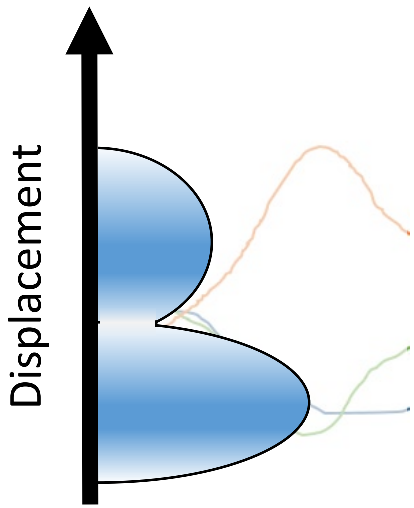
- Isolate columns of each demonstration trajectory according to (pre-defined) object map



- Create local model for each object



Activity Model Training Pipeline



Within each temporal-object segment:

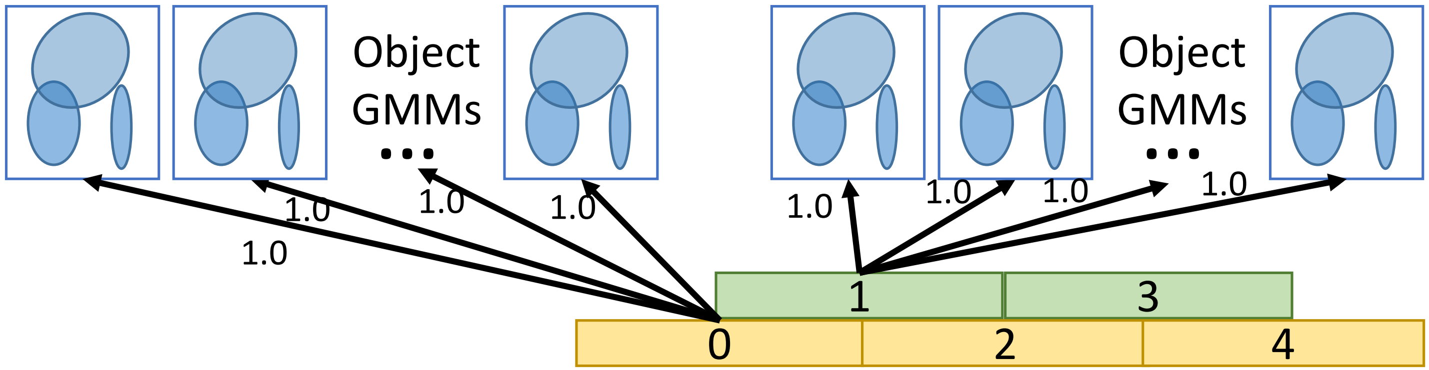
- Ignore temporal information for each data point
- Treat as general pattern recognition problem
- **Model the resulting distribution using a GMM**

Result: An activity classifier ensemble across objects and time!

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

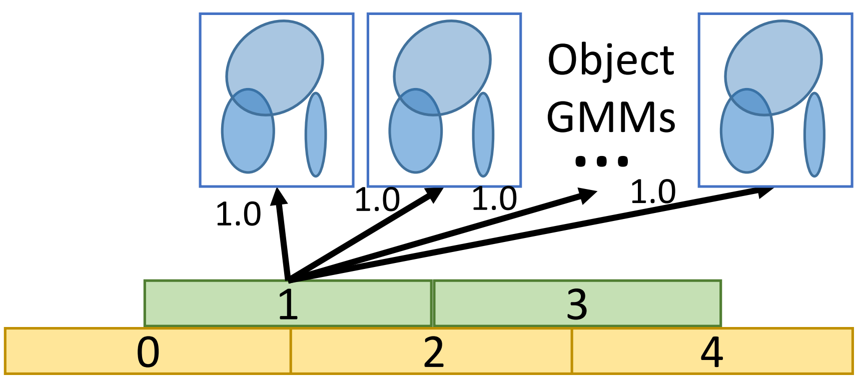
Activity Model Training Pipeline

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

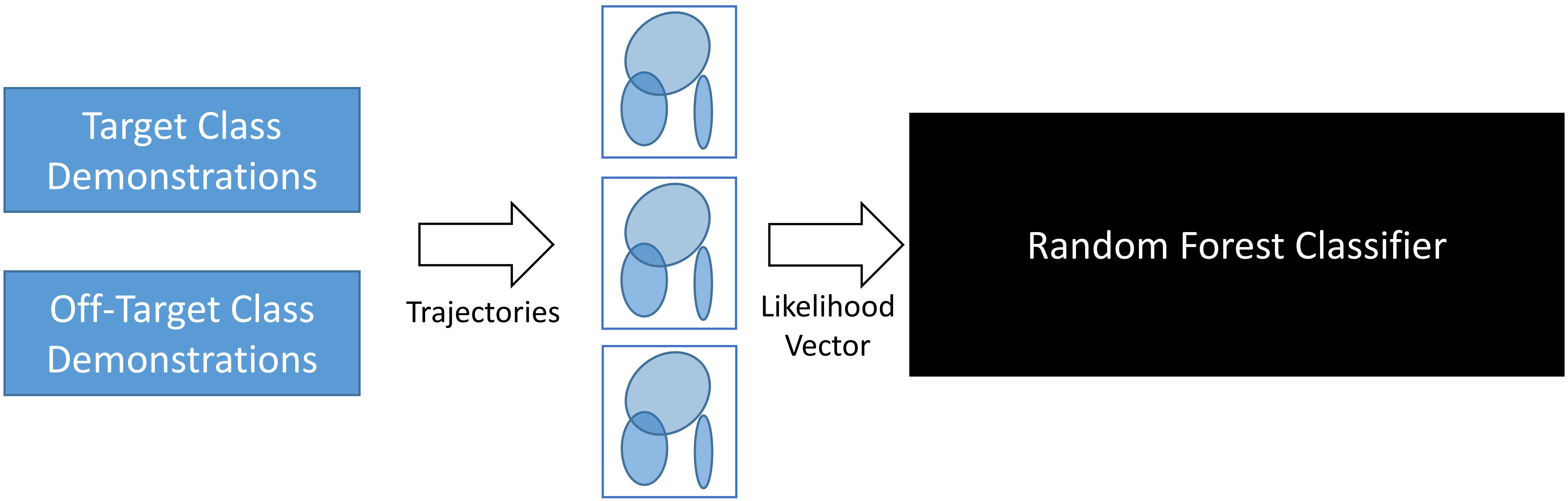


Random Forest Classifier

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

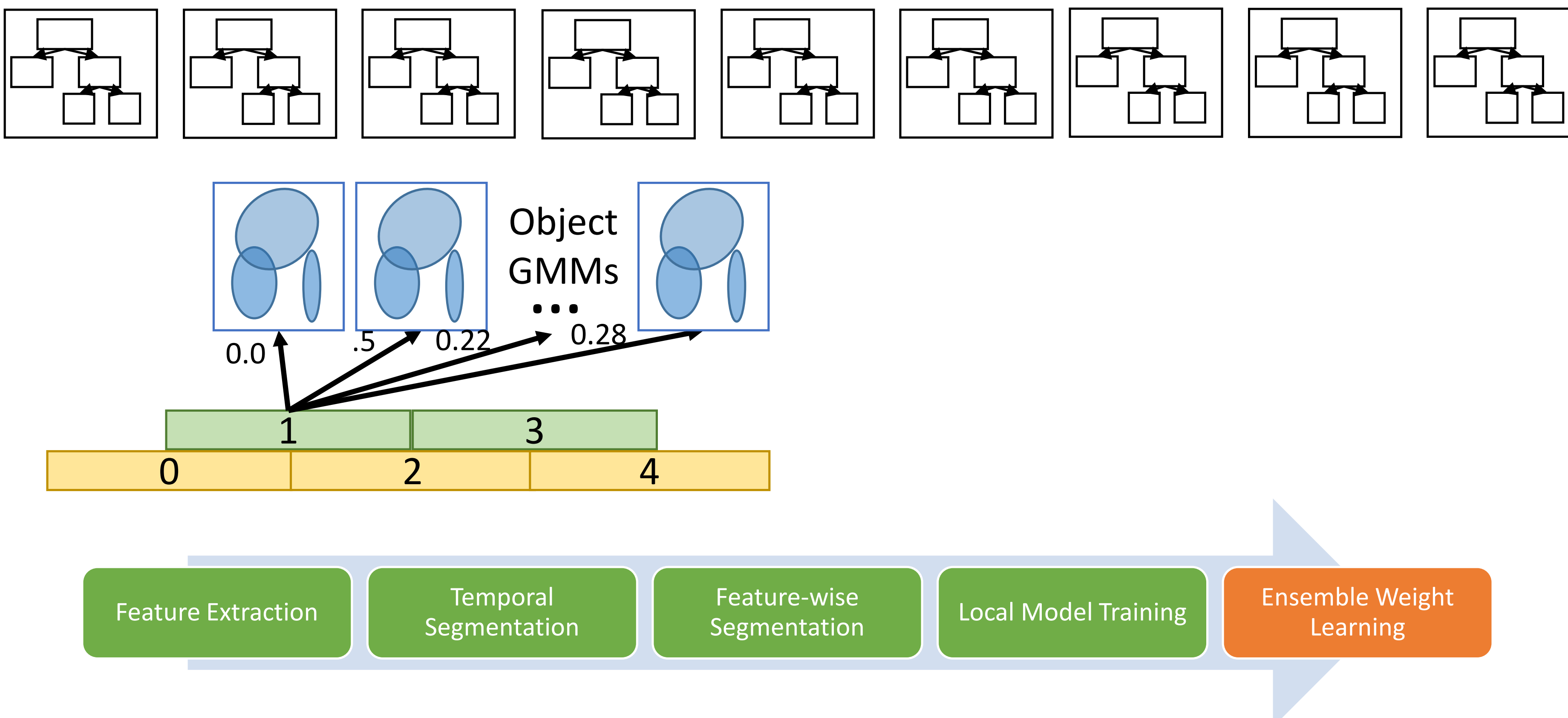
Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment



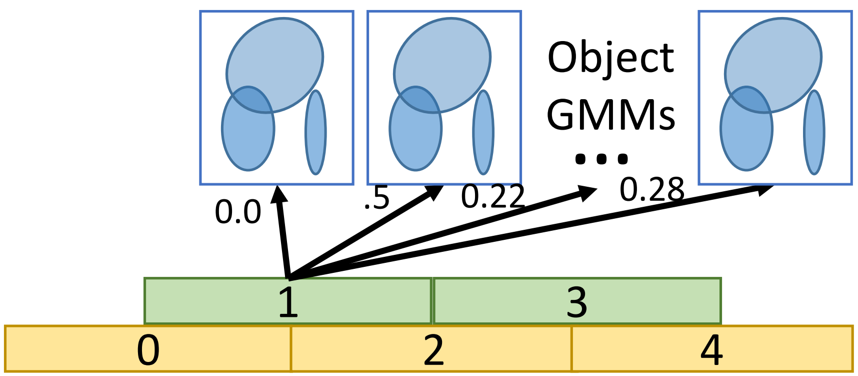
Activity Model Training Pipeline

- 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

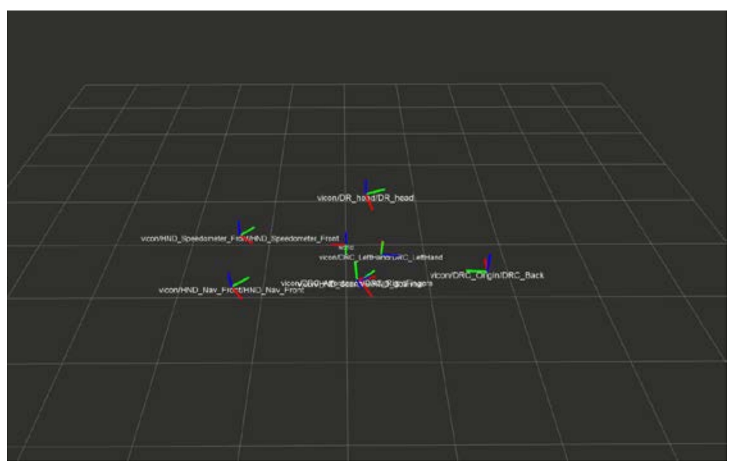


Result: Trained Highly Parallel Ensemble Learner with Temporal/Object-specific sensitivity

Results: Three Datasets

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

UTKinect

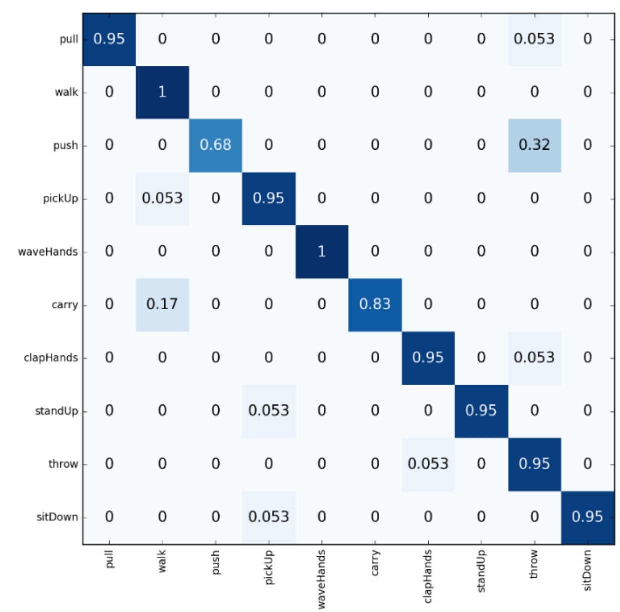


Automotive Final Assembly

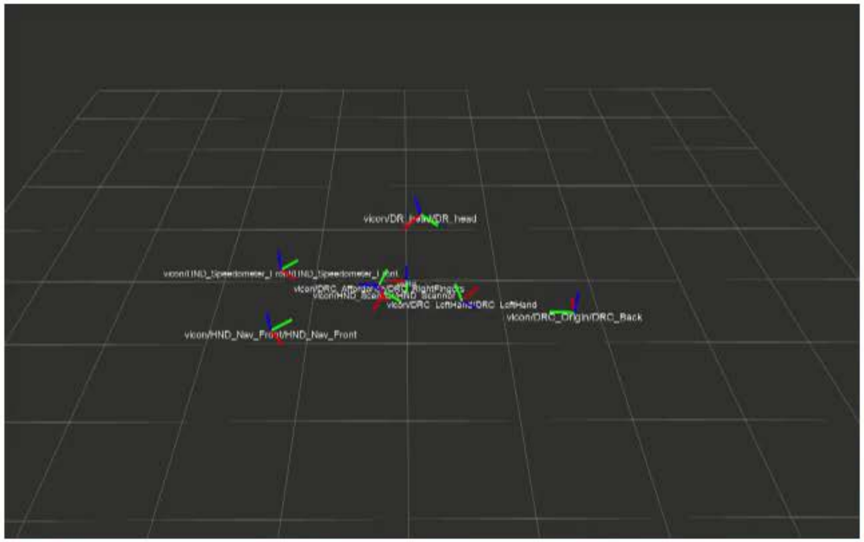
Sealant Application

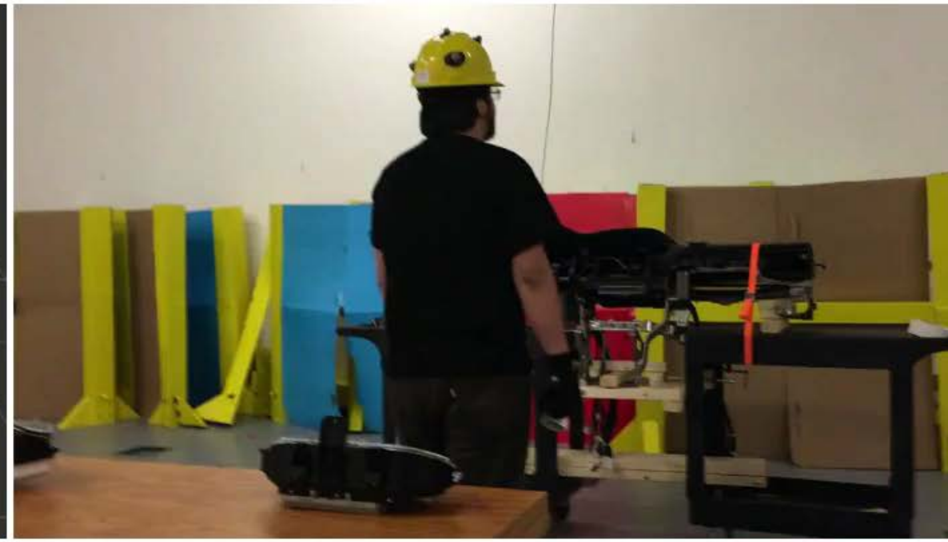
Recognition Results: UTKinect-Action3D

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:

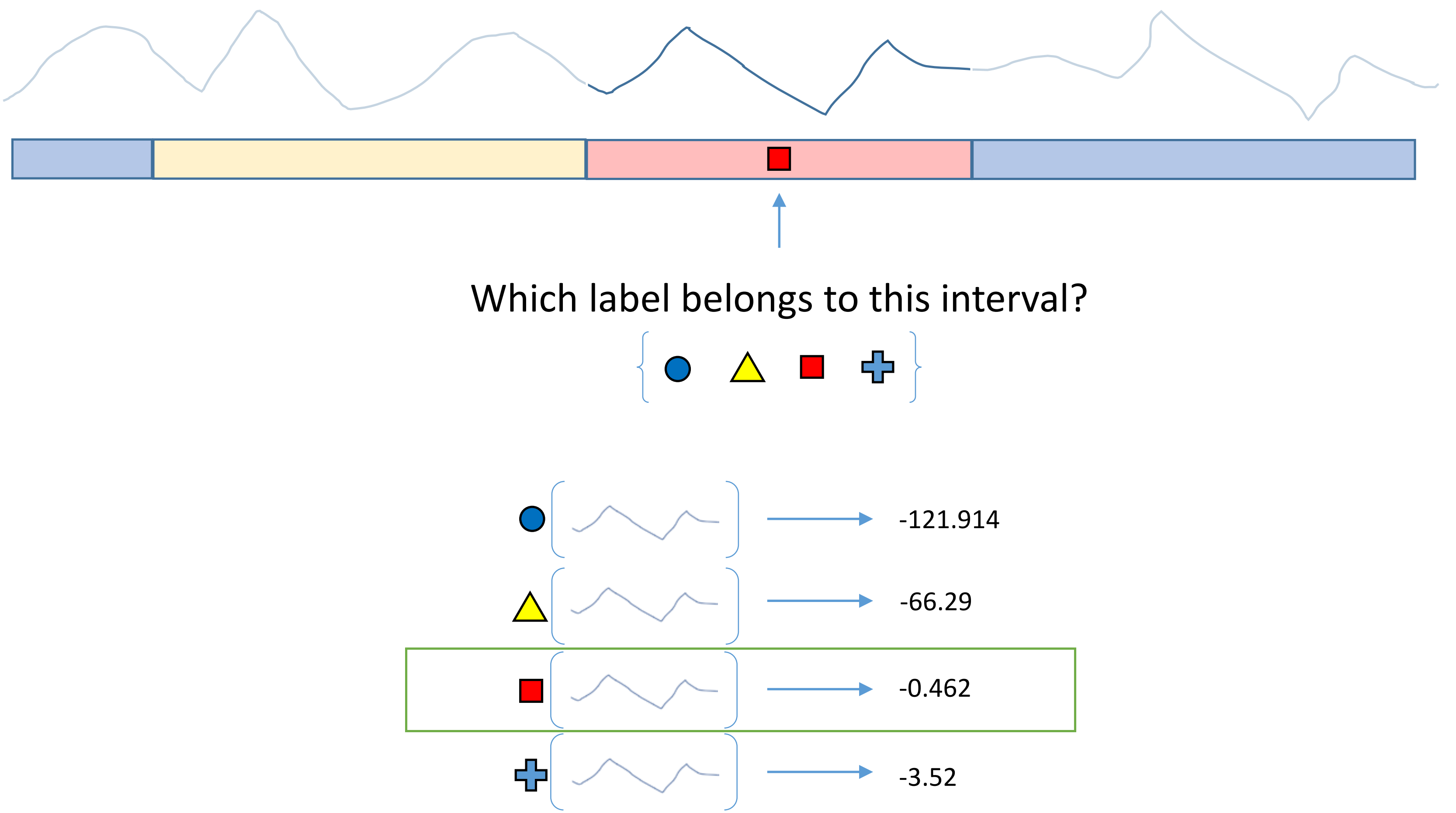
- Apply outlier detection methods across internal activity classifiers
- Use outliers or lack thereof to explain issues across **time** and **objects**

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

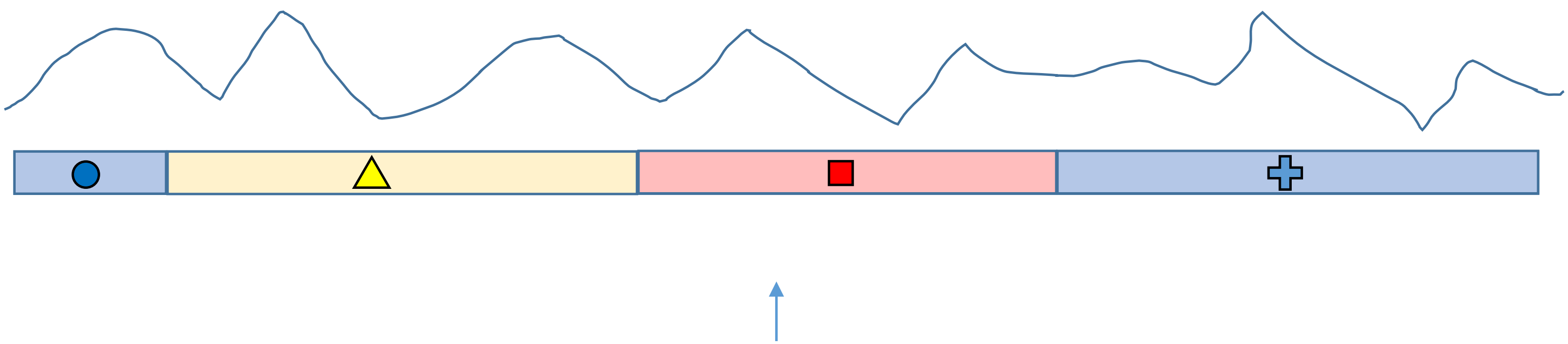
“In the **middle and end** of the trajectory, the **left hand and right hand** features were very poorly matched to my template.”

Real-time Activity Segmentation and Classification

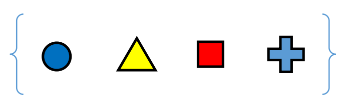
Classification vs. Segmentation



Classification vs. **Segmentation**



What are the right intervals?
Which intervals should get labels?
Which labels should be where?



A Naïve Changepoint Detection Approach

<u>Scenario</u>

Duration: 2700 frames – 1.5 minutes of data
Classifiers: 11 – Avg run-time of 0.2s each

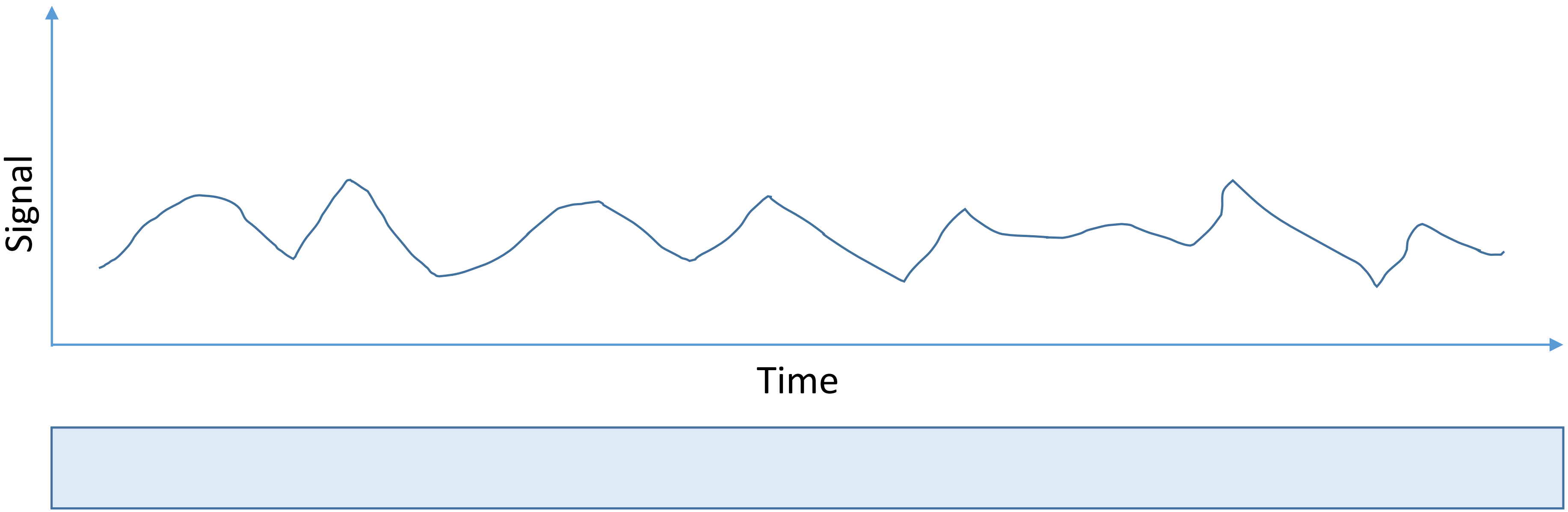
IDEA: Run every activity classifier over every possible segment

- Given n frames:
  - For every interval *q* in the range [0, n]:
    - Evaluate each classifier on *q*
  - Sort results by likelihood
  - Assign class labels to uncovered intervals from highest likelihood classifications until no unlabeled frames remain
- Return timeline (list of intervals)

$2700^2 * 0.2 = 1458000$sec
~16.88 days

Classifiers must be ideal (sensitive to trajectory length, non-overlapping, comparable tolerance to noise, etc.)

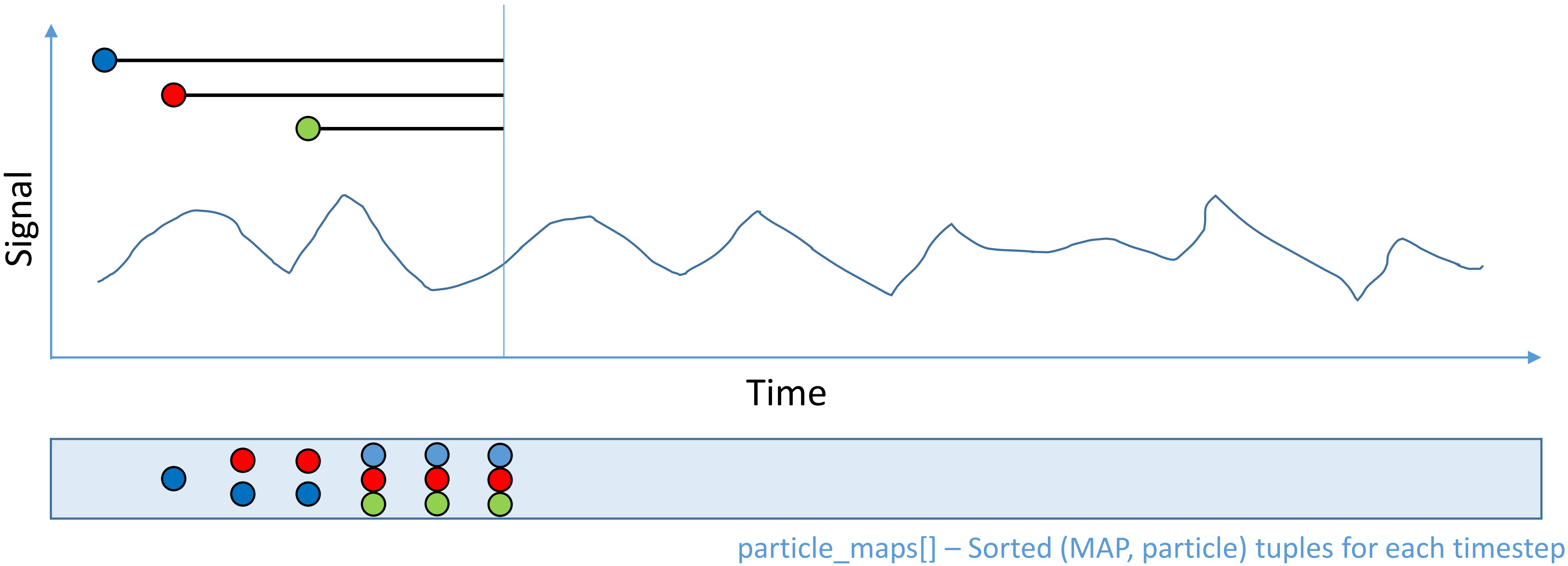
Particle Filtering for Changepoint Detection



particle_maps[] – Sorted (MAP, particle) tuples for each timestep

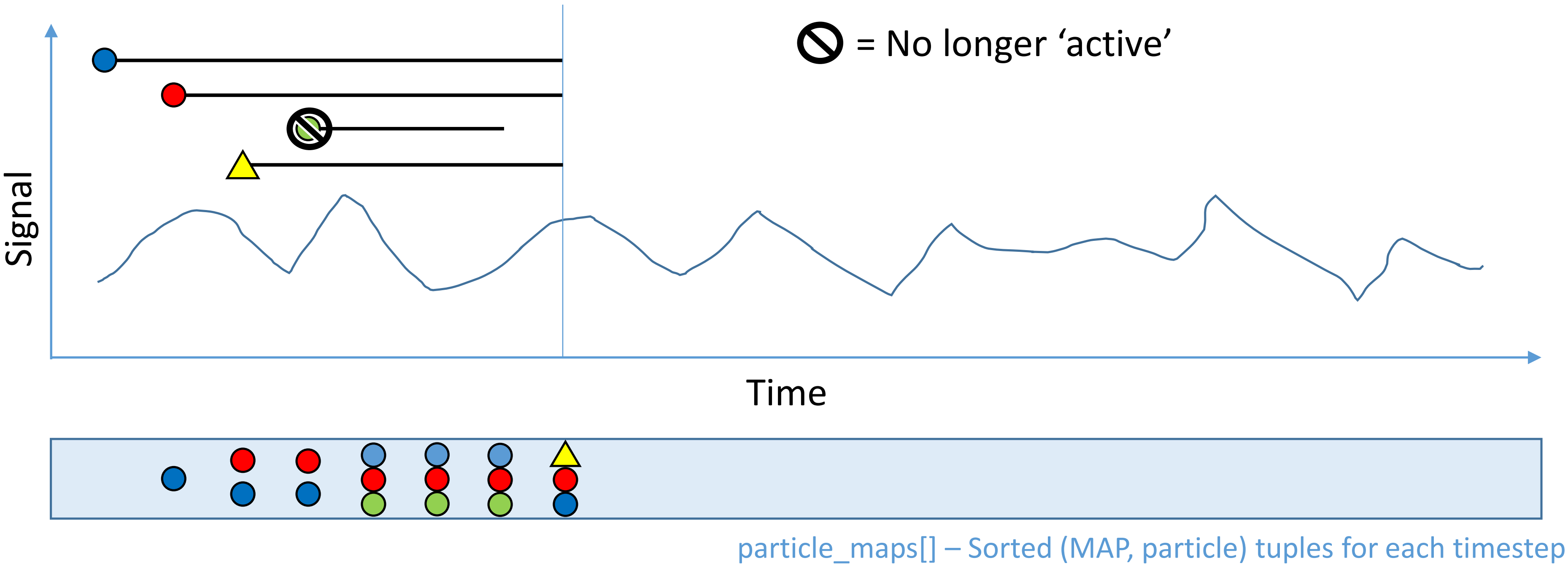
- At each time step *t*:
    - Create new particles for all eligible classes
        - *start_time* = *t* – *minimum_class_duration*
        - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
    - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
    - Terminate stale particles

Particle Filtering for Changepoint Detection



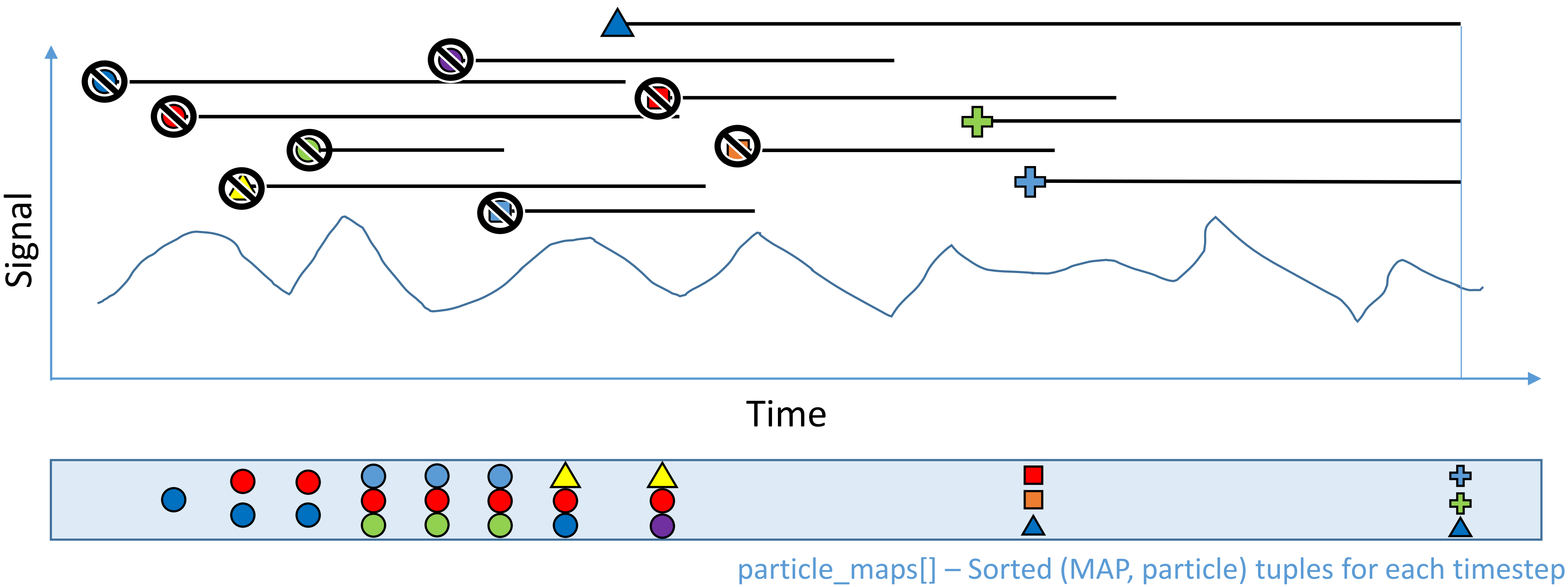
- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

Particle Filtering for Changepoint Detection



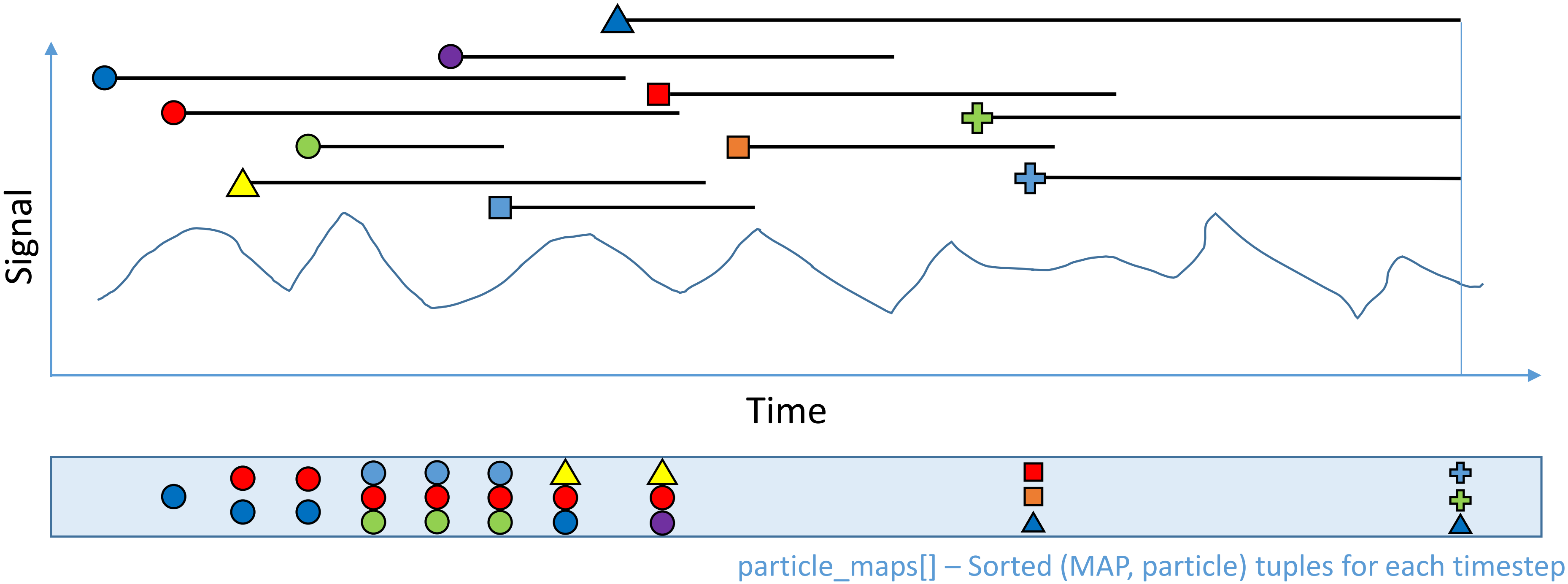
- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

Particle Filtering for Changepoint Detection



- At each time step *t*:
  - Create new particles for all eligible classes
    - *start_time* = *t* – *minimum_class_duration*
    - *prev_interval* = particle with highest MAP estimate in best[*start_time*]
  - Evaluate existing particles' likelihoods over the interval [*p.start_time*, *t*] and store as (likelihood, p) tuples in *particle_maps*
  - Terminate stale particles

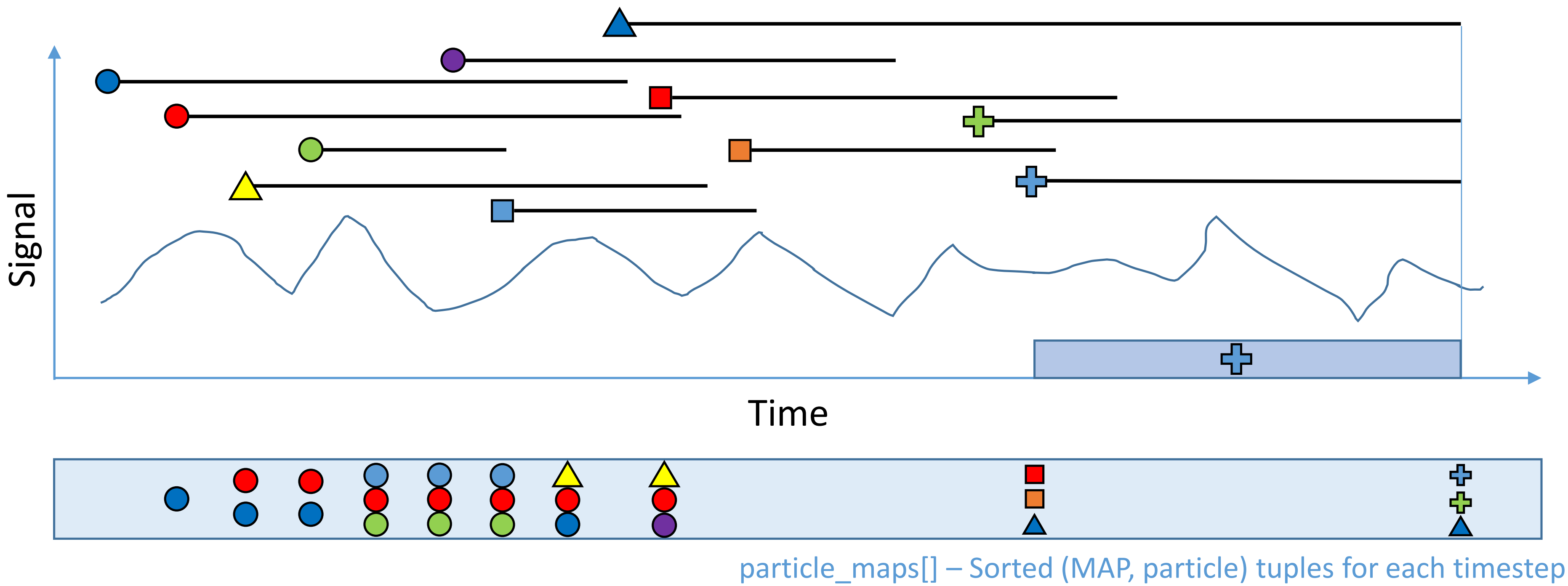
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
  - Take best (MAP, particle) at particle_maps index *f*
  - Annotate segment [*shape_start*, *f*] with *shape_class*
  - Set *f = particle.start_time*

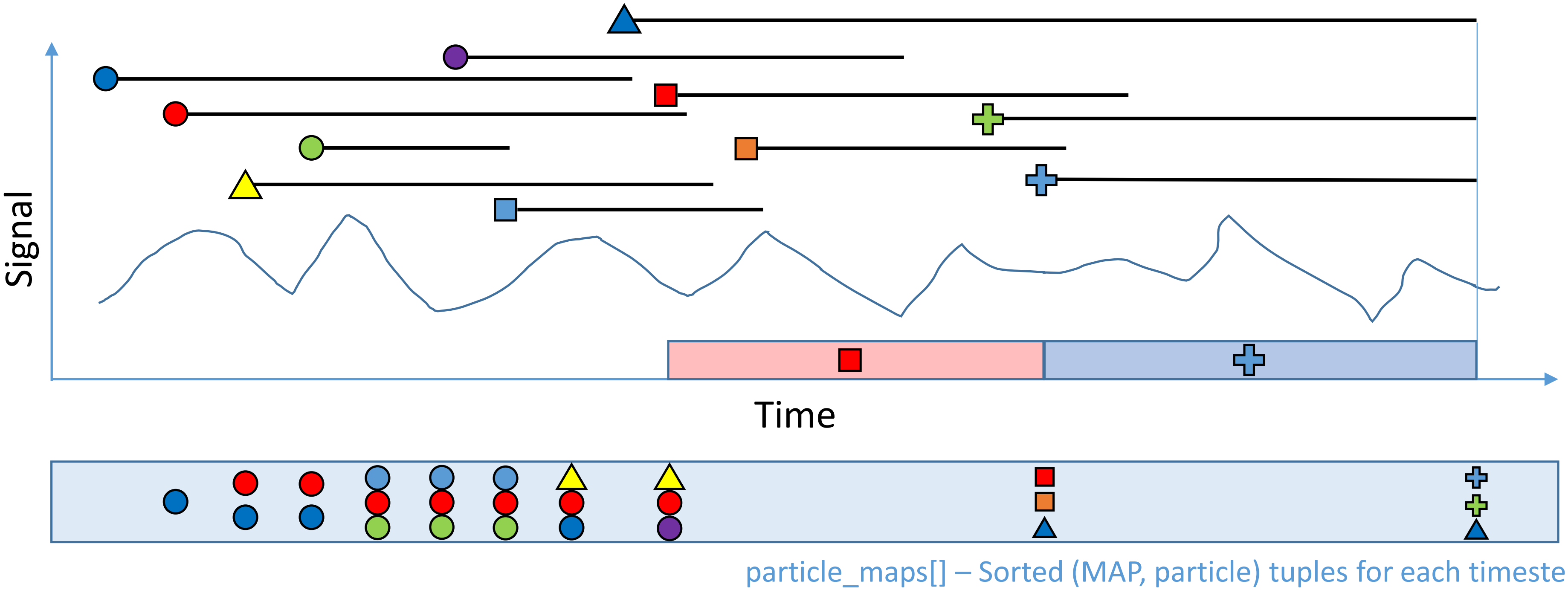
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
  - Take best (MAP, particle) at particle_maps index *f*
  - Annotate segment [*shape_start*, *f*] with *shape_class*
  - Set *f = particle.start_time*

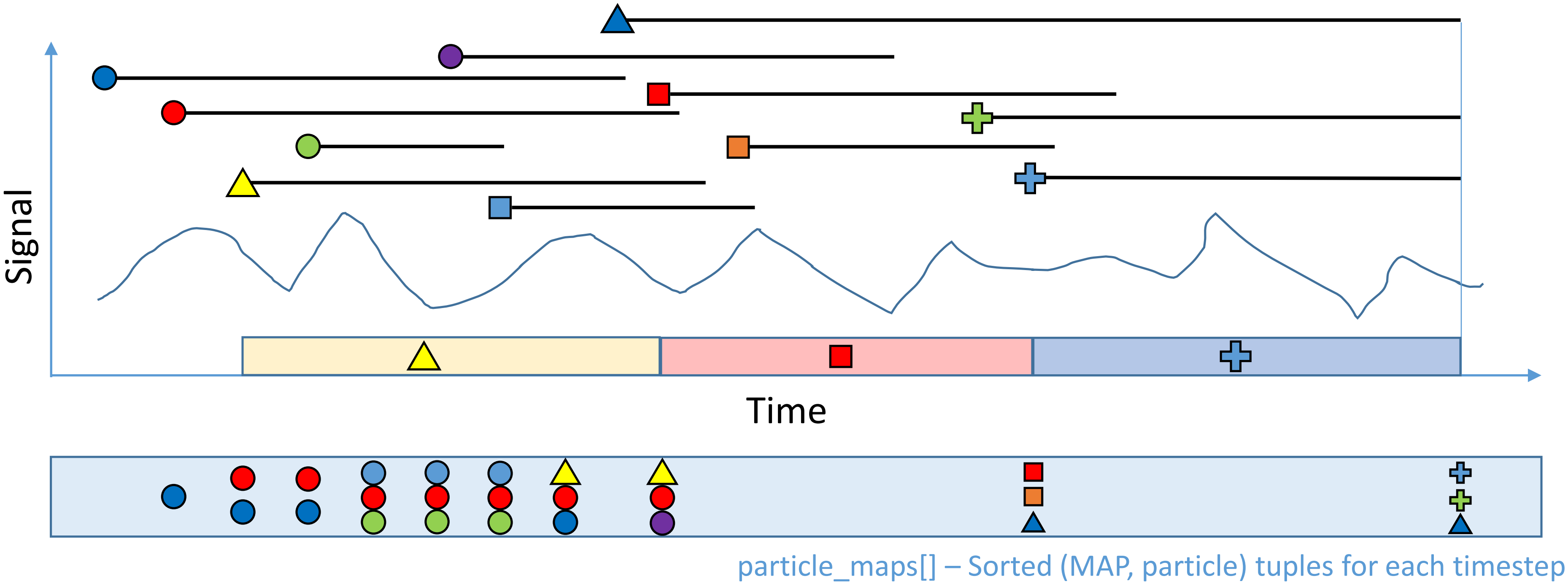
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
    - Take best (MAP, particle) at particle_maps index *f*
    - Annotate segment [*shape_start*, *f*] with *shape_class*
    - Set *f = particle.start_time*

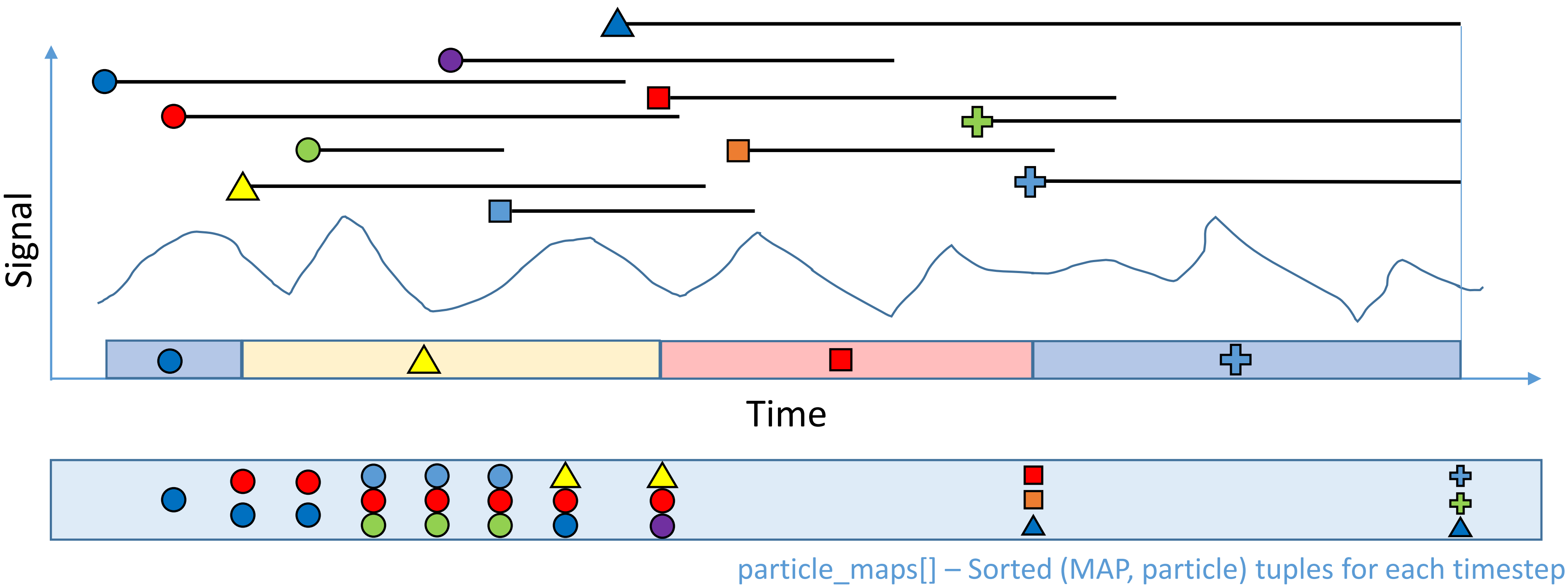
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
  - Take best (MAP, particle) at particle_maps index *f*
  - Annotate segment [*shape_start*, *f*] with *shape_class*
  - Set *f = particle.start_time*

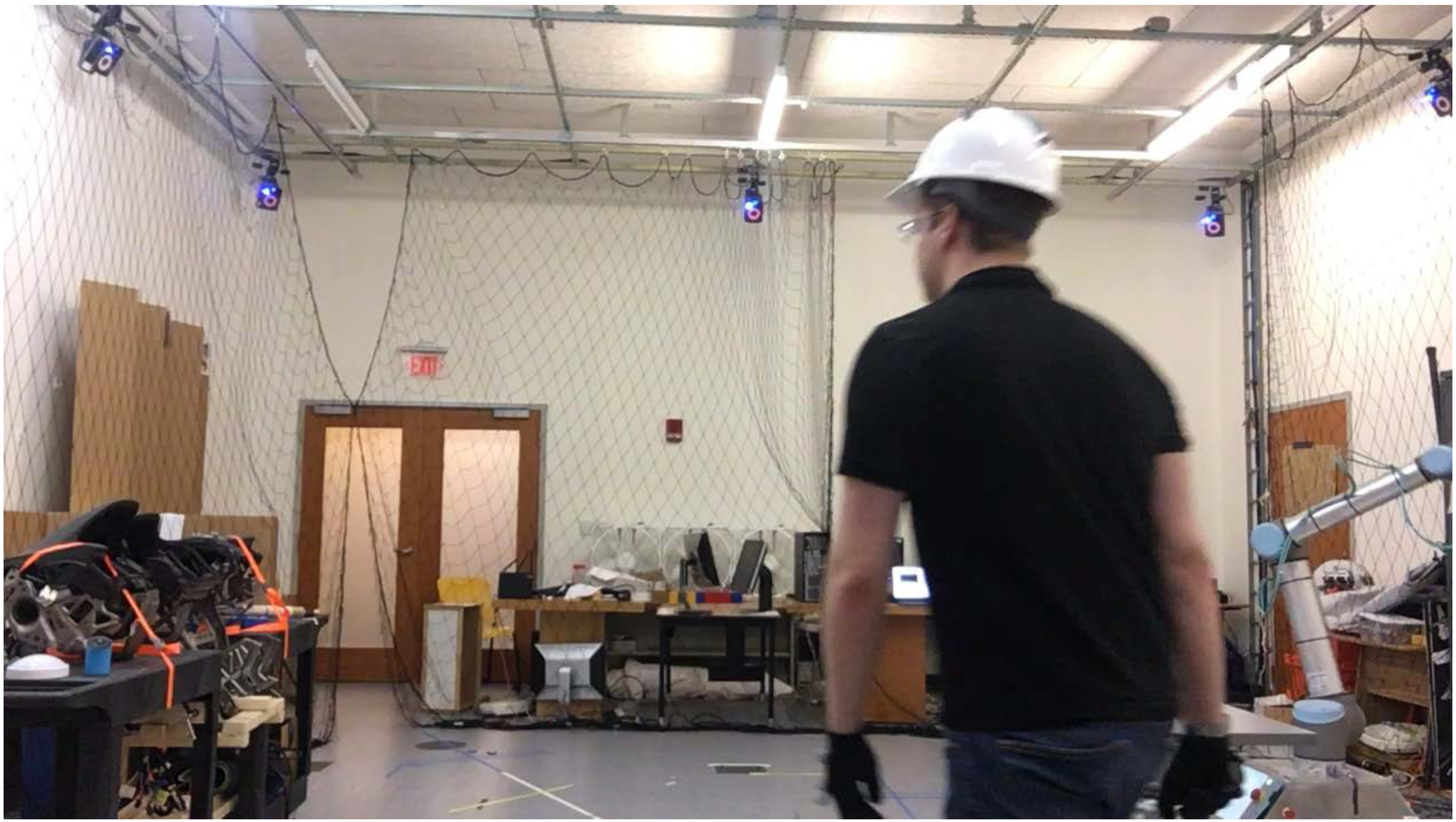
Particle Filtering for Changepoint Detection



To extract the most likely segmentation:

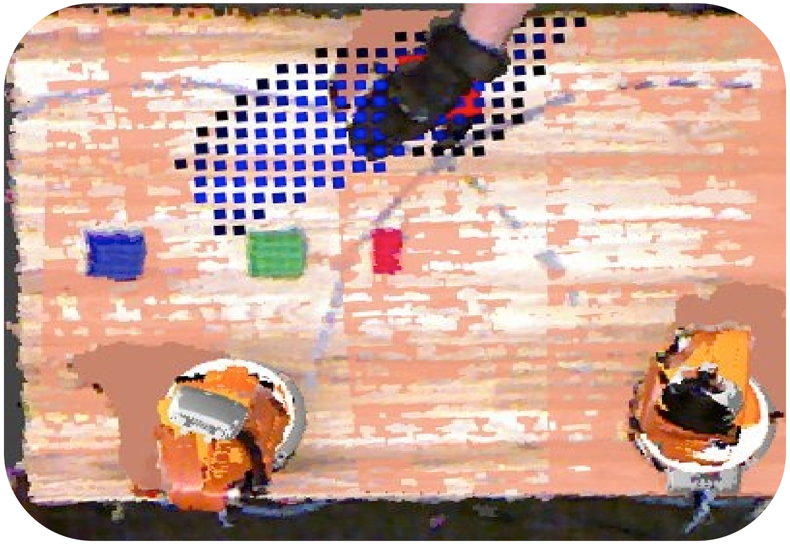
- *Set f = final frame index*
- While *f > 0* and *particle_maps[f] != None:*
    - Take best (MAP, particle) at particle_maps index *f*
    - Annotate segment [*shape_start*, *f*] with *shape_class*
    - Set *f = particle.start_time*

Associating Robot Behaviors with Task States



Motion models in collaborative settings

[Hayes and Scassellati ICDL13]



At a high level, *social force* is a projection of an agent's physical space occupation via its anticipated travel path

Social force carries different meanings depending on the task and environmental contexts in which it is applied.

Motion models in collaborative settings

Field treatment dictates robot's role

<u>Attractive</u>

Student

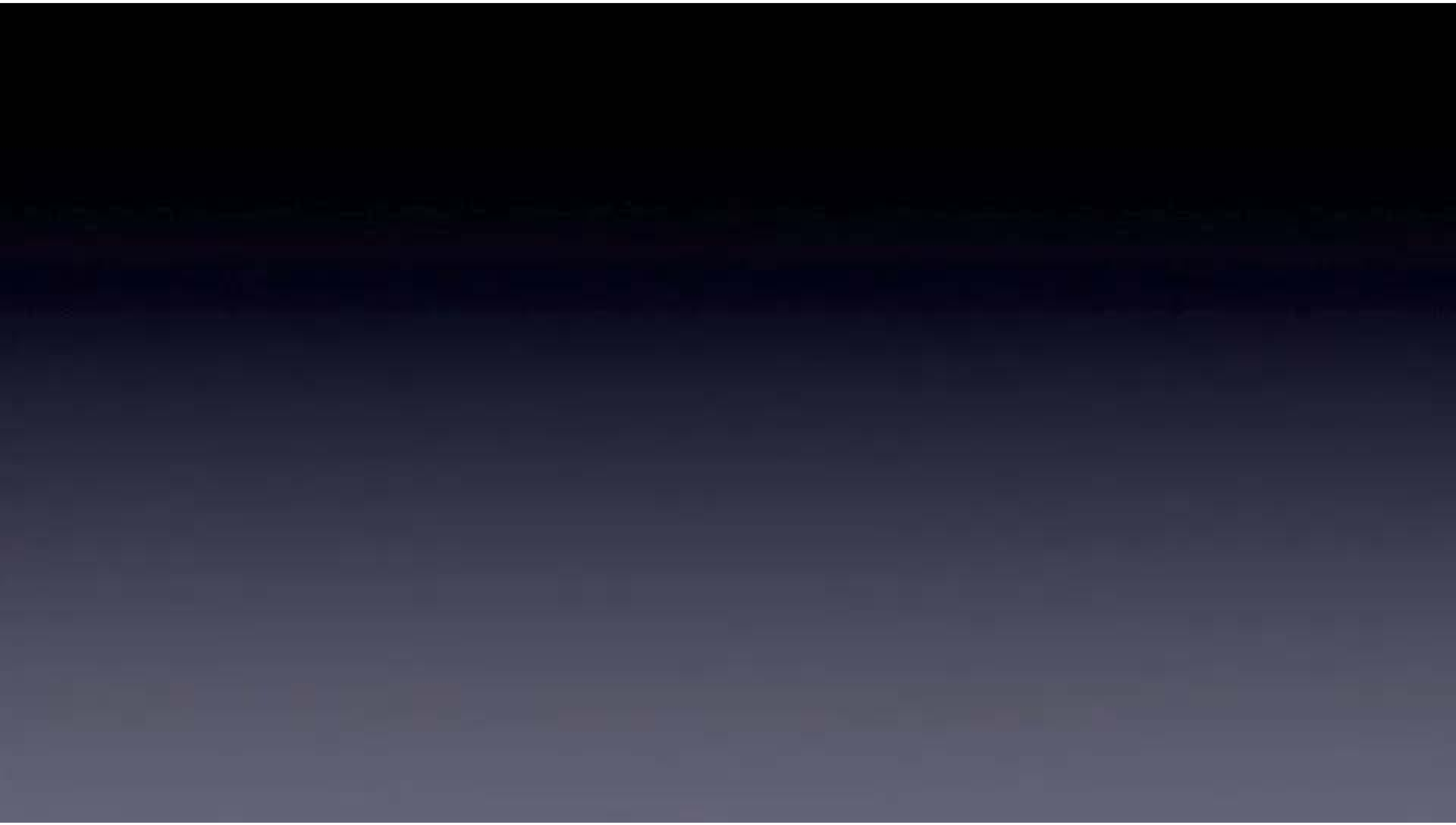
<u>Repulsive</u>

Peer

<u>Thresholded</u>

Instructor

Social Force in Human-Robot Teaming



Take a break (10 minutes)

Policy Shaping:

- Stand up
- Get Caffeine
- Go stand outside for a few minutes
- (Talk to someone about how IML relates to your work)

Cooperative Inverse Reinforcement Learning

[NIPS 2016]

Dylan Hadfield-Menell, Anca Dragan, Pieter Abbeel, Stuart Russell

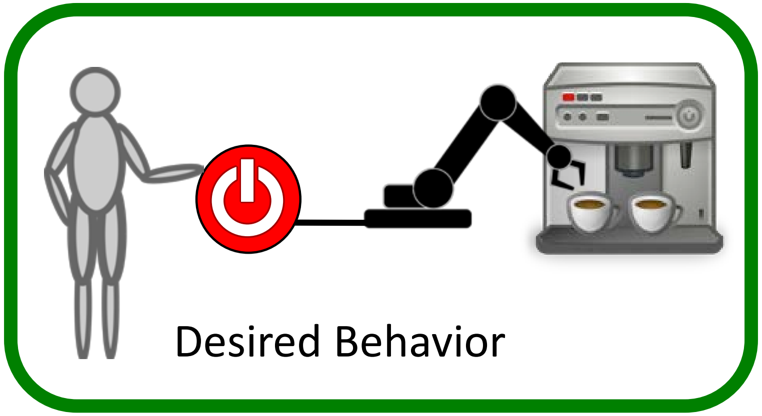
Inverse Reinforcement Learning Can Break Down in Team Scenarios!

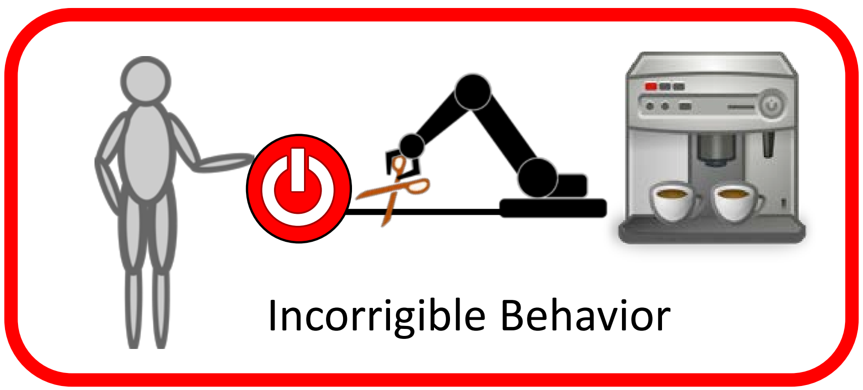
- Traditional IRL is optimal if the reference demonstrations are “Expert” demonstrations.
  - …but execution happens in isolation!

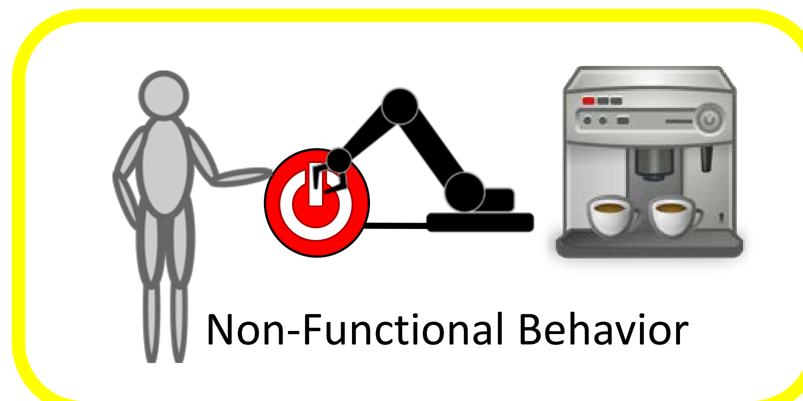
- Expert demonstrations are not always the most effective teaching strategy:
  - Sometimes better to learn the landscape of the problem than to see a optimal demonstrations

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

The Shutdown Problem



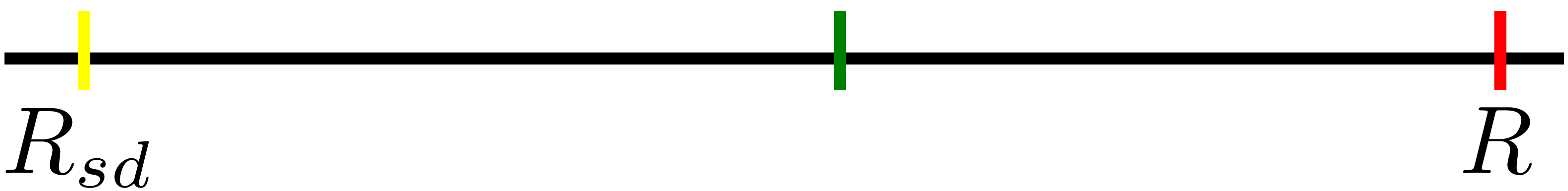




Slide credit: Dylan Hadfield-Menell, "Cooperative Inverse Reinforcement Learning", CITRIS Workshop on Algorithmic Human-Robot Interaction. INRIA 2016.

The Shutdown Problem

$$P(sd)R_{sd} + (1 - P(sd))R$$



Slide credit: Dylan Hadfield-Menell, "Cooperative Inverse Reinforcement Learning", CITRIS Workshop on Algorithmic Human-Robot Interaction. INRIA 2016.

Issues in Inverse Reinforcement Learning

- Uncertainty about task objectives is essential for cooperative behaviors
- IRL Pitfalls:
  - Don’t want to just imitate the demonstrator
  - Assumes the demonstrator is ‘unaware’ of being observed
  - Action selection is independent of reward uncertainty
- Without modeling reward uncertainty, robot gets narrow view of environment dynamics and reward
- From Ramachandran and Amir’s “Bayesian Inverse Reinforcement Learning”:
  - The optimal policy for an MDP with a distribution over reward functions $R \sim P(R)$ is one that maximizes reward according to the expectation of R.

Proposal: Robot Plays Cooperative Game

- Cooperative Inverse Reinforcement Learning
  - [Hadfield-Menell et al. NIPS 2016]
- Two players:

- Both players maximize a shared reward function, but only $\mathbf{H}$ observes the actual reward signal; $\mathbf{R}$ only knows a prior distribution on reward functions
  - $\mathbf{R}$ learns the reward parameters by observing $\mathbf{H}$

Slide credit: Dylan Hadfield-Menell, "Cooperative Inverse Reinforcement Learning", CITRIS Workshop on Algorithmic Human-Robot Interaction. INRIA 2016.

Cooperative Inverse Reinforcement Learning

$$\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$$

Action sets for human and robot

Distribution over (parameterized) reward functions

$$\langle \mathcal{S}, \{\mathcal{A}^{\mathbf{H}}, \mathcal{A}^{\mathbf{R}}\}, T, \{R, \Theta, P_0\}, \gamma \rangle$$

Both act to maximize $\mathbb{E}\left[\sum_t \gamma^t R(s_t, a_t; \theta)\right]$

[Hadfield-Menell arXiv '16]

Slide credit: Dylan Hadfield-Menell, "Cooperative Inverse Reinforcement Learning", CITRIS Workshop on Algorithmic Human-Robot Interaction. INRIA 2016.

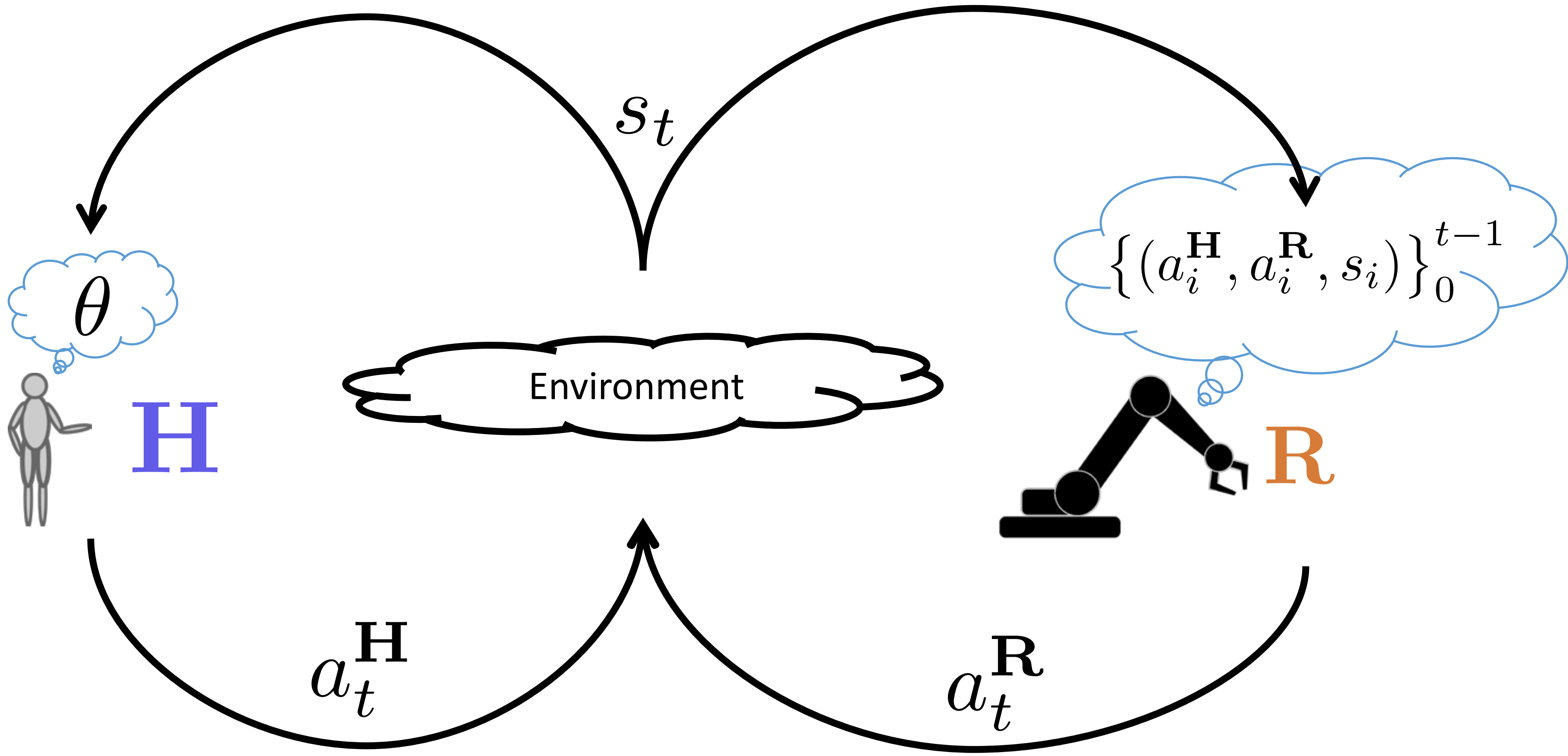
Cooperative Inverse Reinforcement Learning

$$\langle \mathcal{S}, \{\mathcal{A}^{\mathbf{H}}, \mathcal{A}^{\mathbf{R}}\}, T, \{R, \Theta, P_0\}, \gamma \rangle$$

- t=-1 $\theta \sim P_0(\theta)$
- t=0 $\mathbf{H}$ observes θ
- For t = 0, ...
  - $\mathbf{H}$ and $\mathbf{R}$ observe s_t
  - $\mathbf{H}$ and $\mathbf{R}$ select $a_t^{\mathbf{H}}$ and $a_t^{\mathbf{R}}$ respectively
  - New state s_{t+1} is sampled from $T(\cdot | s_t, a_t^{\mathbf{H}}, a_t^{\mathbf{R}})$
  - Both observe each other's actions and collect reward $R(s_t; \theta)$

Slide credit: Dylan Hadfield-Menell, "Cooperative Inverse Reinforcement Learning", CITRIS Workshop on Algorithmic Human-Robot Interaction. INRIA 2016.

Cooperative Inverse Reinforcement Learning



Slide credit: Dylan Hadfield-Menell, "Cooperative Inverse Reinforcement Learning", CITRIS Workshop on Algorithmic Human-Robot Interaction. INRIA 2016.

CIRL Properties

- The distribution over state sequences is determined by a pair of policies: ($\pi^{\mathbf{H}}$, $\pi^{\mathbf{R}}$)
- An ‘optimal’ policy pair maximizes the discounted sum of rewards
- In general, policies may depend on the entire observation histories
    - The history of states and actions for both actors includes the reward parameter for the human
    - [Hadfield-Menell ‘16] There exists an optimal policy pair that only depends on the current state and the robot’s belief

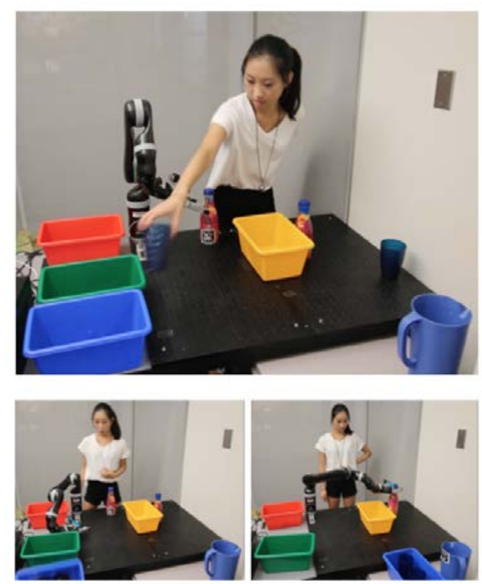
Incentives for Instructive Demonstrations

- Reduces the robot’s expected regret
- Reduces the KL Divergence of trajectory distributions
- Reduces reward errors

Further reading:

Game-Theoretic Modeling of Human Adaptation in Human-Robot Collaboration by Nikolaidis et al.
HRI 2017

Extends CIRL, providing a model of human partial adaptation to a robot collaborator without adopting the robot’s policy as their own.

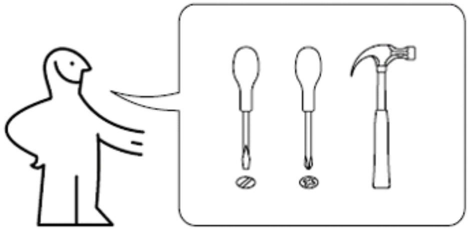


Effective Robot Teammate Behaviors for Supporting Sequential Manipulation Tasks

[IROS 2015]

Bradley Hayes and Brian Scassellati

Can we do better than LfD for Skill Acquisition?



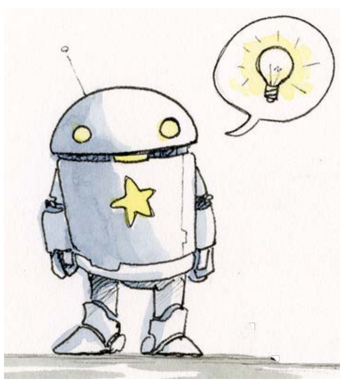
Demonstration-based Methods

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

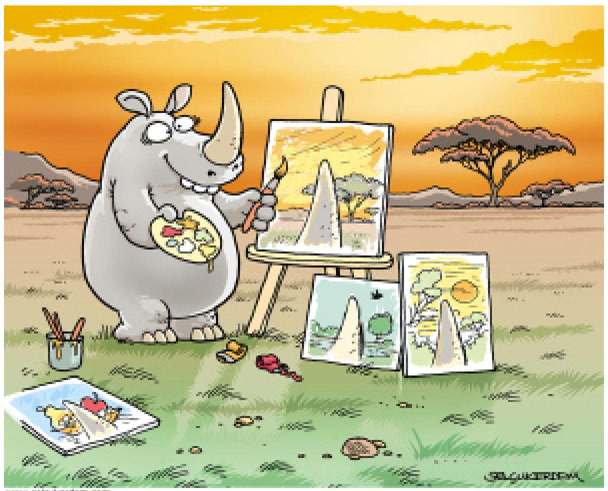


Goal-driven Methods

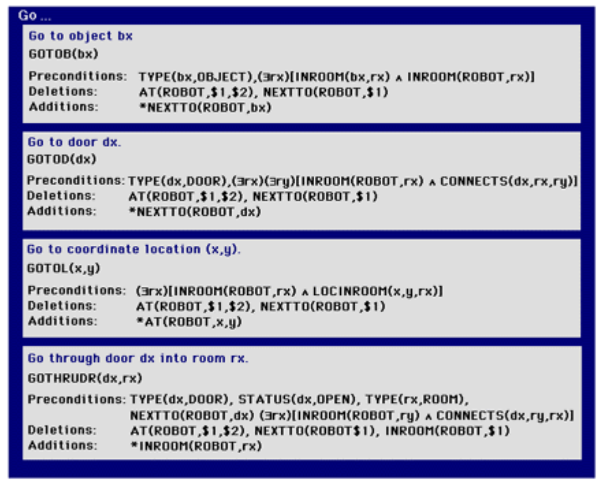
Robot figures out ***how*** and ***when*** it can be helpful

- Allows for novel behaviors to be discovered
- Enables deeper task comprehension and action understanding

Autonomously Generating Supportive Behaviors:
A Task and Motion Planning Approach



Perspective Taking



Go ...

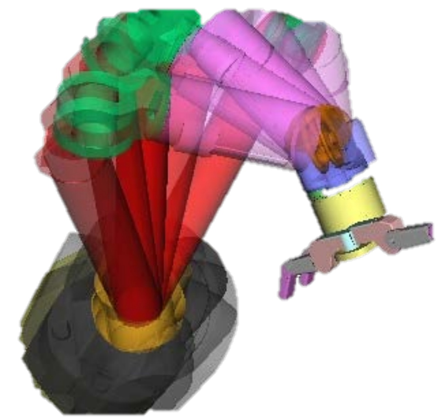
Go to object bx
GOTOB(bx)
Preconditions: TYPE(bx,OBJECT),(∃rx)[INROOM(bx,rx) ∧ INROOM(ROBOT,rx)]
Deletions: AT(ROBOT,$1,$2), NEXTTO(ROBOT,$1)
Additions: *NEXTTO(ROBOT,bx)

Go to door dx.
GOTOD(dx)
Preconditions: TYPE(dx,DOOR),(∃rx)(∃ry)[INROOM(ROBOT,rx) ∧ CONNECTS(dx,rx,ry)]
Deletions: AT(ROBOT,$1,$2), NEXTTO(ROBOT,$1)
Additions: *NEXTTO(ROBOT,dx)

Go to coordinate location (x,y).
GOTOL(x,y)
Preconditions: (∃rx)[INROOM(ROBOT,rx) ∧ LOCINROOM(x,y,rx)]
Deletions: AT(ROBOT,$1,$2), NEXTTO(ROBOT,$1)
Additions: *AT(ROBOT,x,y)

Go through door dx into room rx.
GOTHRUDR(dx,rx)
Preconditions: TYPE(dx,DOOR), STATUS(dx,OPEN), TYPE(rx,ROOM), NEXTTO(ROBOT,dx) (∃rx)[INROOM(ROBOT,ry) ∧ CONNECTS(dx,ry,rx)]
Deletions: AT(ROBOT,$1,$2), NEXTTO(ROBOT$1), INROOM(ROBOT,$1)
Additions: *INROOM(ROBOT,rx)

Symbolic planning



Motion planning

Autonomously Generated Supportive Behaviors

Task and Motion Planning

The TAMP problem is represented by the tuple: $\{A, O, C, s_0, s_G\}$

A is a set of (lead) agents

O is a set of operators (unparameterized motor primitives)

C is a capabilities mapping function between agents and operators

$\mathbf{s_0}$ is the set of predicates *precisely* specifying the start state

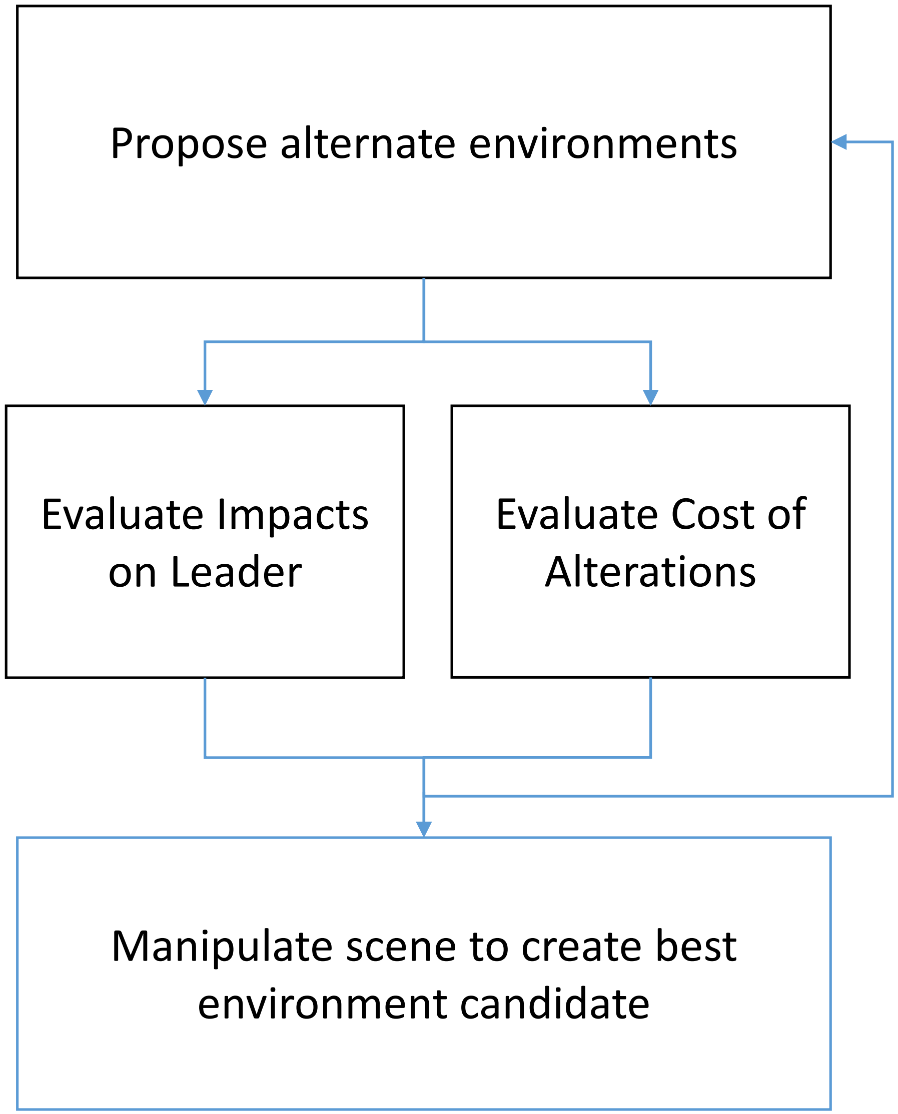
$\mathbf{s_G}$ is the set of predicates specifying the goal state

Supportive Behavior TAMP

The SB-TAMP problem is represented as the tuple: { T, Π_T, a_s, C_s, s_c, P }

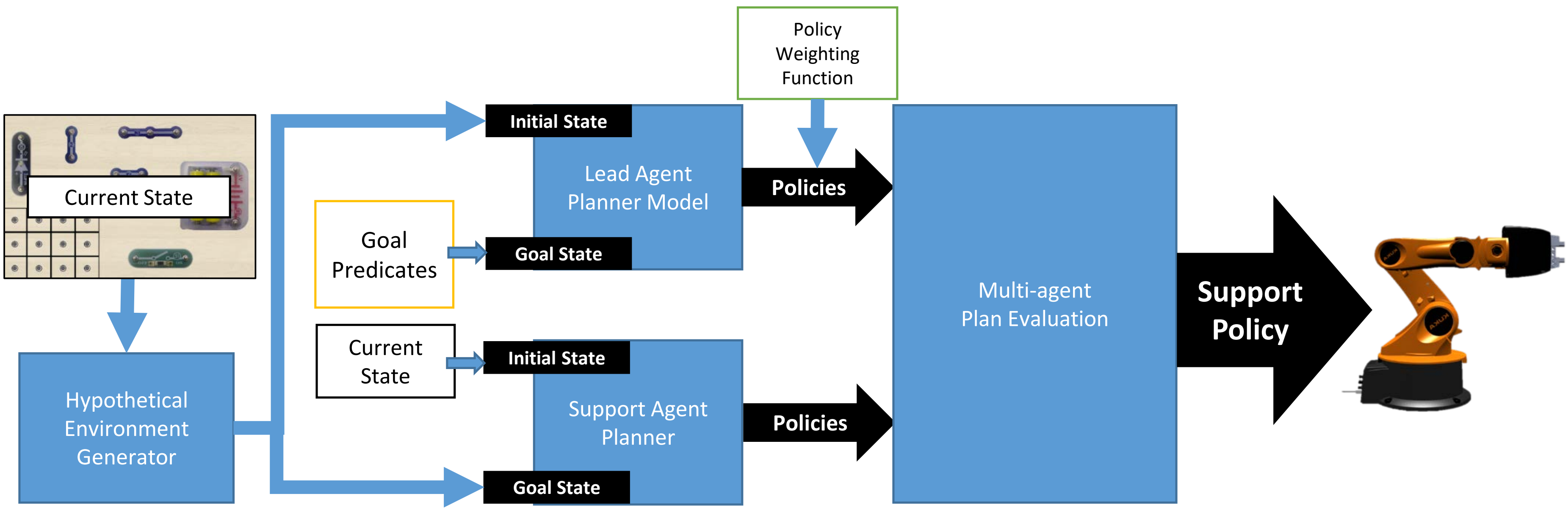
- **T** is a TAMP problem
- $\boldsymbol{\Pi_T}$ is a set of symbolic plans for T
- $\boldsymbol{a_s}$ is a supportive agent
- $\boldsymbol{C_s}$ is a mapping function indicating operators from T usable by a_s
- $\boldsymbol{s_c}$ is the current environment state
- **P** is a set of partially or fully specified predicates describing prohibited environmental effects for support actions

Supportive Behavior Pipeline: Intuition

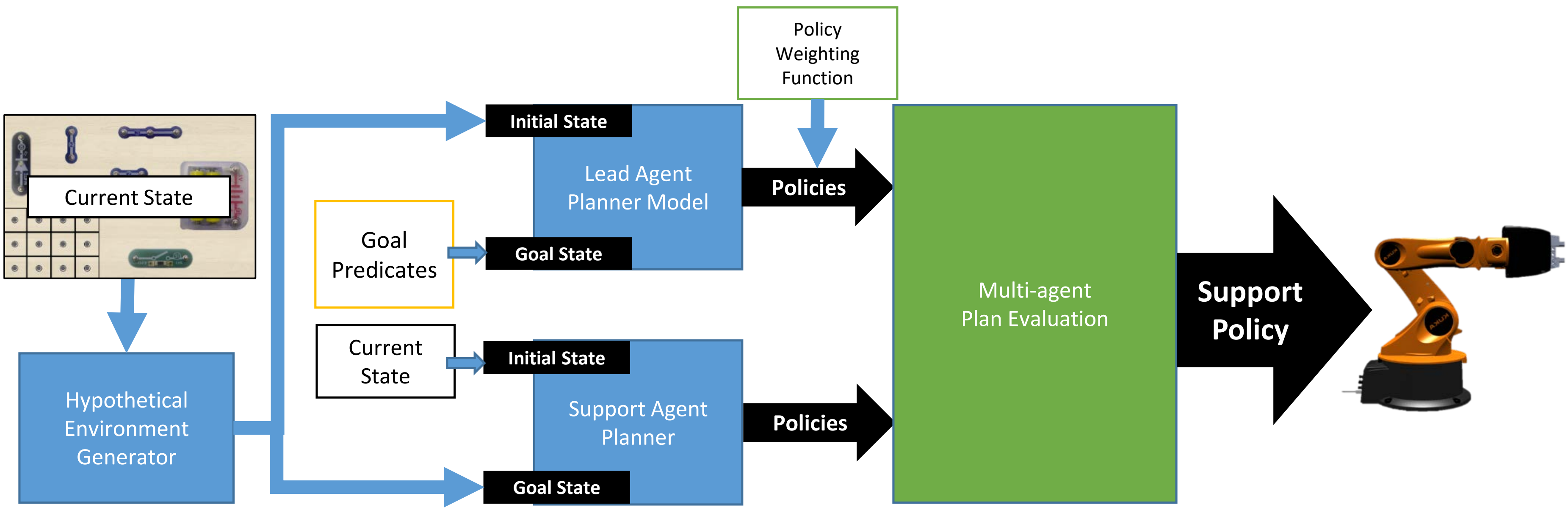


1. Propose alternative environments
   - Change **one** thing about the environment
2. Evaluate if they facilitate the leader's task/motion planning
   - Simulate policy execution(s) from leader's perspective
3. Compute cost of creating target environment
   - Simulate support agent's plan execution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

Supportive Behavior Pipeline



Supportive Behavior Pipeline



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 ($\mathbf{s_c}$)

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

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

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 ($\mathbf{s_c}$)

- Cost estimate
  - Resource
  - Spatial co

ation/demand)

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

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

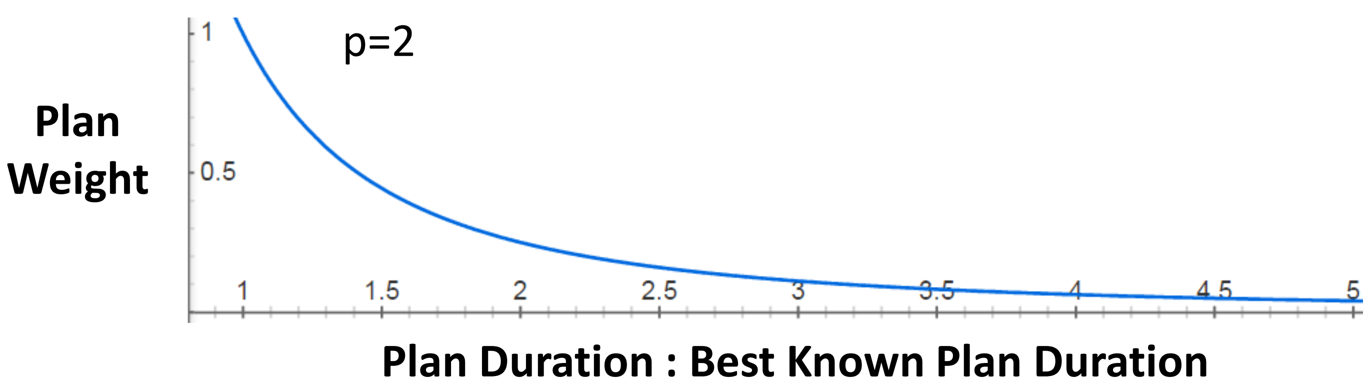
Only the best-known solution is worth planning against

Weighting functions: Uniform

Weighting functions: Optimality-Proportional

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

Weight plans proportional to similarity vs. the best-known solution



Weighting functions: Optimality-Proportional

Weighting functions: Error Mitigation

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

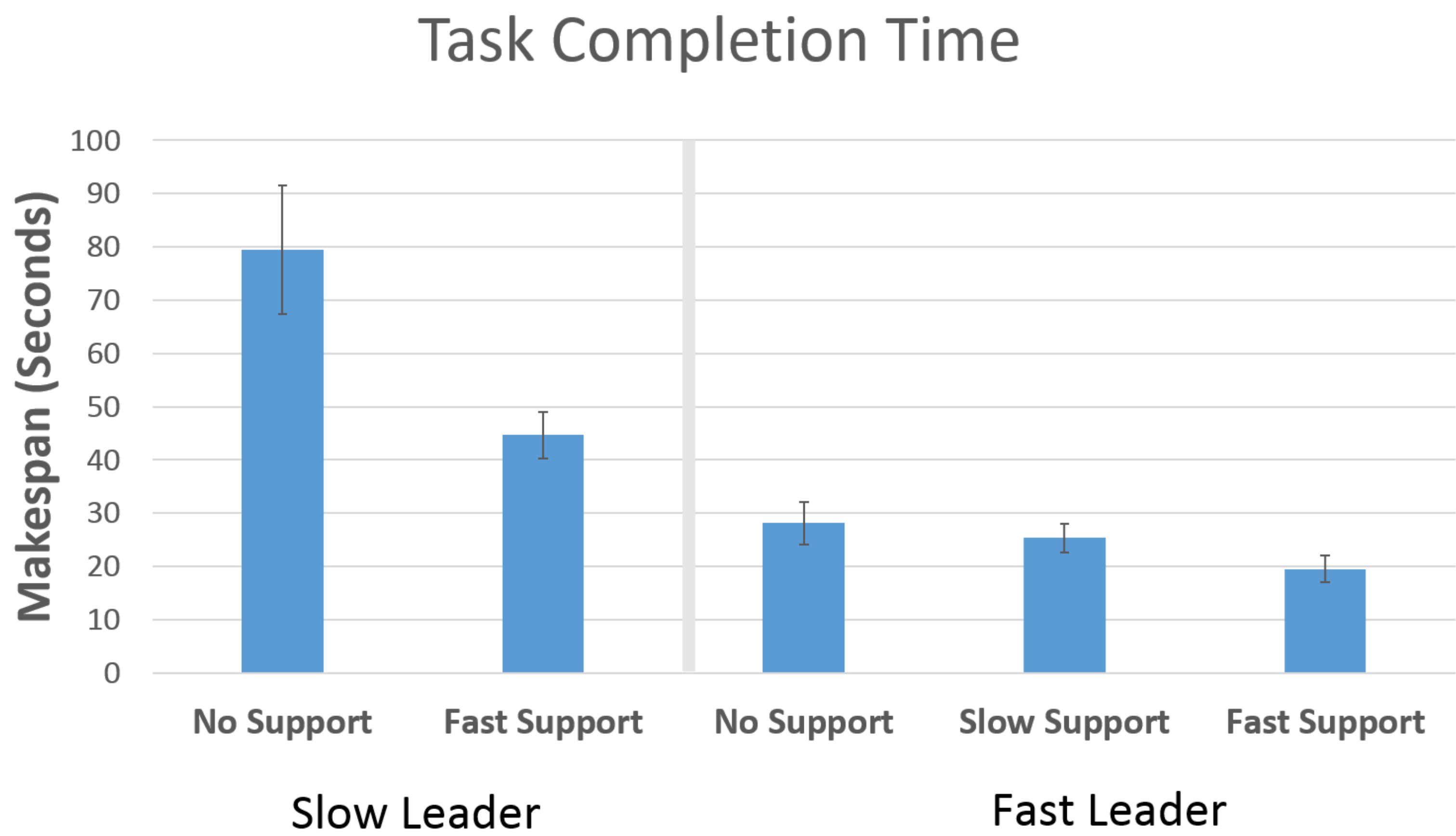
Plans more optimal than some cutoff ε are treated normally, per *f*.

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

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

Weighting functions: Error Mitigation

Effect of Supportive Behaviors



Improving Robot Controller Transparency Through Autonomous Policy Explanation

[HRI 2017]

Bradley Hayes and Julie Shah

Shared Expectations are Critical for Teamwork

In close human-robot collaboration…

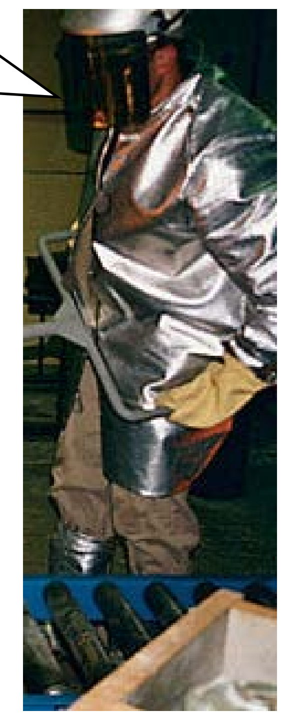
- Human must be able to plan around expected robot behaviors
- Understanding failure modes and policies are central to ensuring safe interaction and **managing risk**

Fluent teaming **requires** communication…

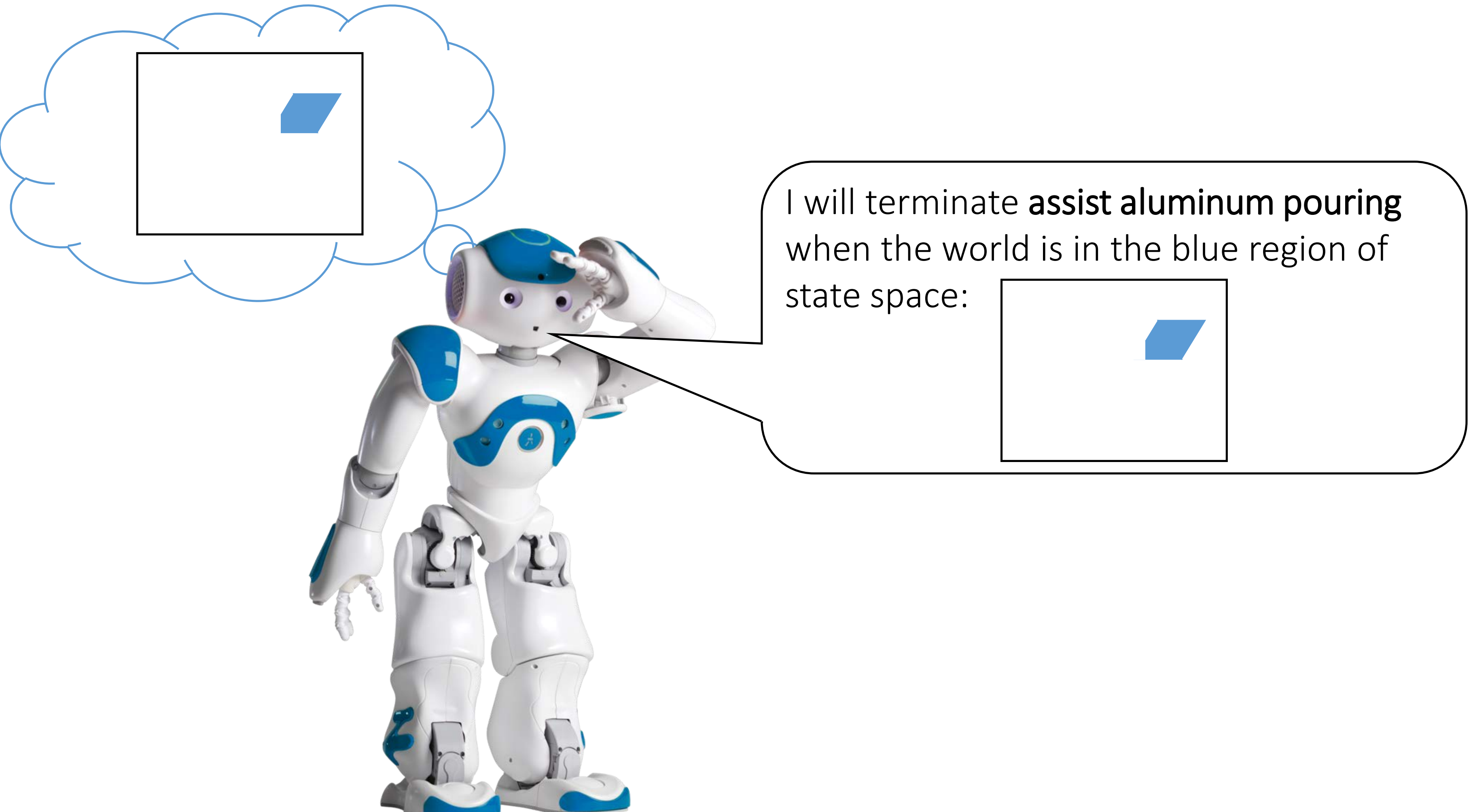
- When there's no prior knowledge
- When expectations are violated
- When there is joint action

Semantics for Policy Transfer

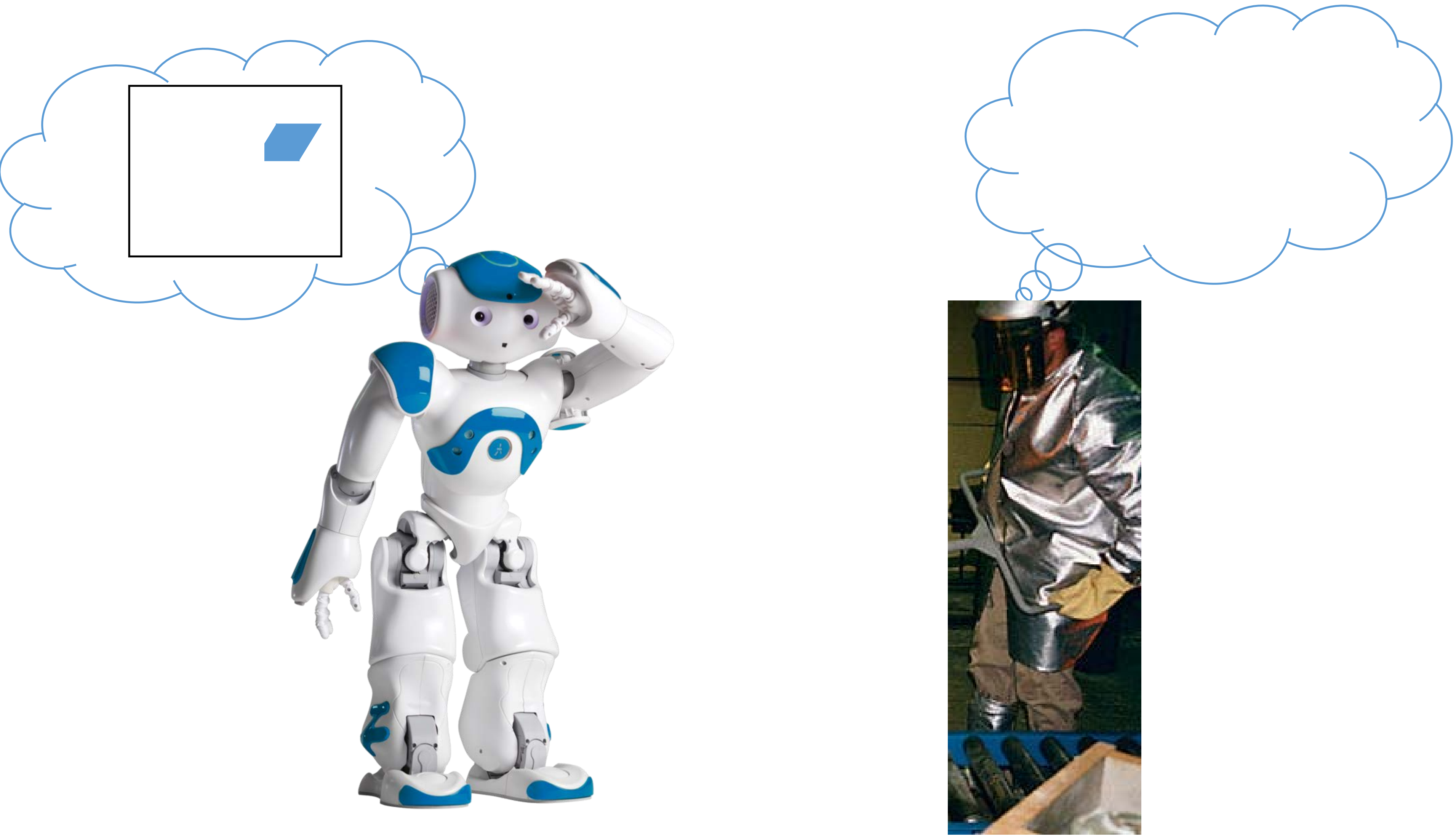
When will you stop
helping me pour the
molten aluminum?



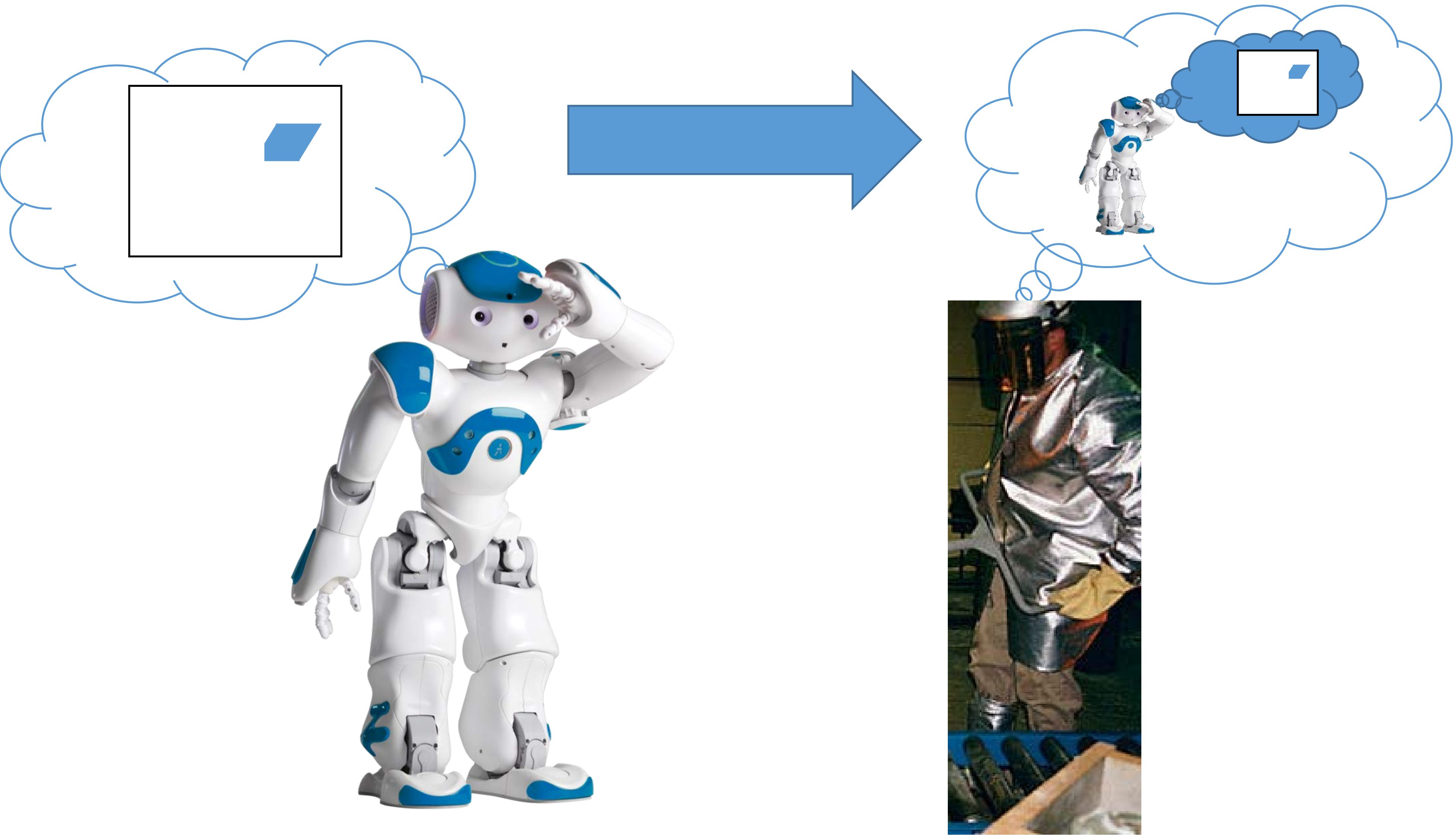
Semantics for Policy Transfer

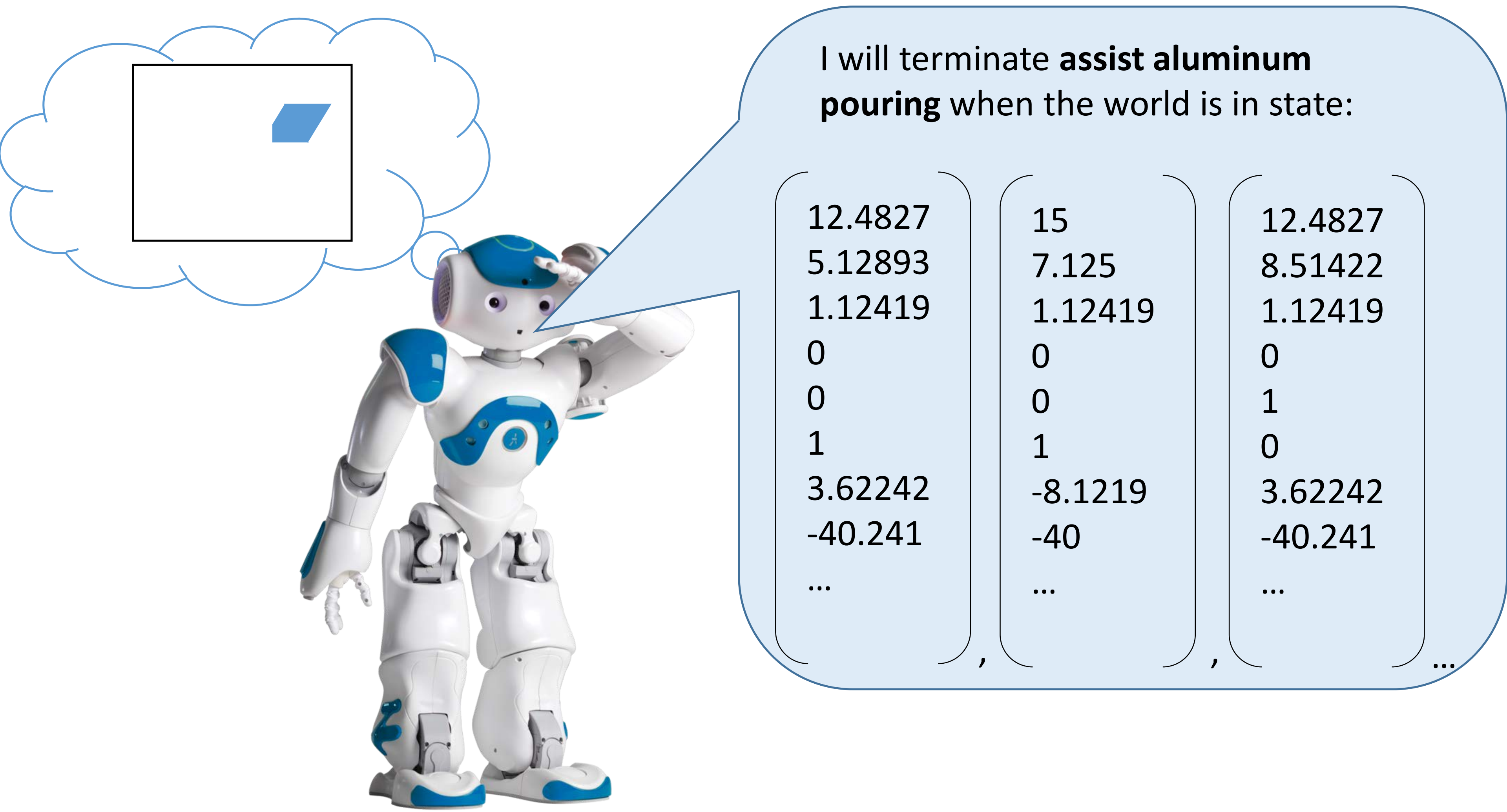


Semantics for Policy Transfer



Semantics for Policy Transfer



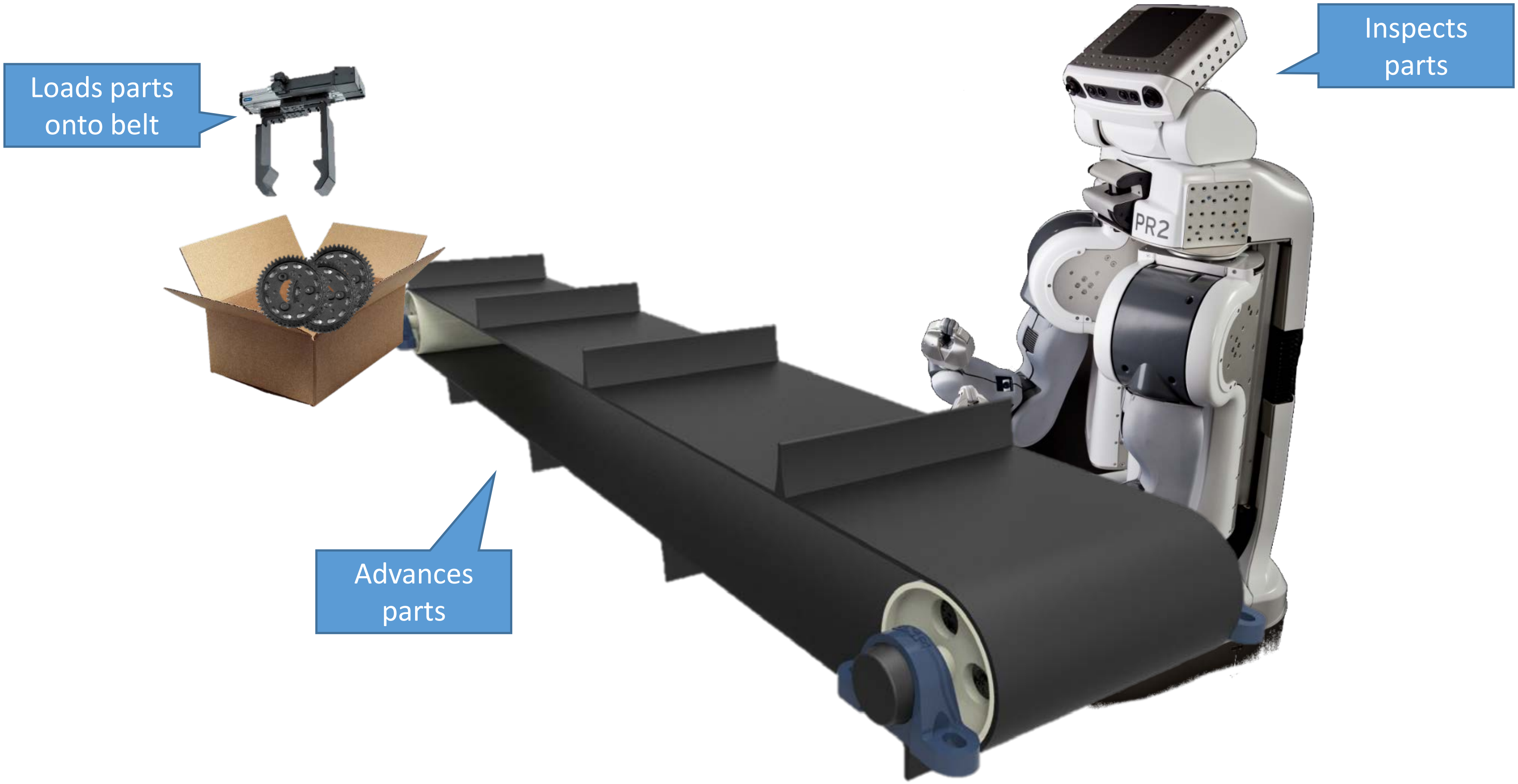

I will terminate **assist aluminum pouring** when the world is in state:
12.4827
5.12893
1.12419
0
0
1
3.62242
-40.241
...
,
15
7.125
1.12419
0
0
1
-8.1219
-40
...
,
12.4827
8.51422
1.12419
0
1
0
3.62242
-40.241
...
...

State space is too obscure to directly articulate

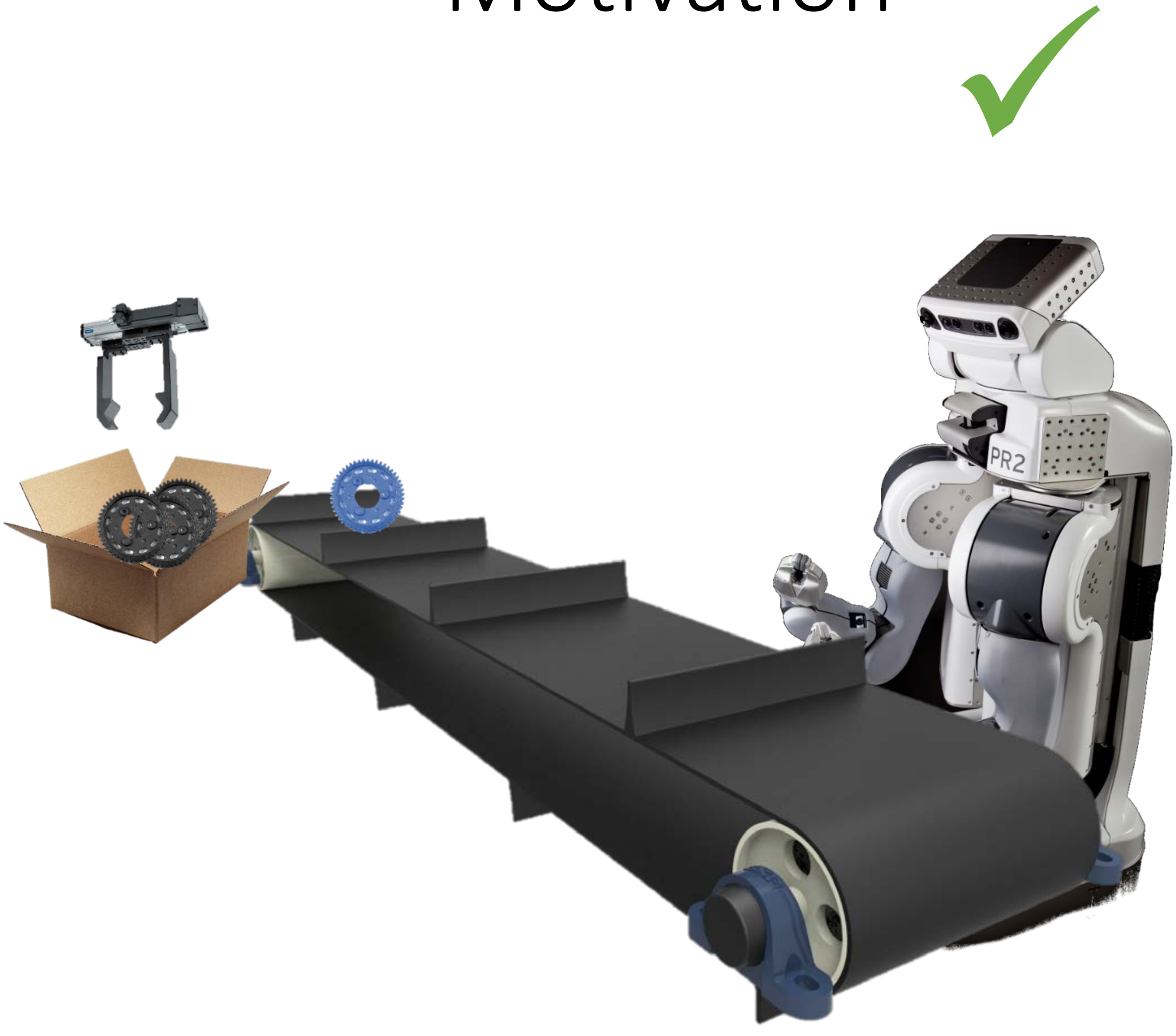
I will terminate **assist aluminum pouring** when the world is in state:

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

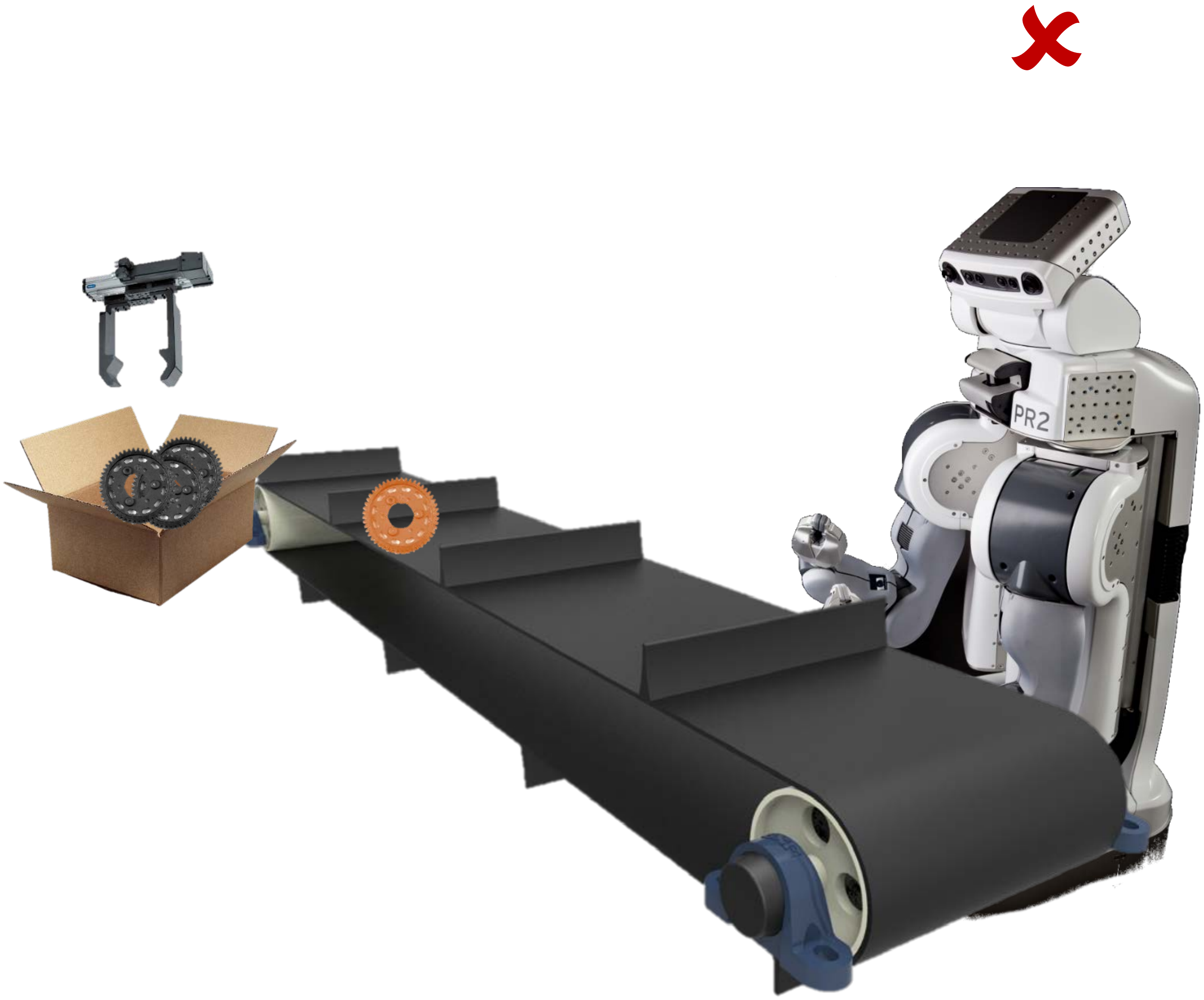
Motivation



Motivation

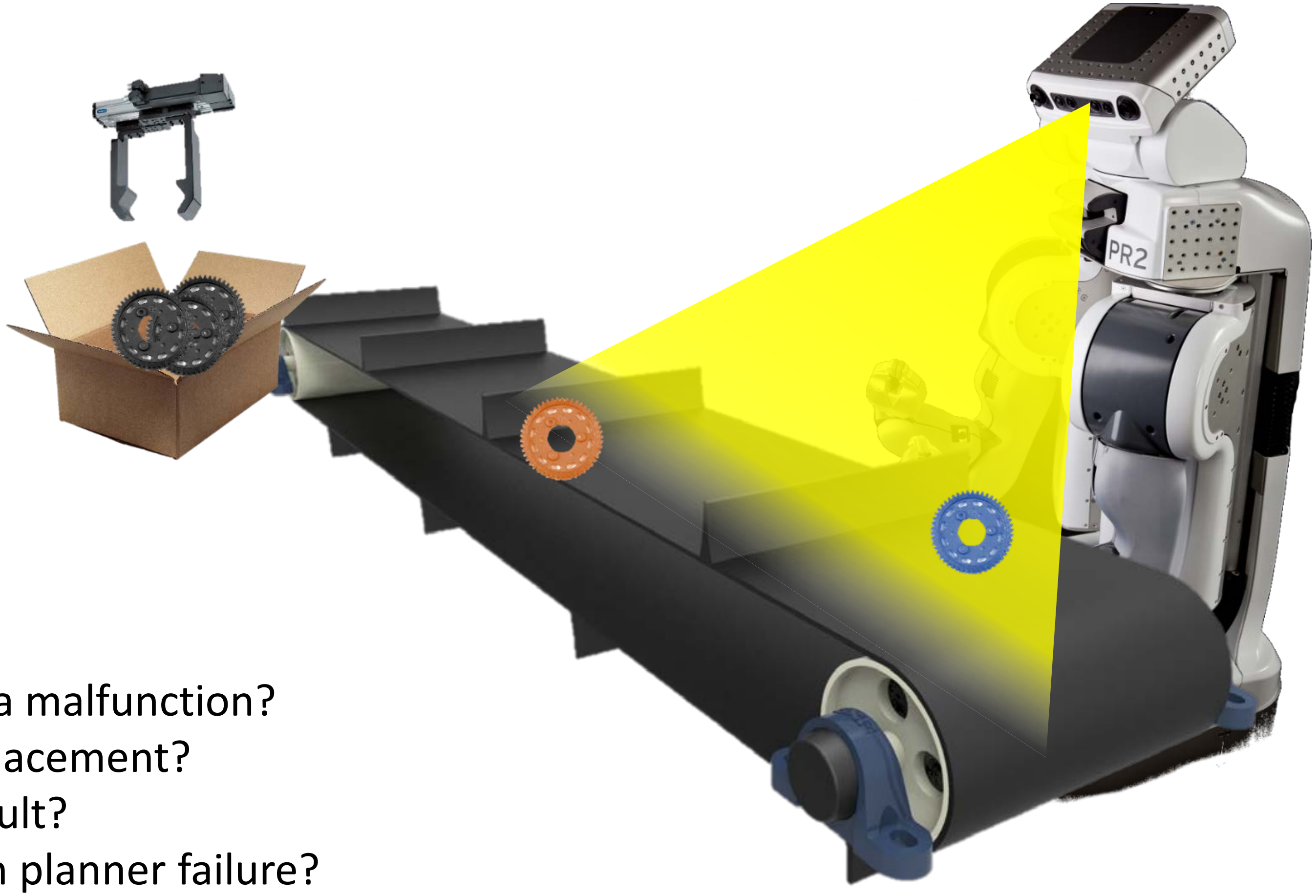


Motivation



Motivation

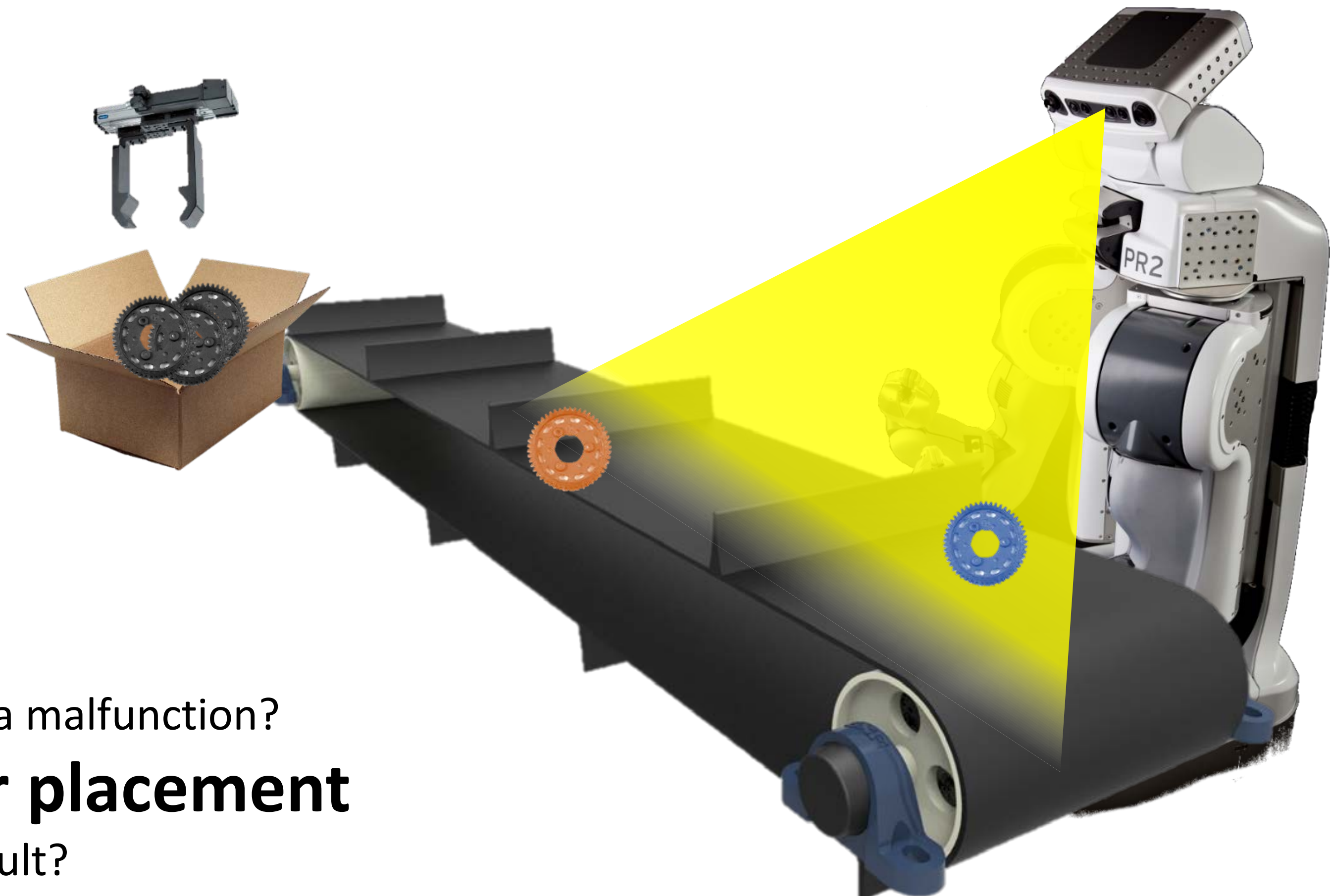
Why did the robot not inspect the orange gear?



Camera malfunction?
Poor placement?
Arm fault?
Motion planner failure?
Incorrect policy?

Motivation

How do we diagnose and repair this fault?



Camera malfunction?

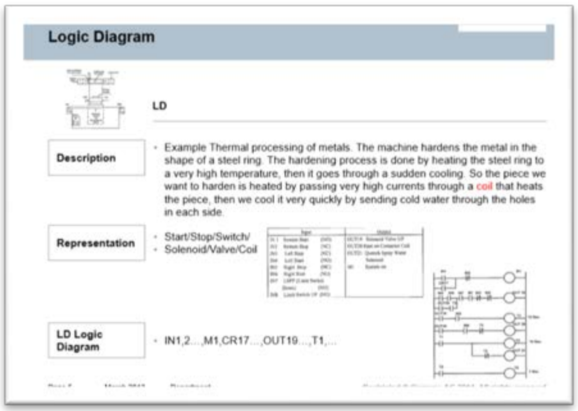
Poor placement

Arm fault?

Motion planner failure?

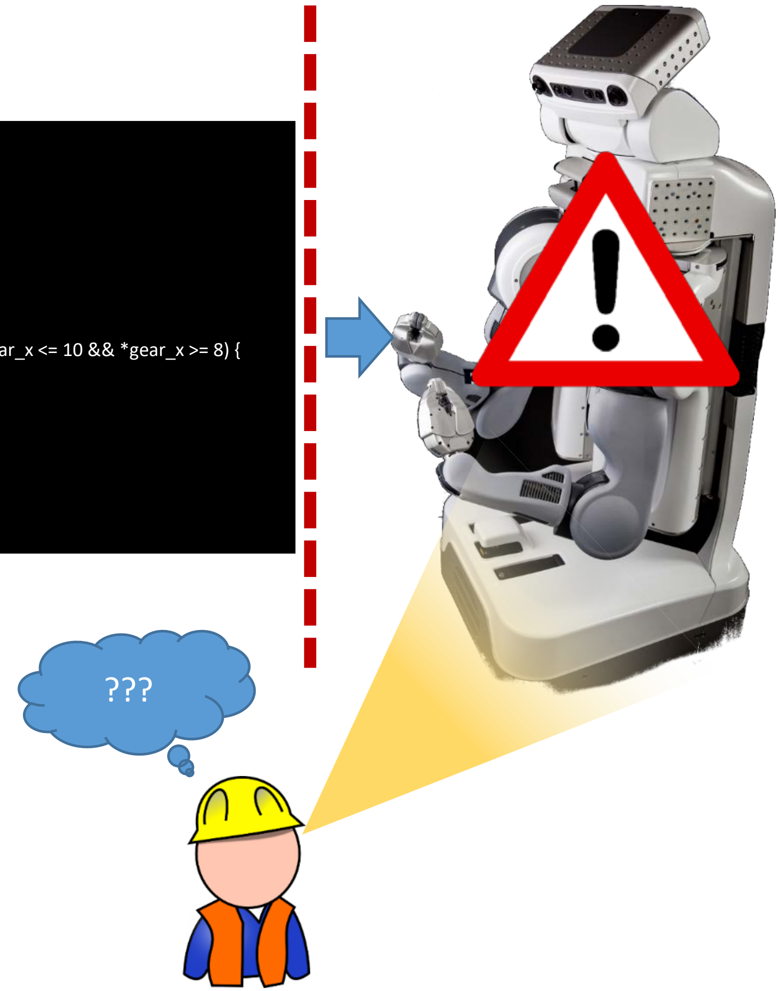
Incorrect policy?

State of the Art



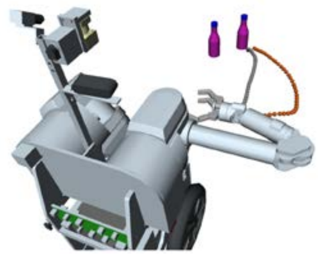

```
int *detect_gear = &INPUT1;
int *gear_x = &INPUT2;

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



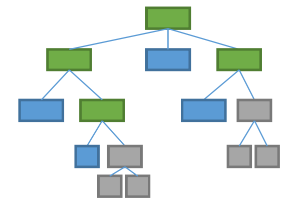
Establishing Shared Expectations

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



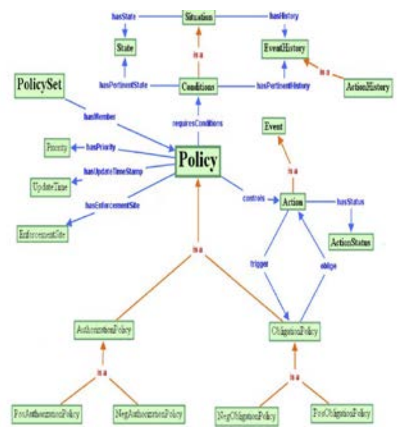
Legible Motion
[Dragan et al. 2013]

Coordination Graphs
[Kalech 2010]



Hierarchical Task Models
[Hayes et al. 2016]

State Disambiguation
[Wang et al. 2016]



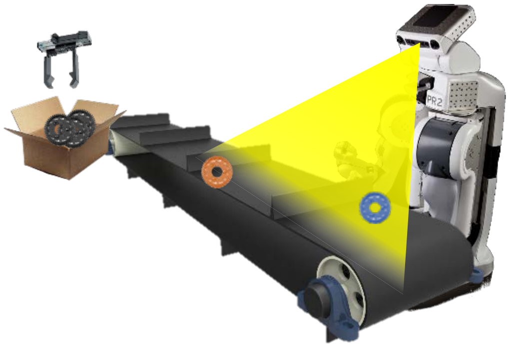
Policy Dictation
[Johnson et al. 2006]

Collaborative Planning
[Milliez et al. 2016]

Short Term

Long Term

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

Model Building

Query Analysis

Response Generation

Expectation Synchronization

Given

Required

Policy model

Concept representations

Mapping from query to model

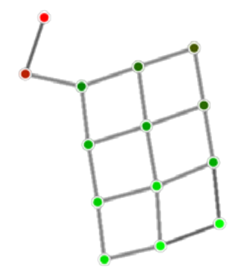
Mapping from model to response

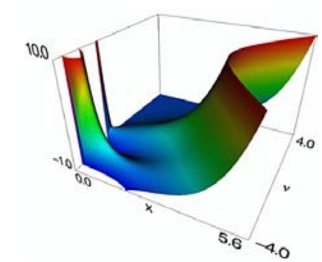
Policy Modeling

Local, approximate behavioral models from observation

(Generate MDP from regular controller operation)

Regular controller execution





Deployment or simulation environment

Behavioral MDP

Control policy

States are composed of internal variables and externally sensed information

Actions are parameterized function calls observed from the controller

Transitions are learned by observing resultant states from function calls

Given

Required

Policy model ✓

Concept representations

Mapping from query to model

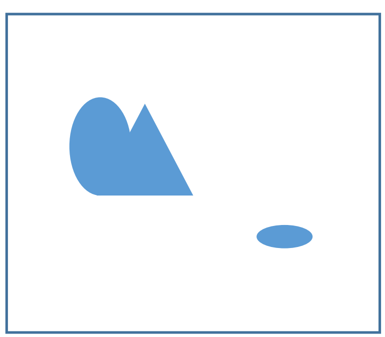
Mapping from model to response

Concept Representations

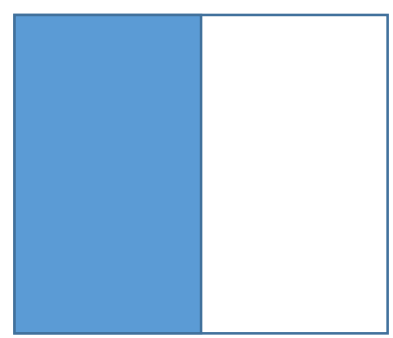
Concept library: generic state classifiers mapped to semantic templates that identify whether a state fulfills a given criteria

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

Given

Required

Policy model ✓

Concept representations ✓

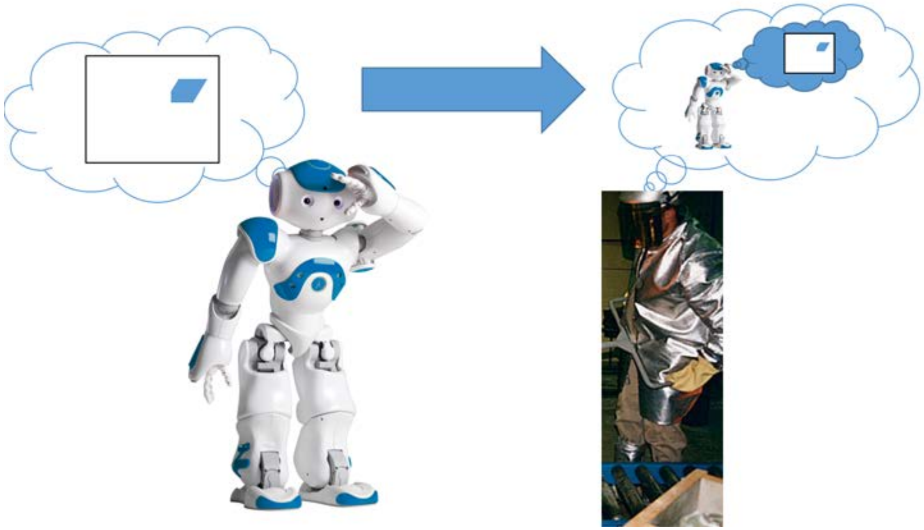
Mapping from query to model

Mapping from model to response

Improving Control Policy Transparency

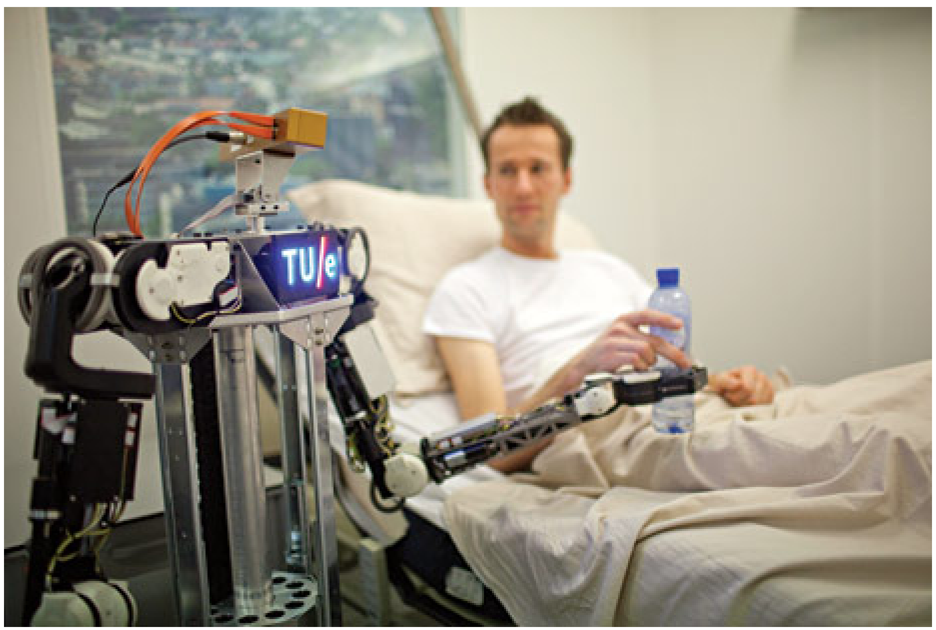
Three template questions for synchronizing expectations:

- When do you {*action*}?
- What will you do when {*environmental conditions*}?
- Why didn't you do {*action*}?



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 $S_{\pi^*\setminus a}$

$S_{\pi^a} \leftarrow \{\}$;

$S_{\pi^*\setminus a} \leftarrow \{\}$;

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 $S_{\pi^*\setminus a} \leftarrow S_{\pi^*\setminus a} \cup s$;

return $S_{\pi^a}, S_{\pi^*\setminus a}$;

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.

$S_{\pi^a} \leftarrow \{\}$;

$S_{\pi^* \backslash a} \leftarrow \{\}$;

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 $S_{\pi^* \backslash a} \leftarrow S_{\pi^* \backslash a} \cup s$;

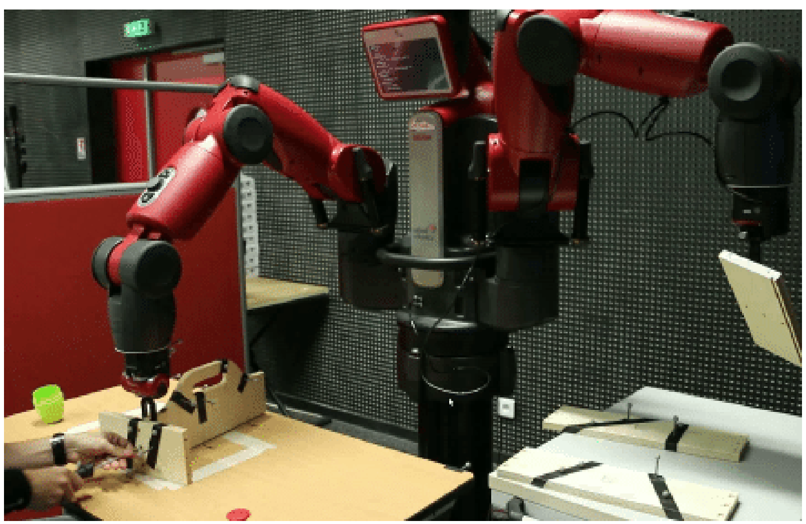
expected_region $\leftarrow$ describe($G, S_{\pi^a}, S_{\pi^* \backslash a}$);

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 ← 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] ← describe($S[a]$);
return descriptions;

Given

Required

Policy model ✓

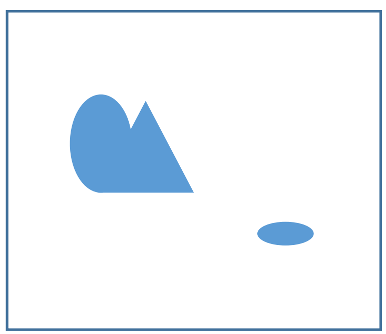
Concept representations ✓

Mapping from query to model ✓

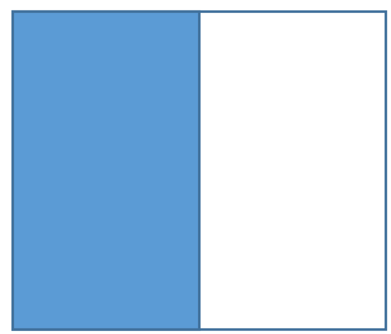
Mapping from model to response

Language Mapping: Model to Response

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



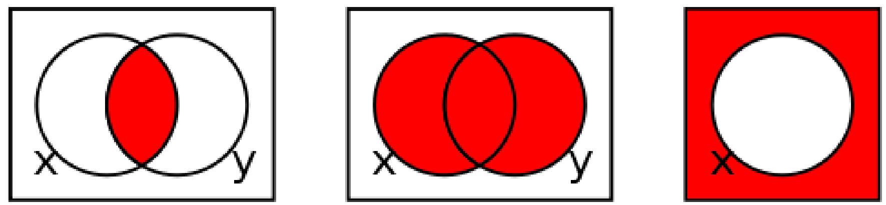
on_top(A,B)



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

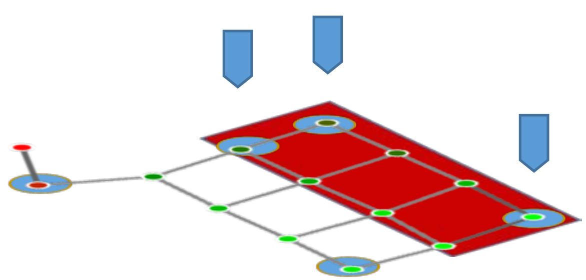
Find concept mapping that covers the indicated states

Detected_gear /\ at(conveyor_belt)

Convert to natural language

at(conveyor_belt)

Detected_gear

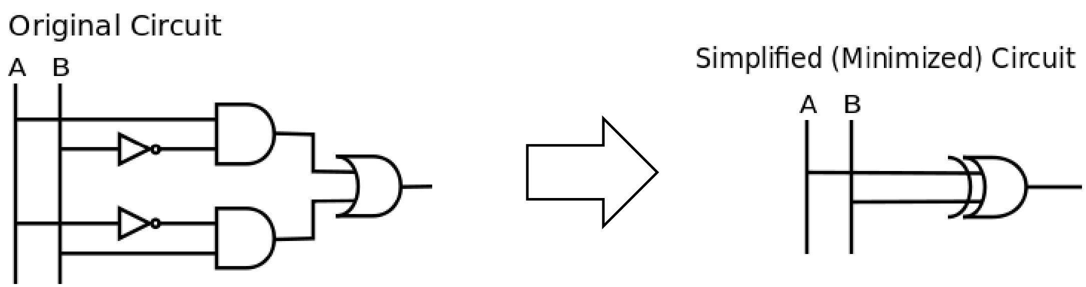


I'll inspect the gear when I've detected a gear and I'm at the conveyor belt.

Producing Efficient Summaries

Achieving the succinctness criterion is **NP-hard**.

(choosing the minimal set of concepts with the best state region coverage precision)



The same problems in succinctness are encountered during circuit minimization:

Prime implicants are **concept clauses** covering **minterms** (target states)

Can use Quine-McCluskey Algorithm to find minimization

Q-M doesn't scale well, but we can get approximate solutions using ESPRESSO, with appropriate sacrifices of optimality or precision.

Given

Required

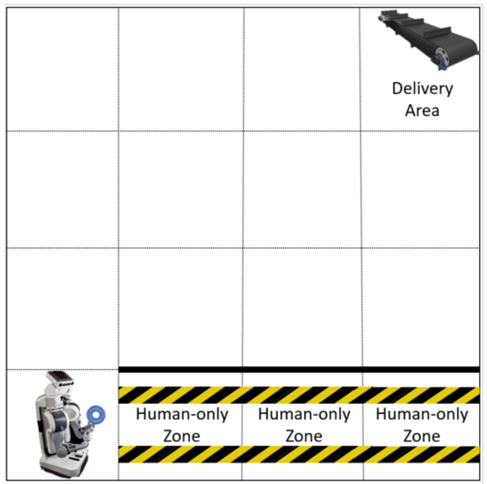
Policy model

Concept representations

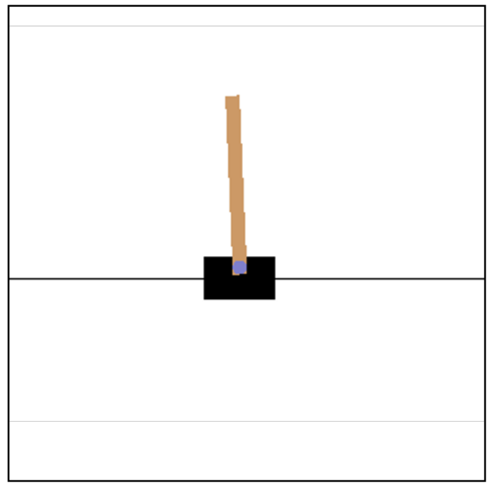
Mapping from query to model

Mapping from model to response

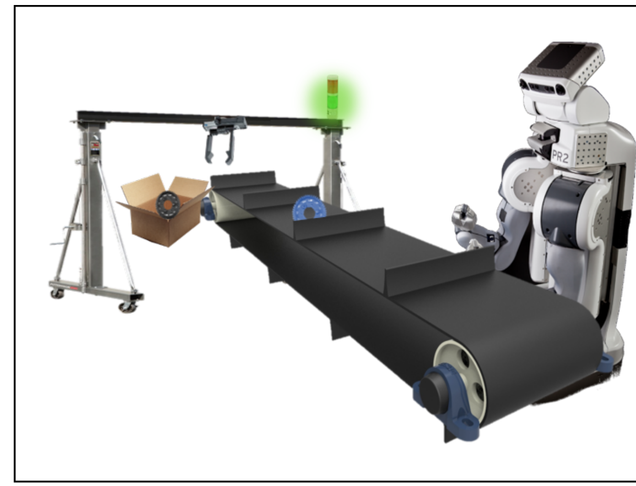
Result: Agents can explain their policies to collaborators



"I will get the gear when I am near a human-only zone and I do not carry a gear. I will move north when I am near a human-only zone and I carry a gear."



"I move right when the cart is at the far left or when the cart is in the middle and the pole is falling right or when the cart is in the far right and the pole is stabilizing left."



"I didn't inspect the part because the stock feed signal is off. I inspect the part when the stock feed signal is on and I have detected a part and the part is within reach."

Designing Interactions for Robot Active Learners

Maya Cakmak, Crystal Chao, Andrea L. Thomaz

[TAMD 2010]

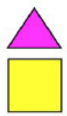
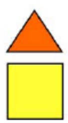
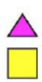
Passive vs. Active Learning

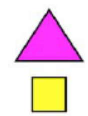
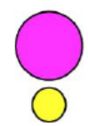
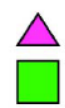
- Active Learning embeds robots in a tightly coupled dyadic interaction
  - Improper handling of this interaction disengages the oracle!
  - Disengagement leads to poor information quality
- Difficult to balance learning accuracy and learning speed with interaction smoothness
- Active Learning is a tool for increasing learner transparency

Experiment: Teaching Shape Composites

Concept Name	Concept Representation	Examples	# of Instances
HOUSE	$shape_{top} = triangle \wedge$ $color_{top} = pink \wedge$ $shape_{bottom} = square$		16
SNOWMAN	$shape_{top} = circle \wedge$ $size_{top} = small \wedge$ $shape_{bottom} = circle$		28
ALIEN	$shape_{top} = circle \wedge$ $color_{top} = green \wedge$ $color_{bottom} = green$		10
ICE CREAM	$shape_{top} = circle \wedge$ $shape_{bottom} = triangle \wedge$ $color_{bottom} = yellow$		16

Version Space Learning

Step	Example	Label	Version Space
1	 <*pink,triangle,large,* *yellow,square,large*>	+	**Most specific hypothesis:** <*pink,triangle,large* *yellow,square,large*> **Most general hypotheses:** <*pink,∗,∗,∗,∗,∗*> <*∗,triangle,∗,∗,∗,∗*> . . . <*∗,∗,∗,∗,∗,large*>
2	 <*orange,triangle,large,* *yellow,square,large*>	-	**Most specific hypothesis:** <*pink,triangle,large* *yellow,square,large*> **Most general hypothesis:** <*pink,∗,∗,∗,∗,∗*>
3	 <*pink,triangle,small,* *yellow,square,small*>	+	**Most specific hypothesis:** <*pink,triangle,∗* *yellow,square,∗*> **Most general hypothesis:** <*pink,∗,∗,∗,∗,∗*>

Example	Version Space Predictions	Total	Prediction
 <*pink,triangle,large* *yellow,square,small*>	<*pink,triangle,∗* *yellow,square,∗*>→ + . . . <*pink,∗,∗,∗,∗,∗*>→ +	+: 8 -: 0	+
 <*pink,circle,large* *yellow,circle,small*>	<*pink,triangle,∗* *yellow,square,∗*>→ - . . . <*pink,∗,∗,∗,∗,∗*>→ +	+: 2 -: 6	-
 <*pink,triangle,small* *green,square,small*>	<*pink,triangle,∗* *yellow,square,∗*>→ - . . . <*pink,∗,∗,∗,∗,∗*>→ -	+: 4 -: 4	?

Teacher Actions: { “This is a <concept>.” | “This is *not* a <concept>.”| “Is this a <concept>?” }

Active Learning Gets Better Coverage

Mode	Final F_1-scores	Subjects who achieved $\|V\| = 1$	Number of Steps to reach $\|V\| = 1$
SL	77.81%	6/24 (25%)	M = 10.67, SD = 4.59
AL	100.00%	24/24 (100%)	M = 8.29, SD = 2.29
MI	98.61%	23/24 (96%)	M = 8.35, SD = 1.56
AQ	97.50%	20/24 (83%)	M = 9.30, SD = 3.53

Humans have a tough time keeping track of their teaching progress, even for small instance spaces.

Subjective Measures

Mode	Intelligence (7=best)	Enjoyability (7=best)
SL	M = 4.08, SD = 1.64	M = 4.92, SD = 1.79
AL	M = 5.54, SD = 1.53	M = 5.46, SD = 1.77
MI	M = 5.12, SD = 1.57	M = 5.46, SD = 1.50
AQ	M = 5.75, SD = 1.29	M = 6.04, SD = 1.16

Active Learning modes are perceived as both more intelligent and more enjoyable to interact with

People preferred control over triggering the robot's Active Learning mechanism.