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

Training
- Feature Extraction
- Keyframe Clustering (Usually KNN)
- Point to Keyframe Classifier (Usually SVM)
- HMM trained on keyframe sequences

Testing
- Feature Extraction
- Keyframe Classification
- HMM Likelihood Evaluation (Forward Algorithm)
- Choose model with greatest posterior probability
Activity **Generation** Workflow

**Training**
- Feature Extraction
- Keyframe Clustering (Usually KNN)
- Point to Keyframe Classifier (Usually SVM)
- Model trained on keyframe sequences

**Sampling**
- Model Selection
- Keyframe Sampling
- Motion Planning
- Motor execution
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

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

• 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.

Figure 3: An algorithm for extracting via-points.

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

Keyframe demonstrations

Demo 1

a b c d
x y y y y z

Clustering

Cluster 1: a b c d
Cluster 2: x y y y y z

Demo 2

x y z

Cluster 1: a x
Cluster 2: b y c
Cluster 3: d z
Implementation on PR2

Learning Kitchen Tasks from Hybrid Demonstrations
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?
<table>
<thead>
<tr>
<th>Skill Learning Wishlist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recognize repeated instances of skills and generalize them to new settings.</td>
</tr>
<tr>
<td>Segment data without a priori knowledge of task structure.</td>
</tr>
<tr>
<td>Identify broad, general classes of skills (eg., manipulations, gestures, goal-based actions.)</td>
</tr>
<tr>
<td>Skill policies should have a flexible encoding such that they can be improved over time.</td>
</tr>
</tbody>
</table>
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]

\[ B | B_0 \sim BP(1, B_0) \]
\[ X_i | B \sim BeP(B) \]
\[ \pi_{j(i)} | f_i, \gamma, \kappa \sim Dir([\gamma, ..., \gamma + \kappa, \gamma, ...] \otimes f_i) \]
\[ z_t^{(i)} \sim \pi_{z_{t-1}^{(i)}} \]
\[ y_t^{(i)} = \sum_{j=1}^{r} A_{j, z_t^{(i)}} y_{t-j}^{(i)} + e_t^{(i)}(z_t^{(i)}) \]

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

- 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

Incremental Semantically Grounded Learning from Demonstration
Abstraction is essential for solving complex problems

- Task and motion planning
- Multi-agent coordination
- Activity recognition
- Goal inference
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

PARADOX
play table and chair

3x
4x
64x
7x
16.5x
1x
Hierarchical Task Structure
IKEA Chair

Assemble Chair

Orient Rear Frame
- Get Frame
- Place Frame in Workspace

Attach Supports
- Attach Left Support
  - Get Peg
  - Place Peg(Light Frame)
  - Place Support(Light Frame)
  - Add Left Support HW
  - Get Bolt
  - Place Bolt(Light Rear Frame)
  - Screw Bolt(Light Rear Frame)
  - Get Nut
  - Place Nut(Light Support)
  - Get Bolt
  - Place Bolt(Light Rear Frame)

Attach Right Support
- Attach Right Support
  - Get Peg
  - Place Peg(Right Frame)
  - Place Support(Right Frame)
  - Add Right Support HW
  - Get Bolt
  - Place Bolt(Right Rear Frame)
  - Screw Bolt(Right Rear Frame)
  - Get Nut
  - Place Nut(Right Support)
  - Get Bolt
  - Place Bolt(Right Rear Frame)

Add Seat
- Get Seat
- Place Seat

Attach Front Frame
- Attach Front Frame
  - Get Peg
  - Place Peg(Left Support)
  - Get Peg
  - Place Peg(Right Support)
  - Get Peg
  - Place Peg(Right Rear Frame)
  - Get Peg
  - Place Peg(Right Rear Frame)
Hierarchical Task Structure
IKEA Chair

Assemble Chair

Orient Rear Frame
- Get Frame
- Place Frame in Workspace

Attach Supports
- Get Frame
- Place in Workspace
- Attach Supports (Left Frame)
  - Get Peg
  - Place Peg (Left Frame)
  - Get Support
  - Place Support (Left Frame)
  - Add Left Support
  - Get Peg
  - Place Peg (Right Frame)
  - Get Support
  - Place Support (Right Frame)
  - Add Right Support
  - Screw Bolt (Left Rear Frame)
  - Get Nut
  - Place Nut (Left Support)
  - Get Bolt
  - Screw Bolt (Right Rear Frame)

Add Left Support
- Get Peg
- Place Peg (Left Support)
- Get Nut
- Place Nut (Left Support)
- Get Bolt
- Screw Bolt (Left Rear Frame)

Attach Right Support
- Get Peg
- Place Peg (Right Support)
- Get Nut
- Place Nut (Right Support)
- Get Bolt
- Screw Bolt (Right Rear Frame)

Add Right Support
- Get Peg
- Place Peg (Right Support)
- Get Nut
- Place Nut (Right Support)
- Get Bolt
- Screw Bolt (Right Rear Frame)

Attach Front Frame
- Get Peg
- Place Peg (Left Support)
- Get Peg
- Place Peg (Right Support)
- Get Front Frame
- Place Front Frame

Attach Seat
- Get Seat
- Place Seat

Mount Front Frame
- Get Front Frame
- Place Front Frame (Supports)

Subtask abstractions allow for MDP factoring
SMDP of “Attach Front Frame” Subtask

Hierarchical View

State | Action
--- | ---
Front frame attached | Get R.Peg
| Have L.Peg
| Place L.Peg
| Get R.Peg
| Place R.Peg
| Get L.Peg
| Place L.Peg
| Place R.Peg
| Placed Pegs
| Have Frame
| Place Frame
| Get Frame

Attach Front Frame

- Place Pegs
- Get Front Frame
- Place Pegs
- Place Pegs
- Mount

2x
No clear structural cues!

Hierarchical View

Attach Front Frame

- Place Pegs
- Mount
  - Place left peg
  - Place right peg
  - Get Front Frame
  - Place peg (left support)
  - Get peg
  - Place peg (right support)

State
- Front frame attached
  - Have Frame
  - Placed Pegs
  - Place Frame

Action
- Get L.Peg
- Get R.Peg
- Place L.Peg
- Place R.Peg
- Get L.Peg
- Place L.Peg
- Placed L.Peg
- Get R.Peg
- Placed R.Peg
- Have L.Peg
- Have R.Peg
- Placed Pegs
- Get Frame
SMDP-Conjugate of “Attach Front Frame” Subtask

Conjugate Graph Transform Algorithm

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

1. $C \leftarrow$ empty Conjugate Task Graph $\{W,F\}$;
2. $\text{origin} \_ \text{vertex} \leftarrow$ new empty vertex;
3. $\text{terminal} \_ \text{vertex} \leftarrow$ new empty vertex;
4. Add $\text{origin} \_ \text{vertex}$ and $\text{terminal} \_ \text{vertex}$ to $W$;
5. $\text{foreach}$ unique $a \in \{\text{action}(e) \mid e \in E\}$ do
   6. Add vertex $\{\text{action} = a\}$ to $W$;
7. $\text{foreach}$ edge $e \in E$ do
   8. If $\text{source}(e) \in \text{initiation} \_ \text{states}(G)$ then
      9. Add $\{\text{from: origin} \_ \text{vertex}, \text{to: } [v \in W \wedge \text{action}(v) = \text{action}(e)], \text{prerequisites: } \emptyset\}$ to $F$;
   10. If $\text{dest}(e) \in \text{termination} \_ \text{states}(G)$ then
       11. Add $\{\text{from: } [v \in W \wedge \text{action}(v) = \text{action}(e)], \text{to: terminal} \_ \text{vertex}, \text{prerequisites: } \text{environment} \_ \text{state(dest(e))}\}$ to $F$;
   12. else
       13. $\text{foreach}$ edge $f \in \text{outbound} \_ \text{edges(dest}(e))$ do
           14. Add $\{\text{from: vertex } v \in W \wedge \text{action}(v) = \text{action}(e), \text{to: } [u \in W \wedge \text{action}(u) = \text{action}(f)], \text{prerequisites: } \text{environment} \_ \text{state(to(e))}\}$
       15. $\text{foreach}$ $a, b \in F \ni a \neq b$ do
           16. If $\text{from}(a) = \text{from}(b) \text{ and } \text{to}(a) = \text{to}(b) \text{ and } \text{prerequisites}(a) \subseteq \text{prerequisites}(b)$ then delete $b$;
Building Hierarchical Structure

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

Building a constraint-based hierarchy

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Goal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get L.Peg</td>
<td>Place L.Peg</td>
<td>{Get L.Peg} o (Place L.Peg) o (Get L.Peg)</td>
</tr>
<tr>
<td>Get R.Peg</td>
<td>Place R.Peg</td>
<td>{Get R.Peg} o (Place R.Peg) o (Get R.Peg)</td>
</tr>
<tr>
<td>Get Frame</td>
<td>Place Frame</td>
<td>{Get Frame} o (Place Frame) o (Get Frame)</td>
</tr>
</tbody>
</table>

Step 0: Start Algorithm
### Building Hierarchical Structure

#### Task Graph to HTN Transform

<table>
<thead>
<tr>
<th>Input: SMDP-like Graph $G$</th>
<th>Output: HTN $H$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H \leftarrow \text{Conjugate-Graph-Transform}(G)$</td>
<td>$H$</td>
</tr>
<tr>
<td>while $</td>
<td>H_{vertices}</td>
</tr>
<tr>
<td>$h_{size} \leftarrow</td>
<td>H_{vertices}</td>
</tr>
<tr>
<td>foreach maximal clique $c = {V, E} \in H$ do</td>
<td>$V, E$</td>
</tr>
<tr>
<td>Compact ${V \in c}$ into single metanode $m$</td>
<td>$V, E$</td>
</tr>
<tr>
<td>foreach maximal chain $c = {V, E} \in H$ do</td>
<td>$V, E$</td>
</tr>
<tr>
<td>Compact ${V \in c}$ into single metanode $m$</td>
<td>$V, E$</td>
</tr>
<tr>
<td>if $h_{size} ==</td>
<td>H_{vertices}</td>
</tr>
</tbody>
</table>
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).
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

Internal nodes are completely connected

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

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

Building a constraint-based hierarchy

Start

State

Action

Goal

Task Graph to HTN Transform

Input: SMDP-like Graph G
Output: HTN H
1. $H \leftarrow$ Conjugate-Graph-Transform($G$)
2. while $|H_{vertices}| > 1$ do
3. \hspace{1cm} $h_{size} \leftarrow |H_{vertices}|$
4. \hspace{1.25cm} foreach maximal clique $c = \{V, E\} \in H$ do
5. \hspace{3cm} Compact $\{V \in c\}$ into single metanode $m$
6. \hspace{1.25cm} foreach maximal chain $c = \{V, E\} \in H$ do
7. \hspace{4cm} Compact $\{V \in c\}$ into single metanode $m$
8. \hspace{1cm} if $h_{size} = |H_{vertices}|$ then break;

Step 0: Start Algorithm
Building Hierarchical Structure

Task Hierarchy

Step 1: Find Cliques (0)

Start

{ }

Get L.Peg

{Get L.Peg} o {Get R.Peg}

Place L.Peg

{Place L.Peg} o {Get L.Peg}

Get R.Peg

{Get R.Peg}

Place R.Peg

{Place R. Peg} o {Get R. Peg} o {Get L.Peg} o {Place L. Peg} o {Get L.Peg}

Get Frame

{Get Frame} o {Get Frame} o {Get L.Peg} o {Get R. Peg} o {Place R. Peg} o {Get L. Peg} o {Place L. Peg} o {Get L.Peg} o {Get L.Peg} o {Place R. Peg} o {Get R. Peg}

Place Frame

{Place Frame} o {Place Frame} o {Get Frame} o {Get L.Peg} o {Get R. Peg} o {Place R. Peg} o {Get L.Peg} o {Place L.Peg} o {Get L.Peg} o {Get L.Peg} o {Place R. Peg} o {Get R. Peg}
Building Hierarchical Structure

Step 2: Find Chains (3)

Get L.Peg → Place L.Peg
Get R.Peg → Place R.Peg
Get Frame → Place Frame

Get L.Peg → Place L.Peg → Place R.Peg → Get R.Peg

State
Action
Goal

Task Hierarchy
Building Hierarchical Structure

Task Hierarchy

Step 1: Find Cliques (1)

Start

State

Action

Goal

Get Frame

Place Frame

{Place L. Peg} o
{Get L.Peg} o
{Place R. Peg} o
{Get R. Peg}

{Place Frame} o
{Get Frame} o
{Place L. Peg} o
{Get L.Peg} o
{Place R. Peg} o
{Get R. Peg}
Building Hierarchical Structure

Start → { } → Get Frame → Place Frame

State

Action

Goal

Task Hierarchy

Step 2: Find Chains (1)
Building Hierarchical Structure

Start

State

Action

Goal

Get Frame

Place Frame

Goal

Step 3: Single node graph!

Task Hierarchy

{ } o

{Place Frame} o

{Get Frame} o

{Place L. Peg} o

{Get L.Peg} o

{Place R. Peg} o

{Get R. Peg}

Attach Frame

Get Frame

Place Frame

Get L.Peg

Place L.Peg

Get R.Peg

Place R.Peg

{ }
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)

4 Demonstrations of “Activity X”
Related Work

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

(Perez D’Arpino ICRA15)

Model each timestep
Related Work

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

(Perez D’Arpino ICRA15)

Model each timestep with mean-centered Gaussian
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

Training

Feature Extraction → Keyframe Clustering (Usually KNN) → Point to Keyframe Classifier (Usually SVM) → HMM trained on keyframe sequences

Testing

Feature Extraction → Keyframe Classification → HMM Likelihood Evaluation (Forward Algorithm) → Choose model with greatest posterior probability

- 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,”
New Activity Recognition Approaches?

End-to-end Network

End-to-end Network
In real deployments, humans need to be able to understand robot decisions.

**This is your machine learning system?**

Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.

**What if the answers are wrong?**

Just stir the pile until they start looking right.
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

[Feature Extraction] [Temporal Segmentation] [Feature-wise Segmentation] [Local Model Training] [Ensemble Weight Learning]
Activity Model Training Pipeline

1. Feature Extraction
2. Temporal Segmentation
3. Feature-wise Segmentation
4. Local Model Training
5. Ensemble Weight Learning
Activity Model Training Pipeline

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

Displacement vs. Time

0% 100%
Activity Model Training Pipeline

Two Temporal Segment Parameters: Width and Stride

Displacement

Time

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

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

Displacement

{Width=0.2, Stride=1.}

1 2 3 4 5

0% Time 100%
Activity Model Training Pipeline

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

Displacement

Time

0% 1 3 5 7 9 2 4 6 8

{Width=0.2, Stride=.5}
Activity Model Training Pipeline

Object Map:
Dictionary that maps IDs to sets of column indices
E.g., {“Hands”: [0,1,2,5,6,7]}

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0.04</th>
<th>-0.57</th>
<th>-0.54</th>
<th>-0.74</th>
<th>-0.22</th>
<th>-0.75</th>
<th>-0.96</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.04</td>
<td>1</td>
<td>0.45</td>
<td>0.43</td>
<td>-0.03</td>
<td>-0.4</td>
<td>-0.44</td>
<td>-0.28</td>
<td>0.16</td>
</tr>
<tr>
<td>-0.07</td>
<td>0.45</td>
<td>1</td>
<td>0.39</td>
<td>0.49</td>
<td>0.2</td>
<td>-0.16</td>
<td>0.15</td>
<td>0.35</td>
</tr>
<tr>
<td>-0.4</td>
<td>0.41</td>
<td>0.39</td>
<td>1</td>
<td>0.06</td>
<td>0.2</td>
<td>0.11</td>
<td>0.13</td>
<td>0.38</td>
</tr>
<tr>
<td>-0.51</td>
<td>-0.03</td>
<td>0.49</td>
<td>1</td>
<td>0.36</td>
<td>0.35</td>
<td>0.02</td>
<td>0.15</td>
<td>0.39</td>
</tr>
<tr>
<td>-0.74</td>
<td>-0.4</td>
<td>0.2</td>
<td>0.2</td>
<td>0.36</td>
<td>1</td>
<td>0.97</td>
<td>0.57</td>
<td>0.19</td>
</tr>
<tr>
<td>-0.22</td>
<td>-0.44</td>
<td>-0.16</td>
<td>0.11</td>
<td>0.62</td>
<td>0.37</td>
<td>1</td>
<td>0.11</td>
<td>-0.25</td>
</tr>
<tr>
<td>-0.95</td>
<td>-0.16</td>
<td>0.11</td>
<td>0.13</td>
<td>0.12</td>
<td>0.59</td>
<td>0.11</td>
<td>1</td>
<td>0.17</td>
</tr>
</tbody>
</table>
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

Within each temporal segment:

<table>
<thead>
<tr>
<th>Displacement</th>
<th>1.</th>
<th>0.04</th>
<th>-0.57</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.64</td>
<td>1.02</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>-0.57</td>
<td>1.45</td>
<td>1.</td>
</tr>
<tr>
<td></td>
<td>-0.4</td>
<td>0.11</td>
<td>0.39</td>
</tr>
<tr>
<td></td>
<td>-0.54</td>
<td>0.03</td>
<td>0.19</td>
</tr>
<tr>
<td></td>
<td>0.74</td>
<td>-0.4</td>
<td>0.2</td>
</tr>
<tr>
<td></td>
<td>-0.22</td>
<td>-0.44</td>
<td>-0.16</td>
</tr>
<tr>
<td></td>
<td>n/a</td>
<td>n/a</td>
<td>n/a</td>
</tr>
</tbody>
</table>
Activity Model Training Pipeline

Within each temporal-object segment:

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

Result: An activity classifier ensemble across objects and time!
Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment
Activity Model Training Pipeline

Need to find the most discriminative Object GMMs per time segment

Random Forest Classifier
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

![Diagram showing the pipeline with feature extraction, temporal segmentation, feature-wise segmentation, local model training, and ensemble weight learning.]
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)
Recognition Results: UTKinect-Action3D

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slama et al. (2015) [21]</td>
<td>88.5%</td>
</tr>
<tr>
<td>Chhrungoo et al. (2014) [18]</td>
<td>89.45%</td>
</tr>
<tr>
<td>Xia et al. (2012) [11]</td>
<td>90.9%</td>
</tr>
<tr>
<td>Wang et al. (2015) [24]</td>
<td>90.9%</td>
</tr>
<tr>
<td>Devanne et al. (2013) [20]</td>
<td>91.5%</td>
</tr>
<tr>
<td><strong>RAPTOR (proposed method)</strong></td>
<td><strong>92.1%</strong></td>
</tr>
</tbody>
</table>

The table above shows the real-time UTKinect activity recognition accuracy for different classifiers compared to the proposed RAPTOR method.
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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTKinect</td>
<td>79.4%</td>
<td>83.1%</td>
<td>84.7%</td>
<td>92.1%</td>
</tr>
<tr>
<td>Static-Reach</td>
<td>69.7%</td>
<td>77.2%</td>
<td>93.8%</td>
<td>97.5%</td>
</tr>
<tr>
<td>Dynamic-AutoFA</td>
<td>91.7%</td>
<td>88.1%</td>
<td>90.5%</td>
<td>92.0%</td>
</tr>
</tbody>
</table>
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

Which label belongs to this interval?

\[
\begin{align*}
\text{\{-121.914\}} \\
\text{\{-66.29\}} \\
\text{\{-0.462\}} \\
\text{\{-3.52\}}
\end{align*}
\]
Classification vs. Segmentation

What are the right intervals?
Which intervals should get labels?
Which labels should be where?
A Naïve Changepoint Detection Approach

Scenario

**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 = 1458000sec
~16.88 days

Classifiers must be ideal (sensitive to trajectory length, non-overlapping, comparable tolerance to noise, etc.)
Particle Filtering for Changepoint Detection

- At each time step $t$:
  - Create new particles for all eligible classes
    - $start\_time = t - \text{minimum\_class\_duration}$
    - $prev\_interval = \text{particle with highest MAP estimate in best}[start\_time]$
  - Evaluate existing particles’ likelihoods over the interval $[p.start\_time, t]$ and store as (likelihood, p) tuples in $particle\_maps$
  - Terminate stale particles

$particle\_maps[]$ – Sorted (MAP, particle) tuples for each timestep
Particle Filtering for Changepoint Detection

- At each time step $t$:
  - Create new particles for all eligible classes
    - $start\_time = t - minimum\_class\_duration$
    - $prev\_interval = \text{particle with highest MAP estimate in best[start\_time]}$
  - Evaluate existing particles’ likelihoods over the interval $[p.start\_time, t]$ and store as (likelihood, p) tuples in $particle\_maps$
  - Terminate stale particles

particle\_maps[] – Sorted (MAP, particle) tuples for each timestep
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
  - $start\_time = t - \text{minimum\_class\_duration}$
  - $prev\_interval = \text{particle with highest MAP estimate in best}[start\_time]$
- 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 \( f = \text{final frame index} \)

• While \( f > 0 \) and \( \text{particle\_maps}[f] \neq \text{None} \):
  • Take best (MAP, particle) at particle\_maps index \( f \)
  • Annotate segment \([\text{shape\_start}, f]\) with \text{shape\_class}
  • Set \( f = \text{particle\.start_time} \)
Particle Filtering for Changepoint Detection

To extract the most likely segmentation:

- Set \( f = \text{final frame index} \)
- While \( f > 0 \) and \( \text{particle\_maps}[f] \neq \text{None} \):
  - Take best (MAP, particle) at \( \text{particle\_maps}[f] \)
  - Annotate segment \([\text{shape\_start}, f]\) with \( \text{shape\_class} \)
  - Set \( f = \text{particle.start\_time} \)
To extract the most likely segmentation:

- Set $f = \text{final frame index}$
- While $f > 0$ and $\text{particle_maps}[f] \neq \text{None}$:
  - Take best (MAP, particle) at particle_maps index $f$
  - Annotate segment $[\text{shape_start}, f]$ with $\text{shape_class}$
  - Set $f = \text{particle.start_time}$
To extract the most likely segmentation:

- Set \( f = \text{final frame index} \)
- While \( f > 0 \) and \( \text{particle\_maps}[f] \neq \text{None} \):
  - Take best (MAP, particle) at particle\_maps index \( f \)
  - Annotate segment \([\text{shape\_start}, f]\) with \(\text{shape\_class}\)
  - Set \( f = \text{particle\_start\_time} \)
To extract the most likely segmentation:

- Set $f = \text{final frame index}$
- While $f > 0$ and particle_maps[f] != None:
  - Take best (MAP, particle) at particle_maps index $f$
  - Annotate segment $[\text{shape\_start}, f]$ with $\text{shape\_class}$
  - Set $f = \text{particle\_start\_time}$
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

Attractive
Student

Repulsive
Peer

Thresholded
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

The Shutdown Problem

\[ P(sd)R_{sd} + (1 - P(sd))R \]

Non-Functional Behavior

Desired Behavior

Incorrigible Behavior

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 H observes the actual reward signal; R only knows a prior distribution on reward functions
    - R learns the reward parameters by observing H
Cooperative Inverse Reinforcement Learning

\[ \langle S, A, T, R, \gamma \rangle \]

Action sets for human and robot

Distribution over (parameterized) reward functions

\[ \langle S, \{ A^H, A^R \}, T, \{ R, \Theta, P_0 \}, \gamma \rangle \]

Both act to maximize

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

[Hadfield-Menell arXiv ‘16]

Cooperative Inverse Reinforcement Learning

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

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

Cooperative Inverse Reinforcement Learning

CIRL Properties

• The distribution over state sequences is determined by a pair of policies: $(\pi^H, \pi^R)$

• 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  Symbolic planning  Motion planning

Autonomously Generated Supportive Behaviors
The TAMP problem is represented by the tuple: \{A, O, C, s_0, s_G\}

\textbf{A} is a set of (lead) agents

\textbf{O} is a set of operators (unparameterized motor primitives)

\textbf{C} is a capabilities mapping function between agents and operators

\textbf{s_0} is the set of predicates \textit{precisely} specifying the start state

\textbf{s_G} is the set of predicates specifying the goal state
The SB-TAMP problem is represented as the tuple: \( \{ T, \Pi_T, a_s, C_s, s_c, P \} \)

- \( T \) is a TAMP problem
- \( \Pi_T \) is a set of symbolic plans for \( T \)
- \( a_s \) is a supportive agent
- \( C_s \) is a mapping function indicating operators from \( T \) usable by \( a_s \)
- \( s_c \) 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

Hypothetical Environment Generator

Current State

Goal Predicates

Current State

Lead Agent Planner Model

Initial State

Goal State

Support Agent Planner

Initial State

Goal State

Multi-agent Plan Evaluation

Policy Weighting Function

Support Policy

Policies
Supportive Behavior Pipeline
Plan Evaluation

Choose the support policy \((\xi \in \Xi)\) that minimizes the expected execution cost of the leader’s policy \((\pi \in \Pi)\) to solve the TAMP problem \(T\) from the current state \(s_c\)

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

\[
\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi \ast \text{cost} (T, \pi, \xi, s_c, \gamma)
\]
Plan Evaluation

Choose the support policy ($\xi \in \Xi$) that minimizes the expected execution cost of the leader’s policy ($\pi \in \Pi$) to solve the TAMP problem $T$ from the current state ($s_c$).

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

Weighting function makes a big difference!

$$\min_{\xi \in \Xi} \sum_{\pi \in \Pi_T} w_\pi \times \text{cost} (T, \pi, \xi, s_c, \gamma)$$
Weighting functions: Uniform, Greedy

\[ w_\pi = 1 \]

Consider all known solutions equivalently likely and important

\[ w_\pi = \begin{cases} 
1 & \text{duration}(T, \pi, \emptyset, s_0, f(x) = 1) = \min_{\text{duration}} \\
0 & \text{otherwise} 
\end{cases} \]

Only the best-known solution is worth planning against
Weighting functions: Uniform
Weighting functions: Optimality-Proportional

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

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

Plan Weight

Plan Duration : Best Known Plan Duration
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 \( \epsilon \) are treated normally, per \( f \).

Suboptimal plans are \textit{negatively weighted}, encouraging 
active mitigation behavior from the supportive robot.

\( \alpha < \frac{1}{\max_\pi w_\pi} \) is a normalization term to avoid harm due to plan overlap.
Weighting functions:
Error Mitigation
Effect of Supportive Behaviors

Task Completion Time

<table>
<thead>
<tr>
<th>Slow Leader</th>
<th>Fast Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>45</td>
</tr>
</tbody>
</table>
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?
I will terminate *assist aluminum pouring* when the world is in the blue region of state space:
Semantics for Policy Transfer
Semantics for Policy Transfer
I will terminate **assist aluminum pouring** when the world is in state:

```
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12.4827</td>
<td>15</td>
<td>12.4827</td>
</tr>
<tr>
<td>5.12893</td>
<td>7.125</td>
<td>8.51422</td>
</tr>
<tr>
<td>1.12419</td>
<td>1.12419</td>
<td>1.12419</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3.62242</td>
<td>-8.1219</td>
<td>3.62242</td>
</tr>
<tr>
<td>-40.241</td>
<td>-40</td>
<td>-40.241</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

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

| 12.4827 | 15 |
| 5.12893 | 7.125 |
| 1.12419 | 1.12419 |
| 0 | 0 |
| 0 | 0 |
| 1 | 0 |
| 1 | 1 |
| 3.62242 | -8.1219 |
| -40.241 | -40 |
| ... | ... |

State space is too obscure to directly articulate
Motivation

- Loads parts onto belt
- Inspects parts
- Advances parts
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?
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]

State Disambiguation
[Wang et al. 2016]

Coordination Graphs
[Kalech 2010]

Hierarchical Task Models
[Hayes et al. 2016]

Policy Dictation
[Johnson et al. 2006]

Collaborative Planning
[Milliez et al. 2016]
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.”
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)

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: \( \text{State} \rightarrow \{ \text{True, False} \} \)

- 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”)

\[
\begin{align*}
\text{on\_top}(A,B) & \\
\text{camera\_powered} & \\
\end{align*}
\]
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 $S_{\pi^a}$, Set of non-target states $S_{\pi^\ast \setminus a}$

1. $S_{\pi^a} \leftarrow \{\}$;
2. $S_{\pi^\ast \setminus a} \leftarrow \{\}$;
3. foreach $s \in V$ do
4.     $a \leftarrow$ most frequent action executed from $s$;
5.     if $a == a_t$ then $S_{\pi^a} \leftarrow S_{\pi^a} \cup s$;
6.     else $S_{\pi^\ast \setminus a} \leftarrow S_{\pi^\ast \setminus a} \cup s$;
7. return $S_{\pi^a}, S_{\pi^\ast \setminus a}$;
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.

1. $S_{\pi^a} \leftarrow \{\}$;
2. $S_{\pi \setminus a} \leftarrow \{\}$;
3. **foreach** $D \in \{1, \ldots, D_{const}\}$ **do**
   4. **foreach** $s \in \{v \in V \mid \text{distance}(v, s_p) \leq D\}$ **do**
      5. $a \leftarrow$ most frequent action executed from $s$;
      6. **if** $a == a_t$ **then** $S_{\pi^a} \leftarrow S_{\pi^a} \cup s$;
      7. **else** $S_{\pi \setminus a} \leftarrow S_{\pi \setminus a} \cup s$;

8. $\text{expected\_region} \leftarrow \text{describe}(G, S_{\pi^a}, S_{\pi \setminus a})$;
9. $\text{current\_region} \leftarrow \text{describe}(G, \{s_p\}, S_{\pi^a})$;
10. return $\text{diff(}\text{expected\_region, current\_region})$, $\text{expected\_region}$;
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 \( \text{cluster\_max} \)

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

1. \( S \leftarrow \text{dict}() \)
2. \( \text{descriptions} \leftarrow \text{dict}() \)
3. \( \text{DNF}\_\text{description} \leftarrow \text{convert\_to\_DNF\_formula}(d, C) \)
4. foreach \( s \in \{ v \in V \mid \text{test\_dnf}(v, \text{DNF\_description}) \text{ is True} \} \) do
5. \( S[\pi(s)] \leftarrow S[\pi(s)] \cup s \)
6. if \( |S| > \text{cluster\_max} \) then
7. \[ \text{return too\_many\_actions\_error} \]
8. foreach \( a \in S \) do
9. \( \_\text{descriptions}[a] \leftarrow \text{describe}(S[a]) \)
10. return \( \text{descriptions} \).
Given

Required

Policy model
Concept representations
Mapping from query to model
Mapping from model to response
Recall: Concept library provides dictionary of classifiers that cover state regions

\begin{align*}
\text{on\_top}(A,B) & \\
\text{camera\_powered} & 
\end{align*}
Using Concepts to Describe State Regions

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

<table>
<thead>
<tr>
<th>Concept Name</th>
<th>Concept Representation</th>
<th>Examples</th>
<th># of Instances</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOUSE</td>
<td>( \text{shape}<em>{top} = \text{triangle} \land \text{color}</em>{top} = \text{pink} \land \text{shape}_{bottom} = \text{square} )</td>
<td><img src="image1" alt="Example of a house" /></td>
<td>16</td>
</tr>
<tr>
<td>SNOWMAN</td>
<td>( \text{shape}<em>{top} = \text{circle} \land \text{size}</em>{top} = \text{small} \land \text{shape}_{bottom} = \text{circle} )</td>
<td><img src="image2" alt="Example of a snowman" /></td>
<td>28</td>
</tr>
<tr>
<td>ALIEN</td>
<td>( \text{shape}<em>{top} = \text{circle} \land \text{color}</em>{top} = \text{green} \land \text{color}_{bottom} = \text{green} )</td>
<td><img src="image3" alt="Example of a alien" /></td>
<td>10</td>
</tr>
<tr>
<td>ICE CREAM</td>
<td>( \text{shape}<em>{top} = \text{circle} \land \text{shape}</em>{bottom} = \text{triangle} \land \text{color}_{bottom} = \text{yellow} )</td>
<td><img src="image4" alt="Example of an ice cream" /></td>
<td>16</td>
</tr>
</tbody>
</table>
## Version Space Learning

<table>
<thead>
<tr>
<th>Step</th>
<th>Example</th>
<th>Label</th>
<th>Version Space</th>
</tr>
</thead>
</table>
| 1    | ![Triangle](triangle.png) ![Square](square.png) <pink,triangle,large yellow,square,large> | +     | **Most specific hypothesis:** <pink,triangle,large yellow,square,large>  
**Most general hypotheses:** <pink,*,*,*,*,*>  
<*,triangle,*,*,*,*>  
<*,*,*,*,*,*>  
<*,*,*,*,*,large> |
| 2    | ![Triangle](triangle.png) ![Square](square.png) <orange,triangle,large yellow,square,large> | -     | **Most specific hypothesis:** <pink,triangle,large yellow,square,large>  
**Most general hypothesis:** <pink,*,*,*,*,*> |
| 3    | ![Triangle](triangle.png) ![Square](square.png) <pink,triangle,small yellow,square,small> | +     | **Most specific hypothesis:** <pink,triangle,*> yellow,square,*>  
**Most general hypothesis:** <pink,*,*,*,*,*> |

<table>
<thead>
<tr>
<th>Example</th>
<th>Version Space Predictions</th>
<th>Total</th>
<th>Prediction</th>
</tr>
</thead>
</table>
| ![Triangle](triangle.png) ![Square](square.png) <pink,triangle,large yellow,square,large> | +: 8  
<pink,*,*,*,*,*>,+ | + |
| ![Triangle](triangle.png) ![Square](square.png) <pink,triangle,large yellow,square,small> | +: 2  
<pink,*,*,*,*,*>,+ | - |
| ![Triangle](triangle.png) ![Square](square.png) <pink,triangle,small yellow,square,small> | +: 4  
<pink,*,*,*,*,*>,? | ? |

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

<table>
<thead>
<tr>
<th>Mode</th>
<th>Intelligence (7=best)</th>
<th>Enjoyability (7=best)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SL</td>
<td>M = 4.08, SD = 1.64</td>
<td>M = 4.92, SD = 1.79</td>
</tr>
<tr>
<td>AL</td>
<td>M = 5.54, SD = 1.53</td>
<td>M = 5.46, SD = 1.77</td>
</tr>
<tr>
<td>MI</td>
<td>M = 5.12, SD = 1.57</td>
<td>M = 5.46, SD = 1.50</td>
</tr>
<tr>
<td>AQ</td>
<td>M = 5.75, SD = 1.29</td>
<td>M = 6.04, SD = 1.16</td>
</tr>
</tbody>
</table>

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.