Learning Structured Models for Safe Robot Control

Seminar at Microsoft Research Redmond
27th September 2019

Subramanian Ramamoorthy
School of Informatics, The University of Edinburgh
Edinburgh Centre for Robotics
Alan Turing Institute

EDINBURGH CENTRE FOR ROBOTICS

Autonomy in the OR

Level 1: Assistance	Level 2: Partial Automation	Level 3: Conditionally autonomous	Level 4: Highly automated	Level 5: Full autonomy
Basic "remote control" systems	Instrumented tools, still largely in remote control paradigm, e.g., Intuitive Surgical	?	?	Undesirable, perhaps

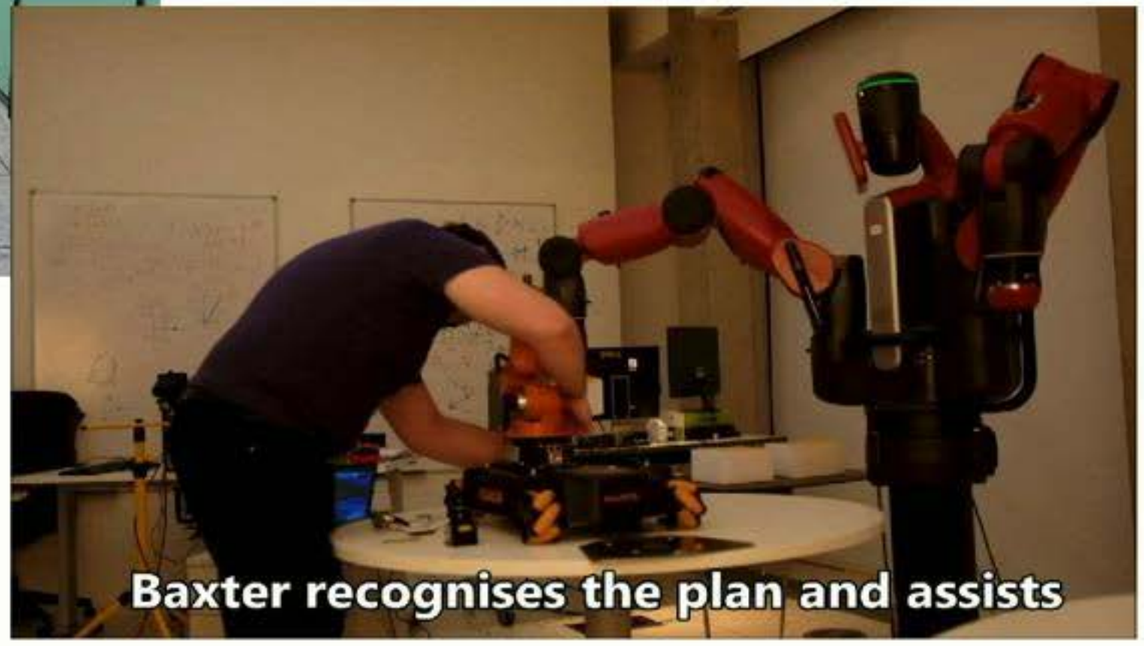
This is the status quo for "robot surgery" currently

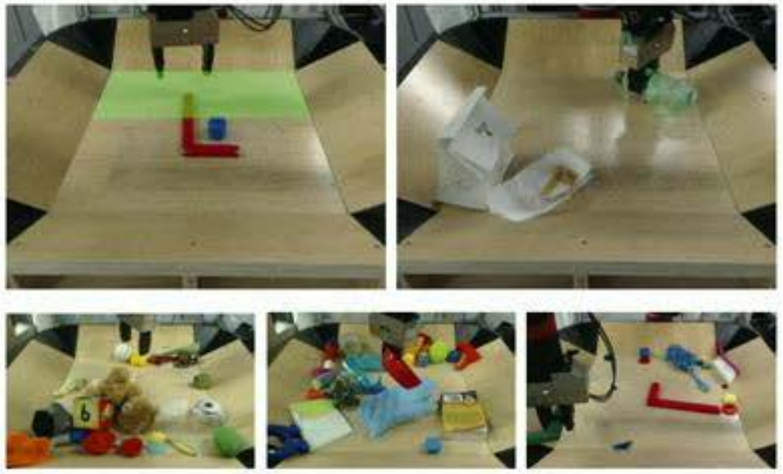
autonomy for assistance; skill prostheses

Many reasons why:

- [NHS] Severe shortage of junior staff (hence missed targets)
- [Global Health] Can a moderately skilled person do expert level work?
- Improved outcomes in terms of accuracy and time; lower lifecycle costs

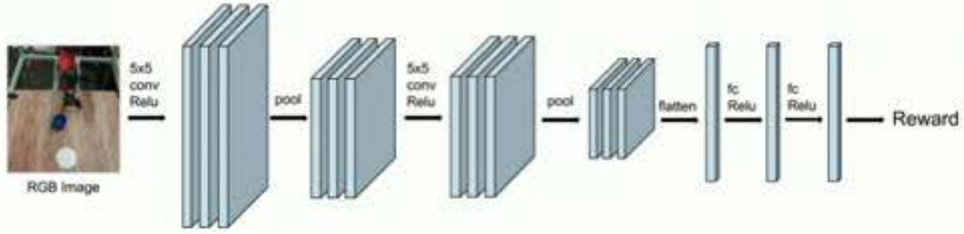
Project *sAlfer surgery*





Avi et al. arXiv:1904.05538

Singh et al. R:SS 2019



The seduction of end-to-end deep learning for robotics

- General purpose
- No domain specific knowledge
- Continually improves

How hard can it be?!

Useful Paradigm in Robotics: Learning from Demonstration

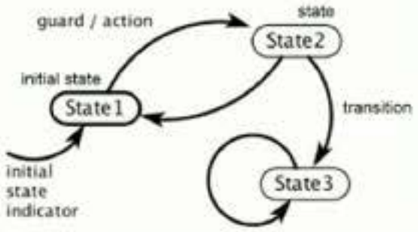
Question for LfD:
What ***Actually*** Defines the Task?

Motivation: What Did You Mean?!

[https://www.youtube.com/watch?v=yyse-BB-u9o]

Turing Project: Safe AI for Surgical Assistance

PIs: S Ramamoorthy (Informatics @ UoE), P Brennan (NHS Lothian, UoE Clinical Brain Sciences)



Specifications :
learnt from data +
extracted from codes of practice
(Amenable to reasoning about safety)

$$
\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi\, \mathcal{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi \wedge \\
 & & \quad \forall t'' \in [t, t'],\ (x, t'') \models \varphi\}
\end{array}
$$

Guaranteed
control synthesis
at Levels 3-4

Programming
by discussion &
model induction

What representations should we use? Do we really need models?!

Model Calibration: An Exploration into What and How We Want to Represent

[with T. Lopez-Guevara, K. Subr, CoRL 2017, RSS-18 & NeurIPS workshops]

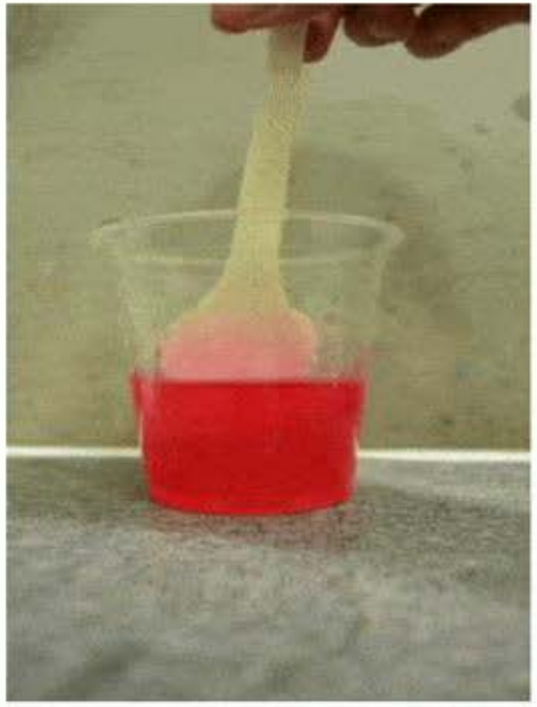
Will it splash?

A.

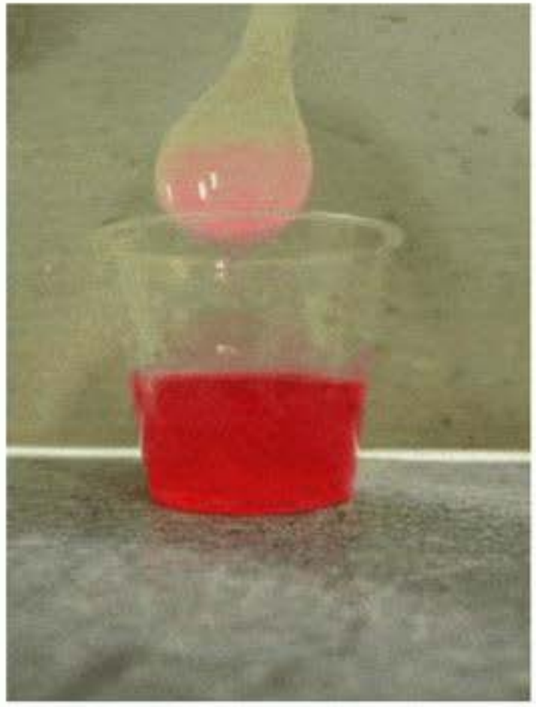
B.

What if we change the liquid?

Will it splash?



Model Intuition:
Thicker liquid, no splashes



Use simulation as Internal Model

Humans use **INTUITION**
to reason about the physical world

FAST BUT **APPROXIMATE**

?

@ Can Stock Photo / Paha_L

How to get properties of sim objects?

NVIDIA
Position Based Dynamics
FAST BUT APPROXIMATE

UNIFIED MODELS

assume shapes are known

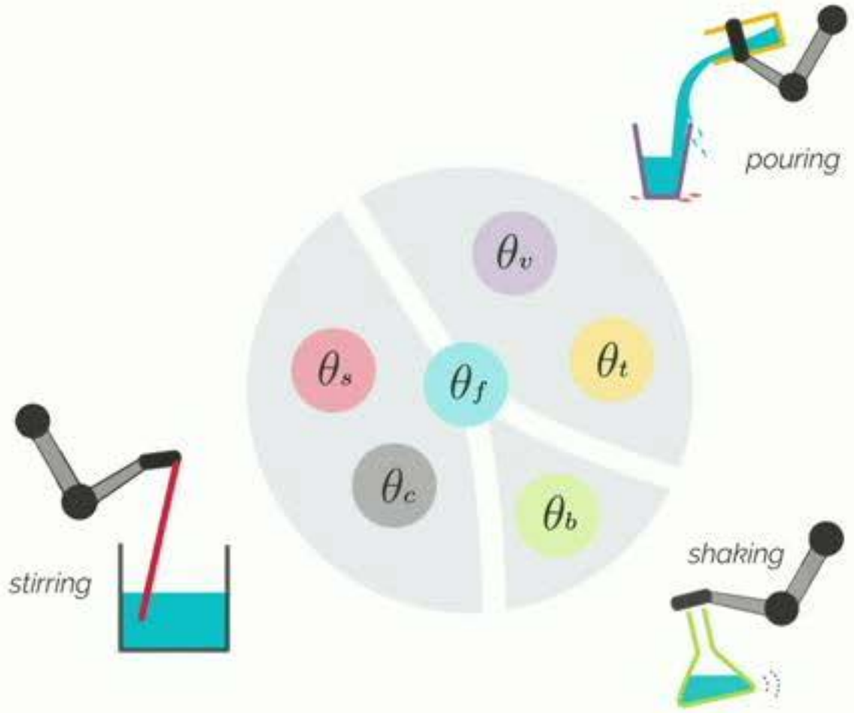
Use pouring to get estimates

θ_v

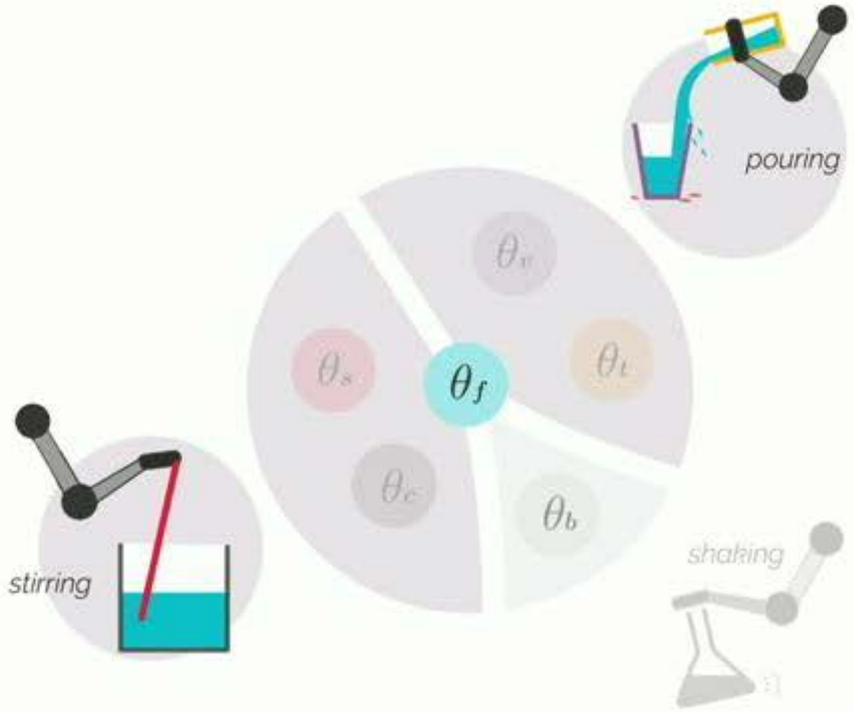
θ_t

θ_f

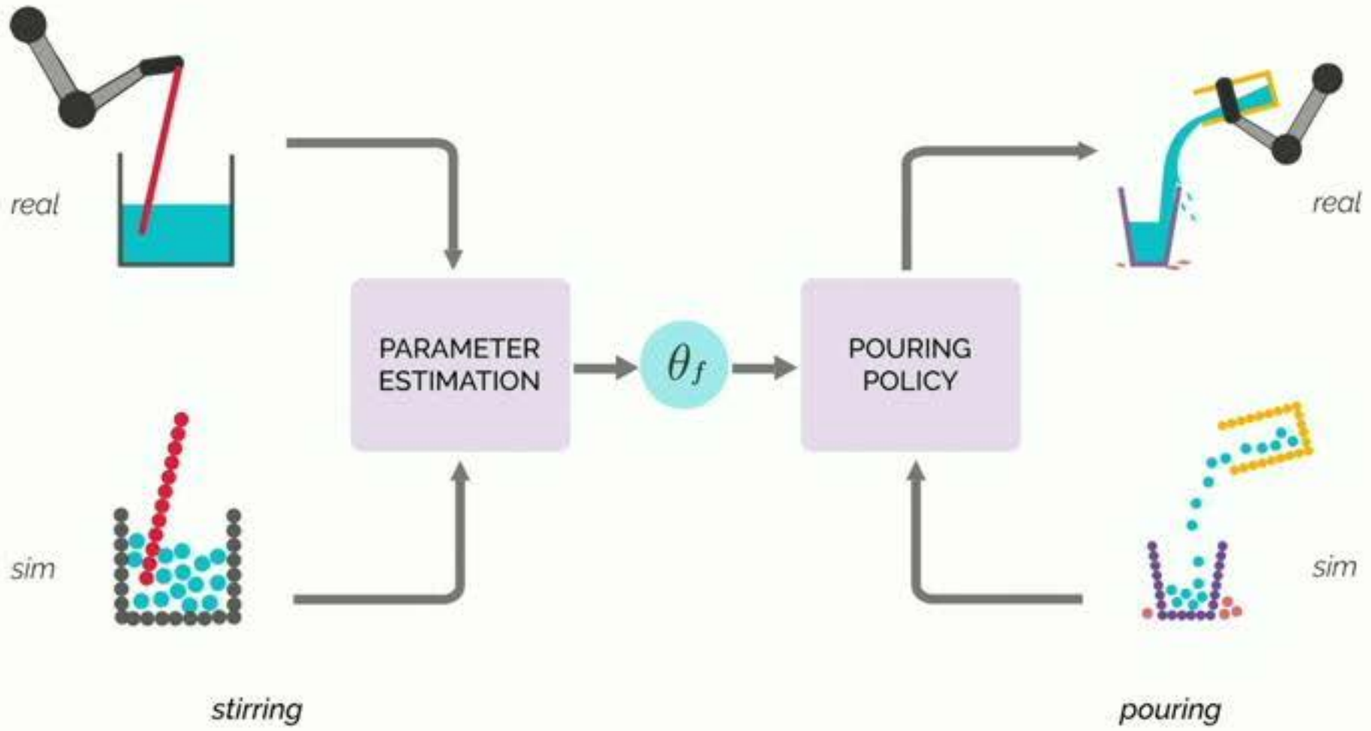
Observation: Different interactions give different information



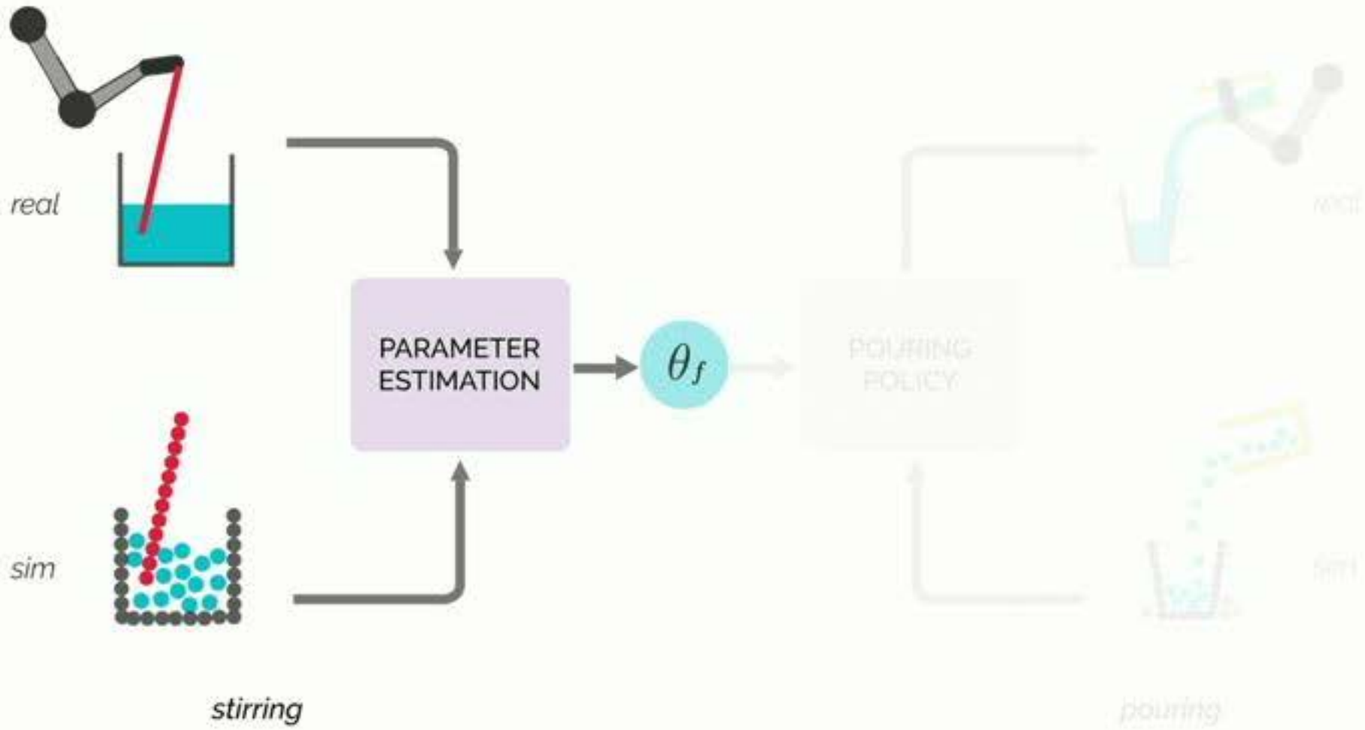
Observation: Different interactions give different information



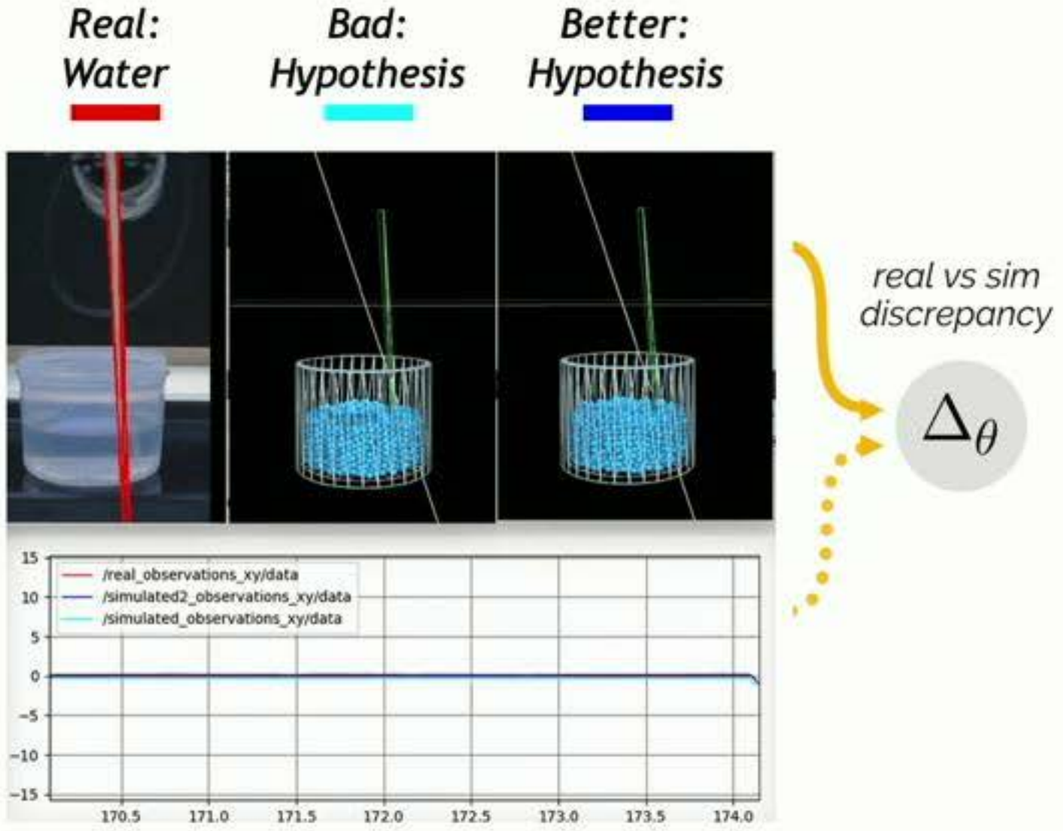
Estimates from stirring: good for pouring?



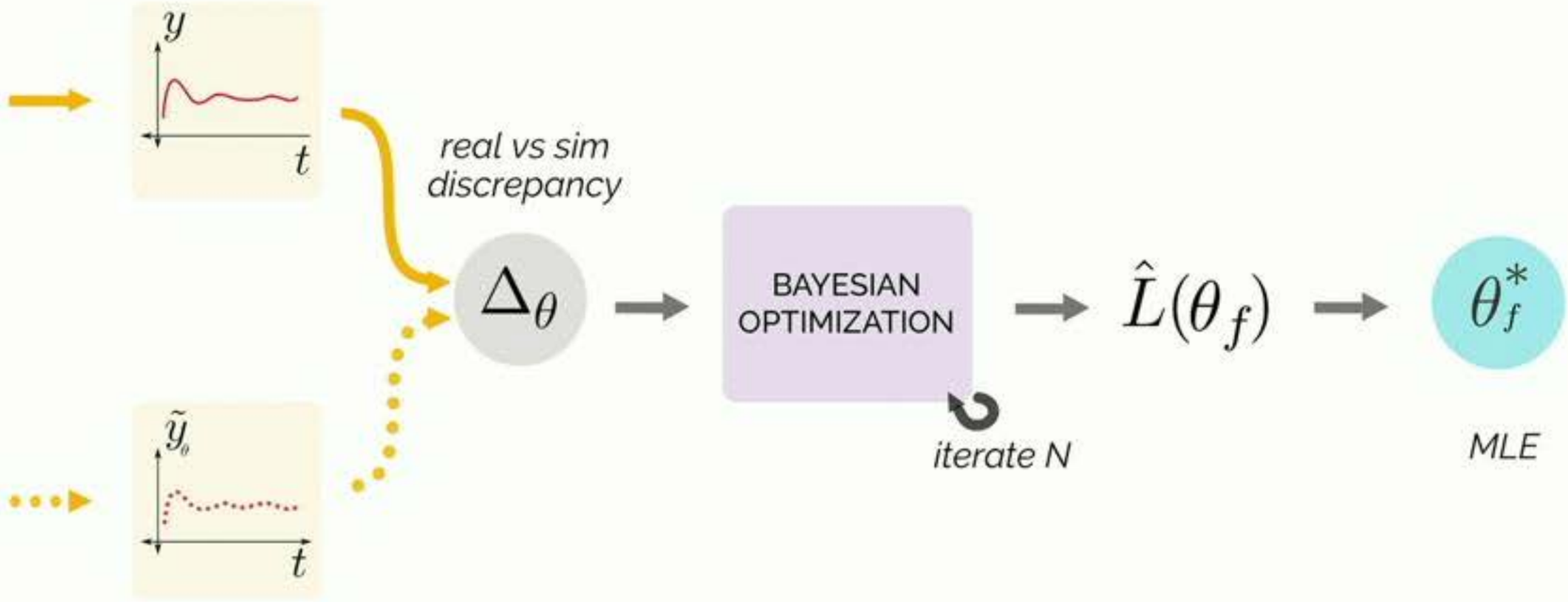
Estimates from stirring: good for pouring?



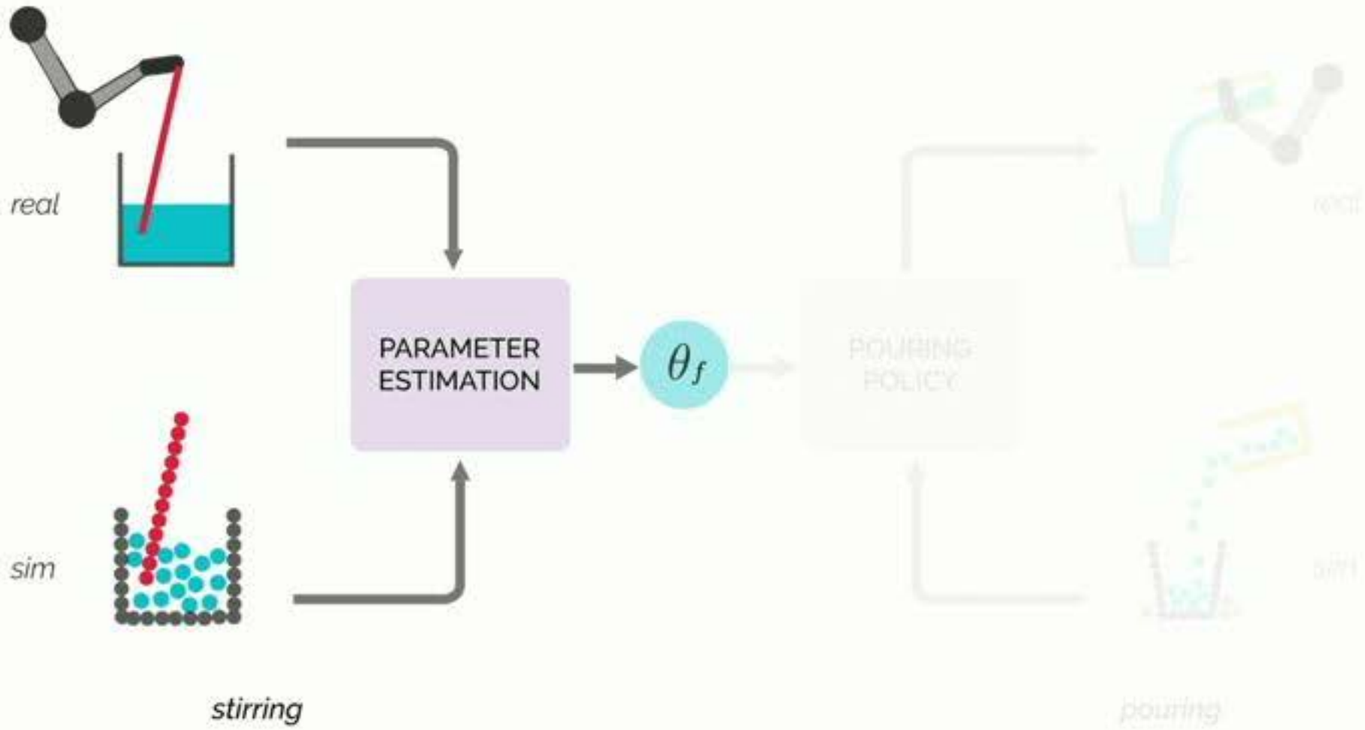
Synchronized stirring: Real vs Sim



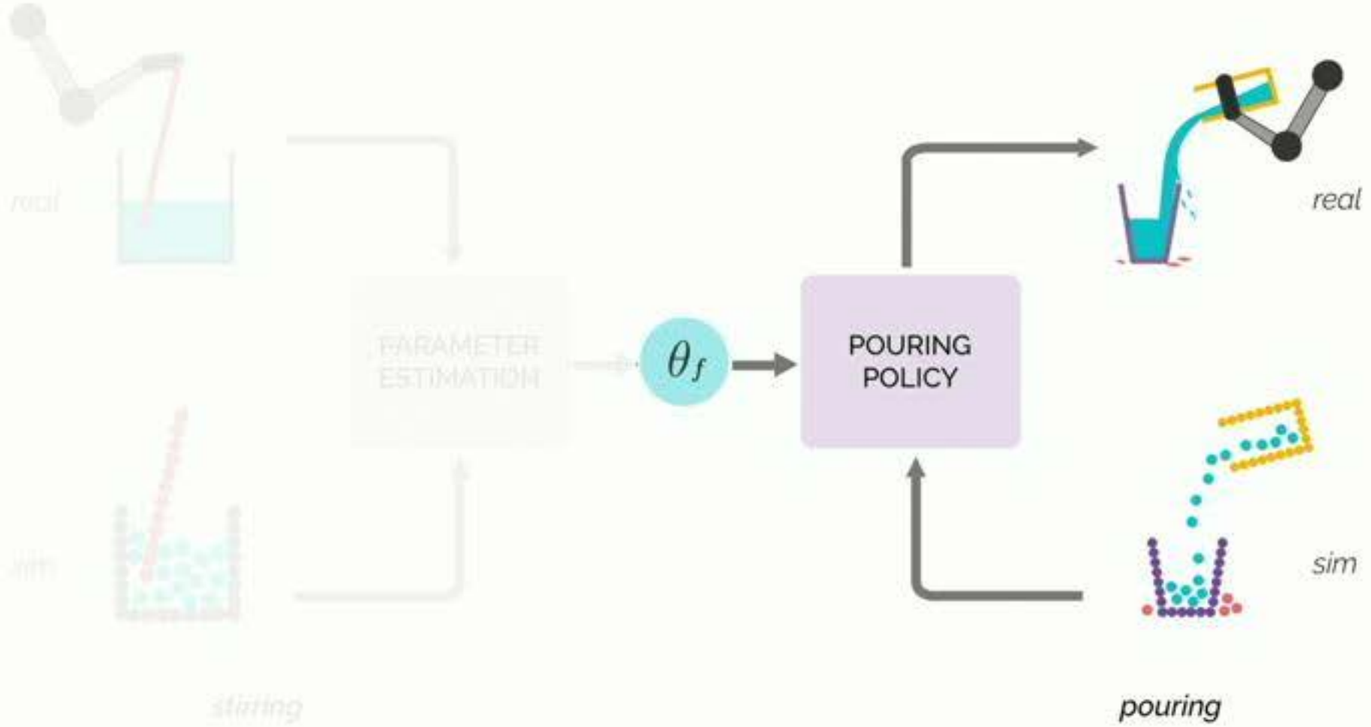
Use discrepancy to guide Optimization



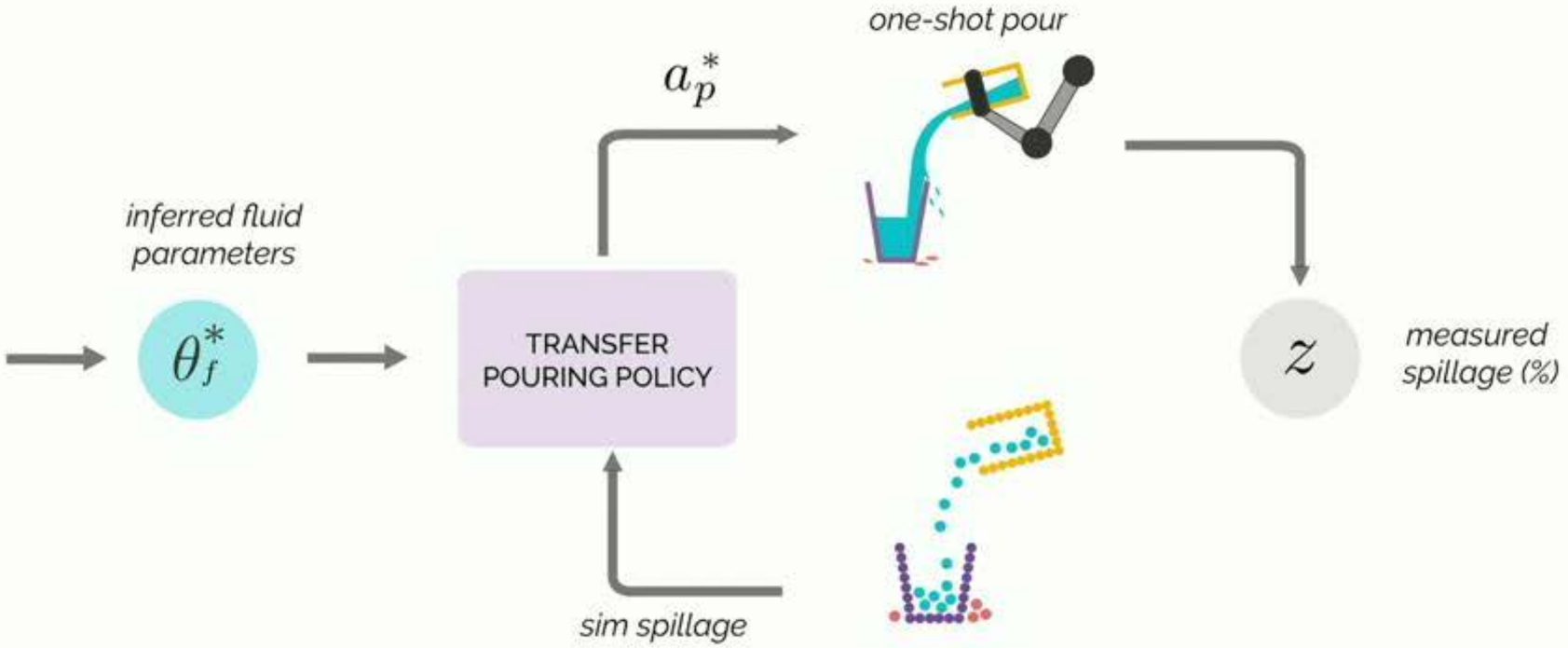
Estimates from stirring: good for pouring?



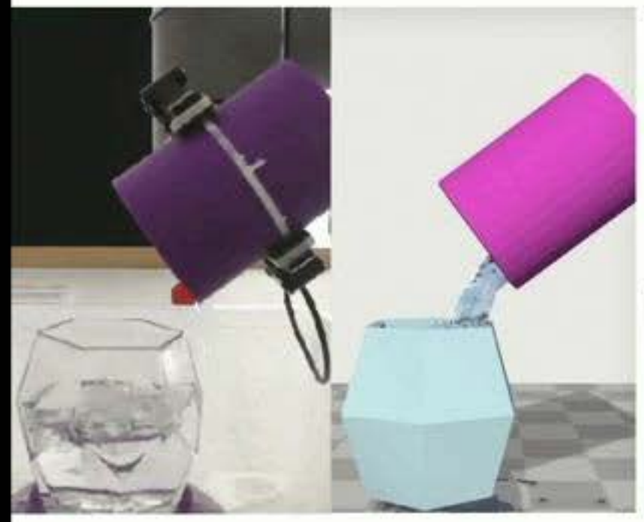
Estimates from stirring: good for pouring?

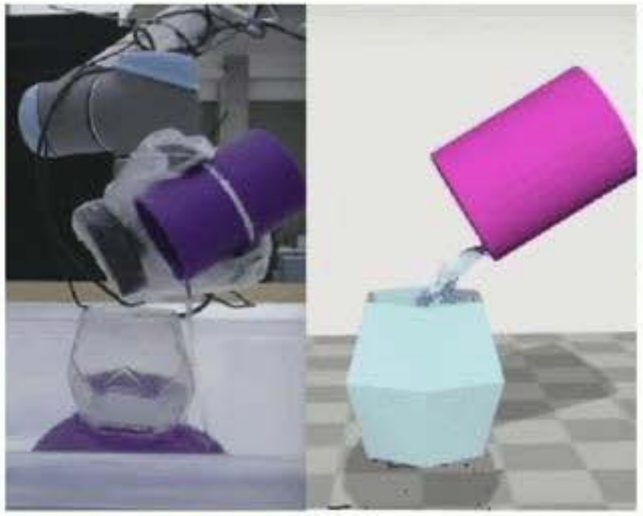


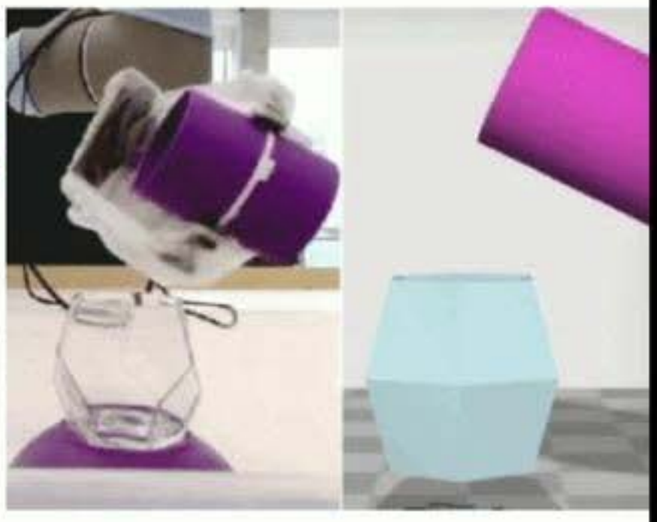
How good are the estimates?



Behaviour







- *water: (low viscosity)*
- *glycerin: (mid viscosity)*
- *gel: (high viscosity)*

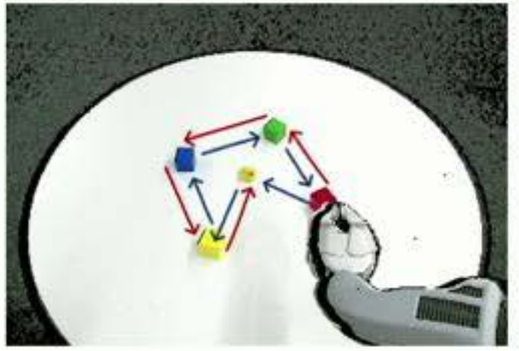
From Explanation to Synthesis: Compositional Program Induction and Switching Density Networks

[with M. Burke, S.V. Penkov, R:SS 2019, CoRL 2019]

Compositional Control is Key to Robotics

- End-to-end learning is effective, but controllers lack **flexibility** and **interpretability** (hence verifiability).

How do we:



- Change the inspection order?
- Add an object?
- Ask what happens next?
- Verify behavior properties?

- **Compositionality** provides **flexibility** and allows for **interpretability**.

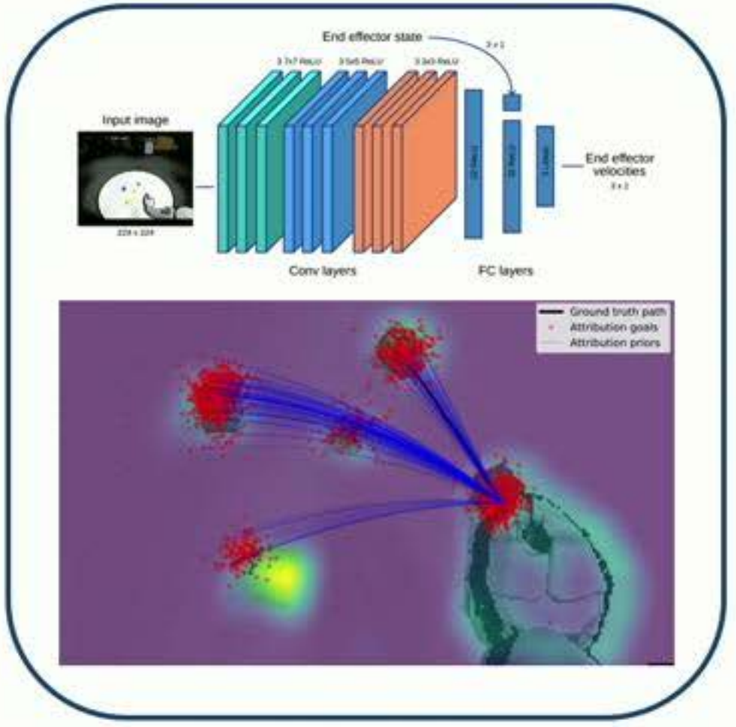
Burridge et al. 1999

Infer compositional controllers from an oracle model

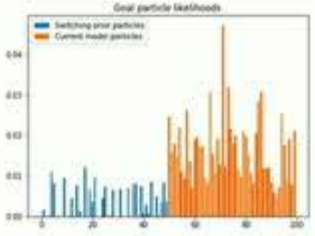
- 'Explain' end-to-end model demonstration using a generative model comprising **switching proportional control laws**.

$$\dot{\theta} \quad = \quad k_p^j(\theta - \theta_g^j)$$

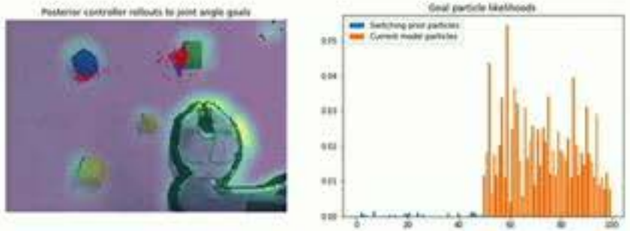
- Inference under this model is challenging so we use **model sensitivity priors** biased towards visual objects.

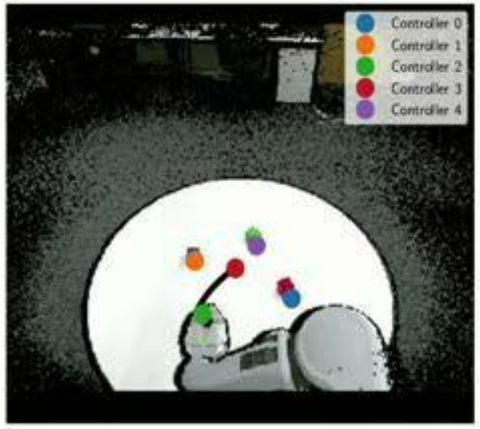


Infer controllers from visuomotor demonstration



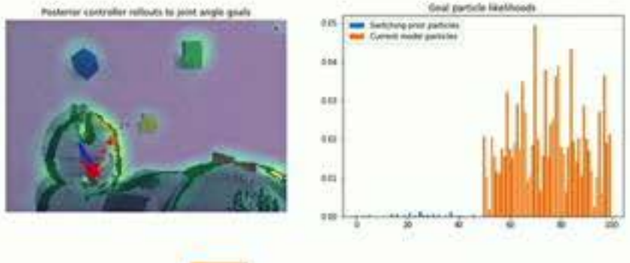
Infer controllers from visuomotor demonstration

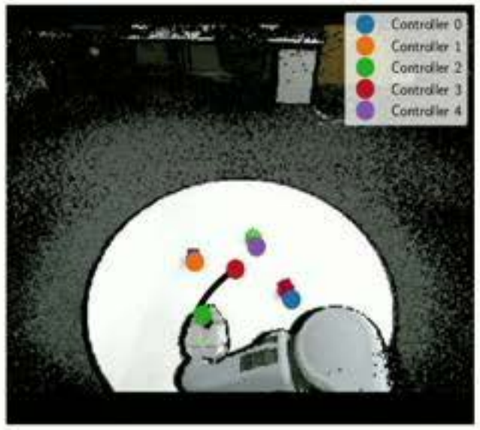




Cluster extracted controllers

Infer controllers from visuomotor demonstration

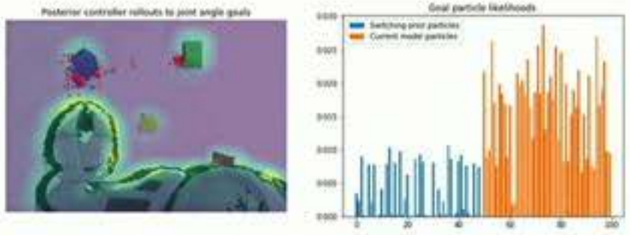


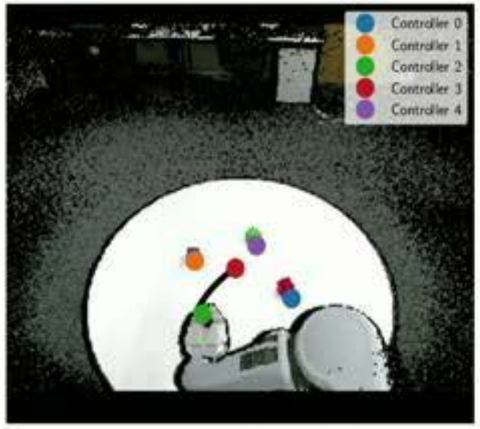


Cluster extracted controllers

Extract symbolic controller sequence

Infer controllers from visuomotor demonstration

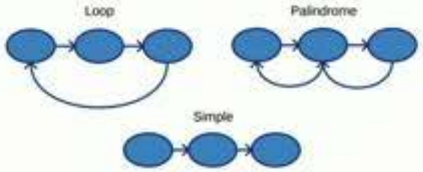




Cluster extracted controllers

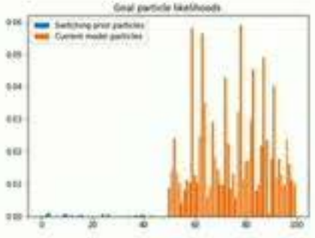
Extract symbolic controller sequence

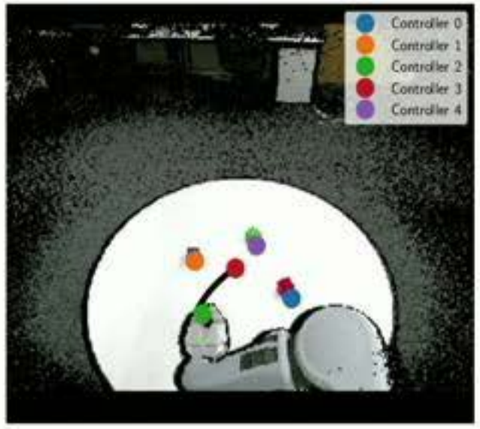
```
# Looping sequence
c_list = [2,1,4,0,3]
for j in range(6):

    # Palindromic sequence
    count = 0
    for k in range(len(c_list)*2-1):
        execute(c_list[count])
        if (k >= len(c_list)-1):
            count = count-1
        else:
            count = count+1

    # Simple sequence
    execute(3)

# Simple sequence
execute(2)
execute(1)
execute(4)
execute(0)
execute(3)
```



Extract program from symbolic sequence by searching for common structures.

Infer controllers from visuomotor demonstration

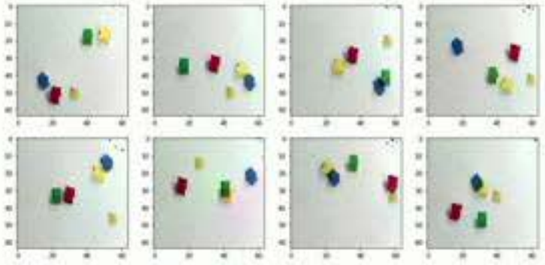




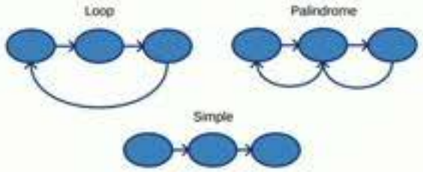
Cluster extracted controllers

```
# Looping sequence
c_list = [2,1,4,0,3]
for j in range(6):

    # Palindromic sequence
    count = 0
    for k in range(len(c_list)*2-1):
        execute(c_list[count])
        if (k >= len(c_list)-1):
            count = count-1
        else:
            count = count+1

    # Simple sequence
    execute(3)

# Simple sequence
execute(2)
execute(1)
execute(4)
execute(0)
execute(3)
```

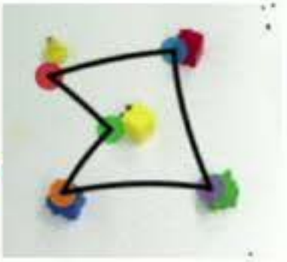
Extract symbolic controller sequence



Crop around goals and augment.

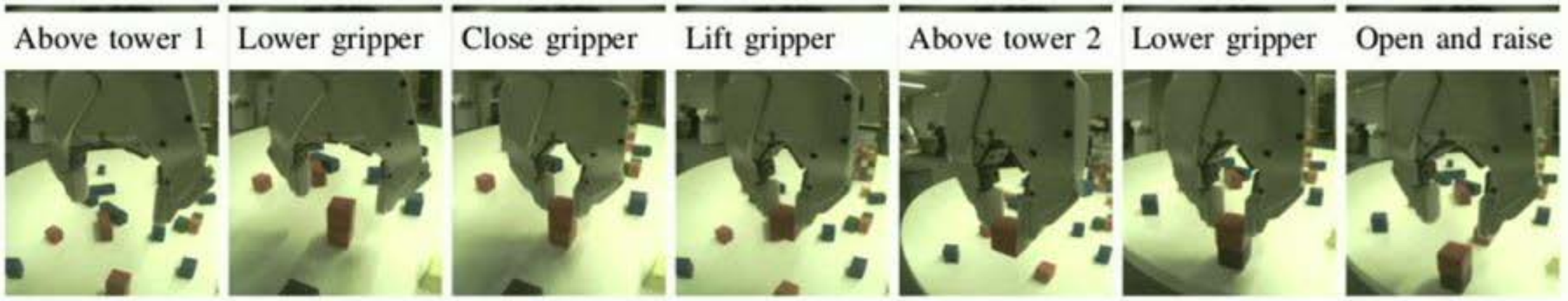


Extract program from symbolic sequence by searching for common structures.

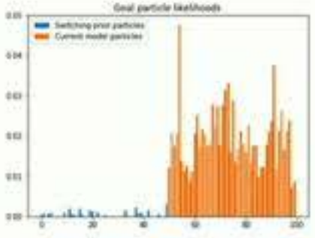


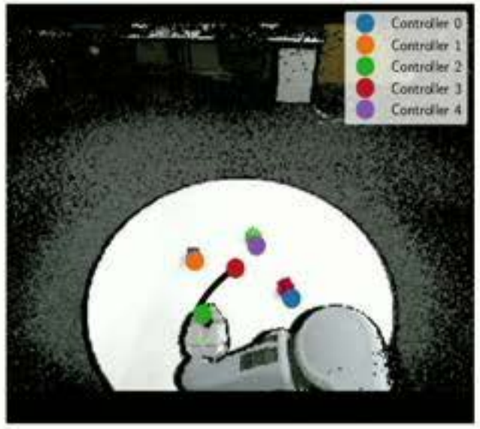
Train visual grounding networks for controller flexibility.

From explanation to synthesis



Infer controllers from visuomotor demonstration

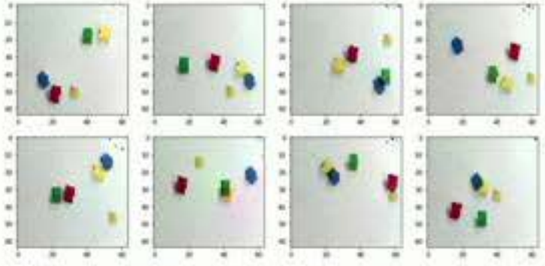




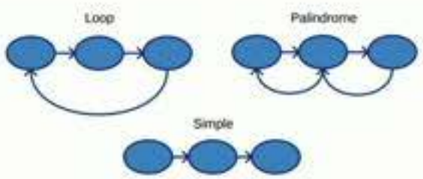
Cluster extracted controllers

```
# Looping sequence
c_list = [2,1,4,0,3]
for j in range(6):

    # Palindromic sequence
    count = 0
    for k in range(len(c_list)*2-1):
        execute(c_list[count])
        if (k >= len(c_list)-1):
            count = count-1
        else:
            count = count+1

    # Simple sequence
    execute(3)

# Simple sequence
execute(2)
execute(1)
execute(4)
execute(0)
execute(3)
```

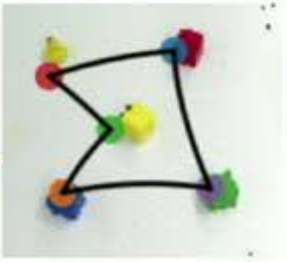
Extract symbolic controller sequence



Crop around goals and augment.

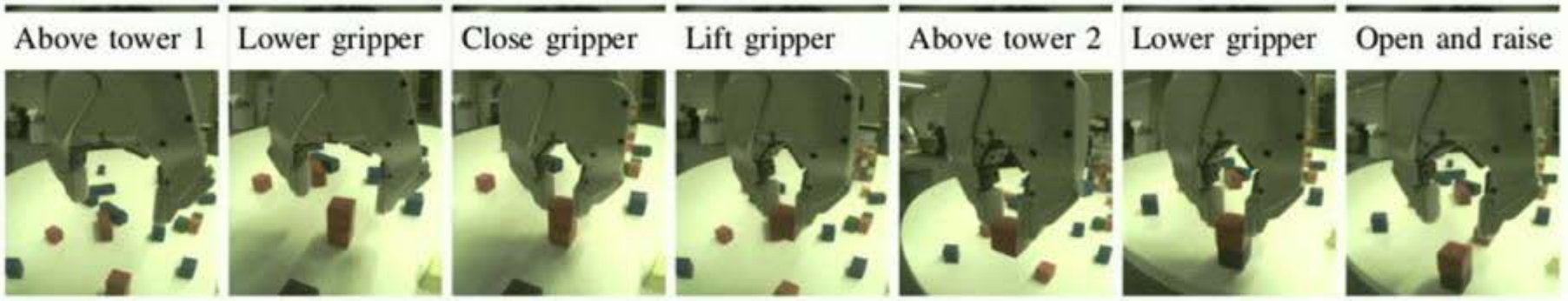


Extract program from symbolic sequence by searching for common structures.



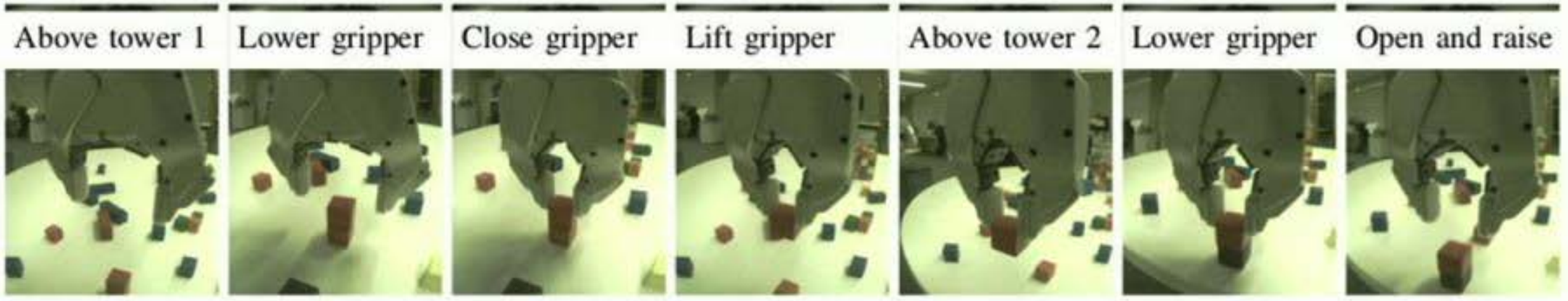
Train visual grounding networks for controller flexibility.

From explanation to synthesis

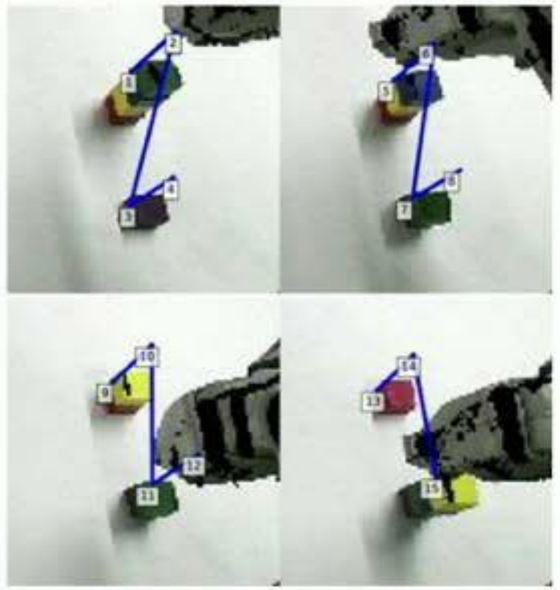


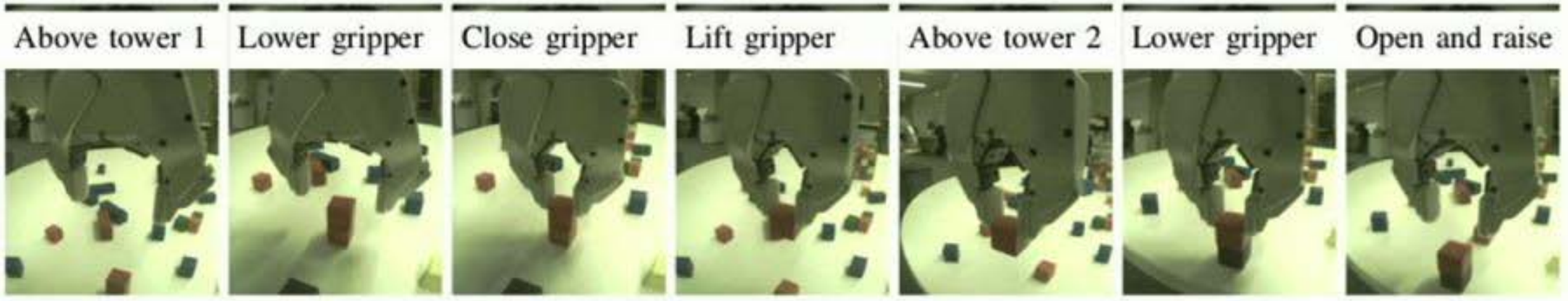
From explanation to synthesis

- Explained deep learning controllers using proportional control law sequences.
- Infer controller sequence using sensitivity analysis.
- Use inferred controllers to synthesise program.
- Ground controllers visually using perception networks.
- Program is flexible, generalisable and interpretable.



From explanation to synthesis





Did we lose anything in the explanation?

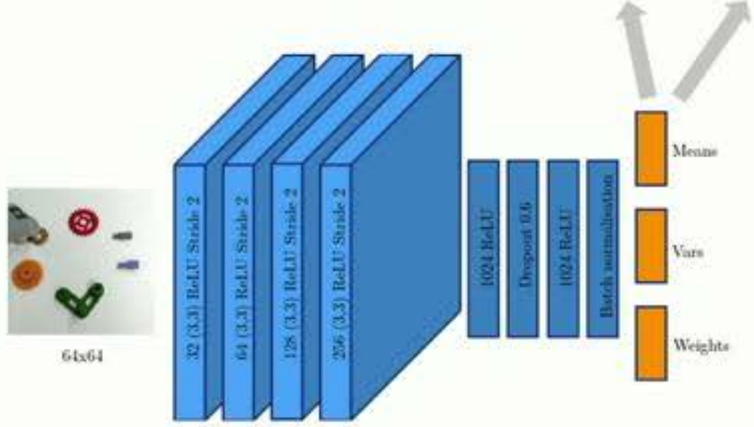
- Maybe…
- Gained flexibility, but might have lost something in the clustering process
- What if the model had been repelled by objects in the scene?

We added an inductive bias about the underlying program we expect to see.

- Goals correspond to visual content in scene
- Motions followed are proportional control laws

Predict controller gains/ goals states directly

$$\dot{\theta} = k_p^j(\theta - \theta_g^j)$$

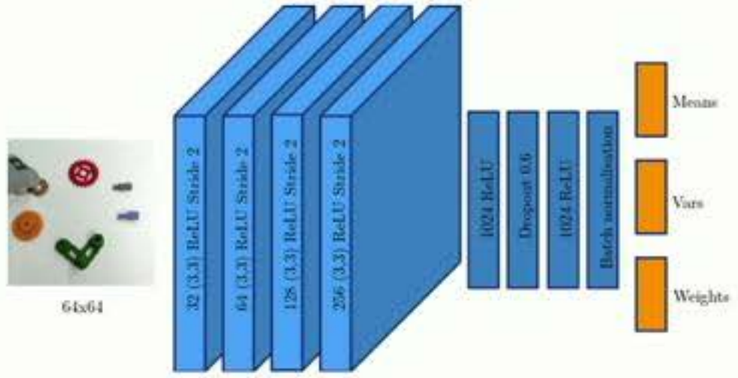


- Loss is difference between measured joint velocities and controller velocities.
- How can we deal with switching controllers?
- Mixture density networks are good for modelling multi-modal densities.

Mixture density networks
Bishop (1994)

$$p(\mathbf{x}|\mathbf{y}) = \sum_{i=1}^{K} \pi_i \mathcal{N}(\mathbf{x}|\mu(\mathbf{y}), \mathbf{\Sigma}(\mathbf{y}))$$

Mixture density networks are **not** good mixture models.

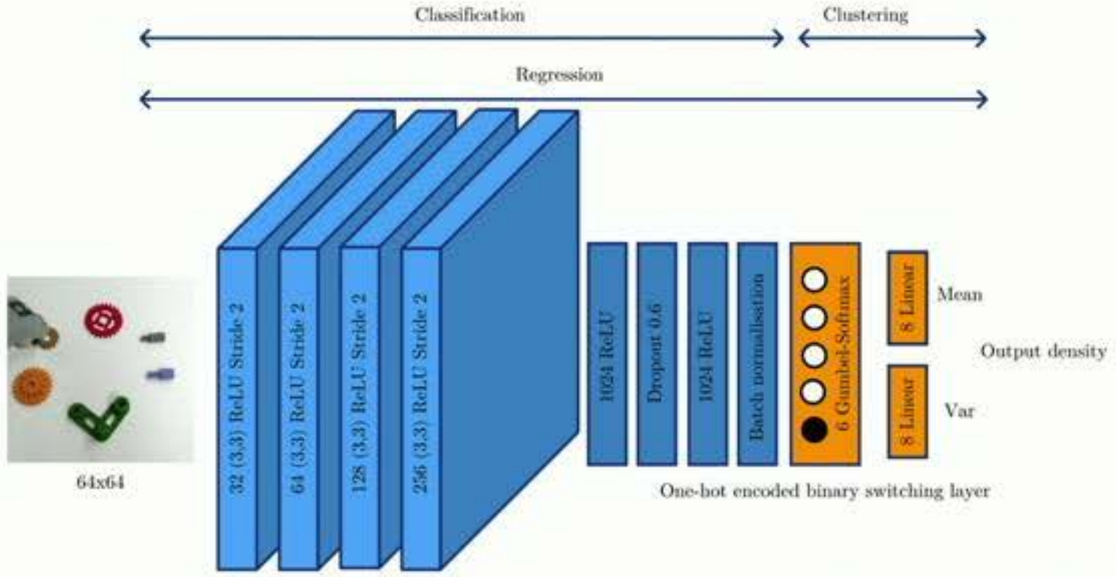


- MDNs are not performing clustering.
- Components are not intrinsically meaningful.
- No structure constraining network predictions.
- Arbitrary mapping from image to mixtures.

Mixture density networks
Bishop (1994)

Need structure for interpretability.

Switching density networks

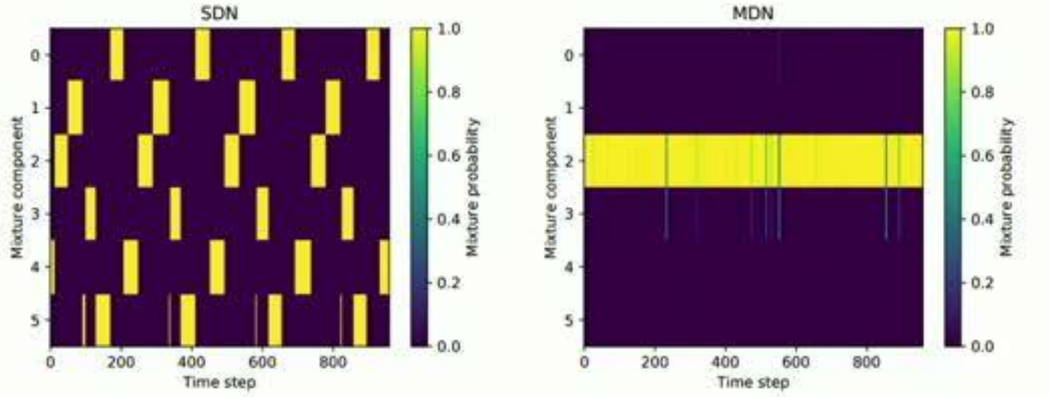


- Explicit discrete latent switching structure
- Maximum likelihood under single Gaussian
- Predict parameters of PID control laws

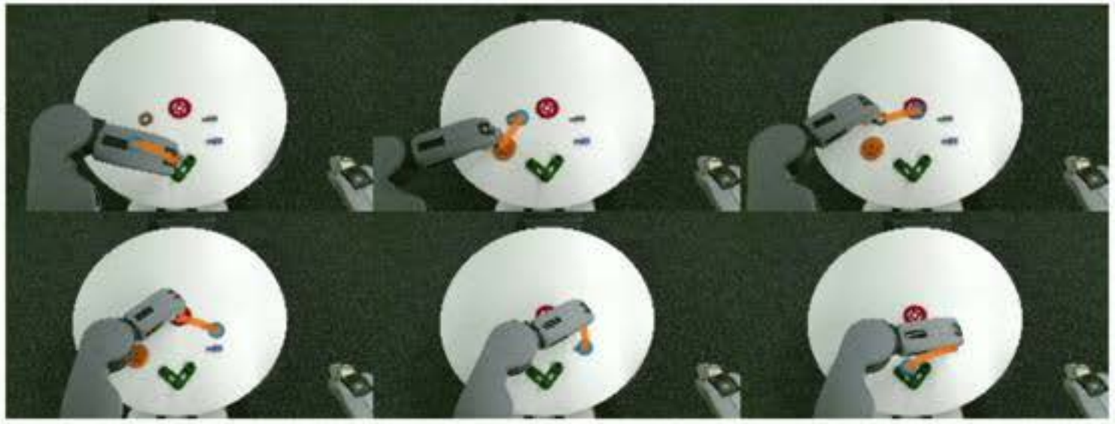
$$\mathbf{u}_k \sim K_p(\mathbf{z}_t, i_t)\left[\mathbf{x}_t - \boxed{\mu(\mathbf{z}_t, i_t)}\right] + K_i(\mathbf{z}_t, i_t)\sum_{l=1}^{L}\left[\mathbf{x}_{k-l} - \boxed{\mu(\mathbf{z}_t, i_t)}\right] + K_d(\mathbf{z}_t, i_t)\frac{\mathbf{x}_k - \mathbf{x}_{k-1}}{\Delta_t} + \mathcal{N}(\mathbf{0}, \boxed{\mathbf{\Sigma}(\mathbf{z}_t, i_t)}),$$

Switching density networks

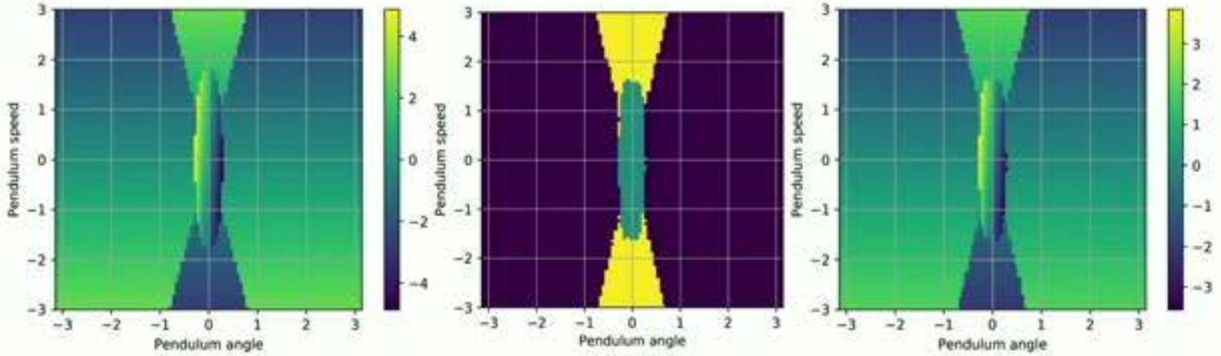
- SDNs learn both sub-control laws and the switching structure governing their use.



- Discrete transition dynamics can be used directly for program simplification.



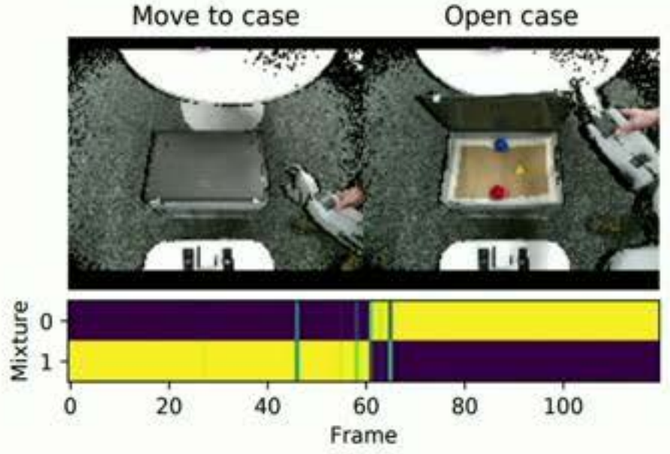
Hybrid System Identification



(a) Hybrid controller (b) Sub-controller regions (c) SDN response

Pendulum balancing using three proportional control laws

Suitcase opening using two PID control laws



Interpretability gains using SDNs

- Hierarchy allows for interpretability
- Gains for tasks with clear discrete latent structure (inspection)
- Minor losses for tasks without (pendulum)
- Predicting controller goal states more effective than predicting actions

Inspection

Controller	RMSE
SDN	0.8 degrees
CNN	1.47 degrees
MDN	3.95 degrees

Pendulum

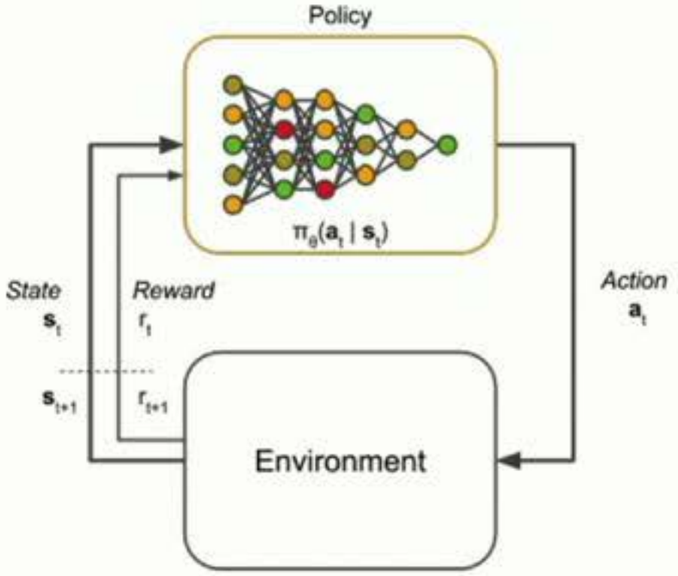
Controller	Reward
Hybrid controller	−0.812 ± 0.514
SDN PID Fully connected	−0.857 ± 0.621
Fully connected	−0.850 ± 0.538

Learning Programatically Structured Representations through Perceptor Gradients

[with S.V. Penkov, ICLR 2019]

27/09/19

Policy Gradients



Trace from rollout: $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$

$(r_1, \ldots, r_T), r_t = r(s_t, a_t)$

RL objective:

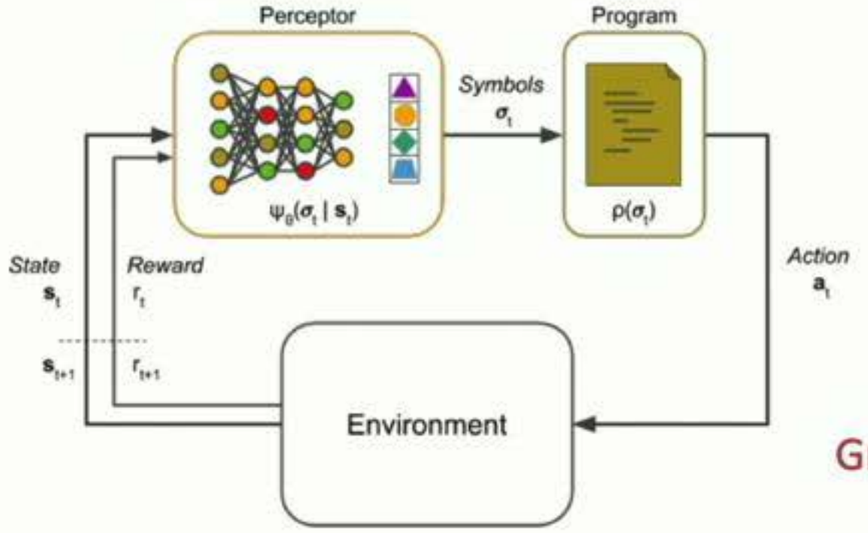
$$J(\theta) = E_{\tau \sim p(\tau;\theta)}[R_0(\tau)] = \int p(\tau;\theta) R_0(\tau) d\tau$$

$$R_t(\tau) = \sum_{i=t}^{T} \gamma^{i-t} r(s_t, a_t)$$

REINFORCE with baseline b_ϕ: $\nabla_\theta J(\theta) \sim \frac{1}{n} \sum_{i=1}^{n} \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) (R_t(\tau^{(i)}) - b_\phi(s^{(i)}))$

Perceptor Gradients

Any functional program
(incl. calls to a memory)



Policy structure:

$$\pi_\theta(a_t|s_t) = p(a_t|\sigma_t)\psi_\theta(\sigma_t|s_t)$$

$$p(a_t|\sigma_t) = \delta_{\rho(\sigma_t)}(a_t)$$

Gradient of log-likelihood of trace sampler $\tau^{(i)}$:

$$\nabla_\theta \log p(\tau^{(i)};\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \psi_\theta(\sigma_t^{(i)}|s^{(i)})$$

So, we can continue to use REINFORCE as before, with this gradient.

Policy factorisation preserves generality of policy gradients

For any decomposition of a policy π_θ into a program ρ and preceptor ψ_θ such that

$$\pi_\theta(\mathbf{a_t}|\mathbf{s_t}) = \delta_{\rho(\sigma_t)}(\mathbf{a_t})\psi_\theta(\sigma_t|\mathbf{s_t})$$

the gradient of the log-likelihood of a trace sample $\tau^{(i)}$ obtained by following π_θ is $\nabla_\theta \log p(\tau^{(i)};\theta) = \sum_{t=0}^{T-1} \nabla_\theta \log \psi_\theta(\sigma_t^{(i)}|\mathbf{s}^{(i)})$

$$\begin{aligned}
\nabla_\theta \log p(\tau^{(i)};\theta) &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) \\
&= \sum_{t=0}^{T-1} \nabla_\theta \log \left[\delta_{\rho(\boldsymbol{\sigma}_t^{(i)})}(\mathbf{a}_t^{(i)})\ \psi_\theta(\boldsymbol{\sigma}_t^{(i)}|\mathbf{s}_t^{(i)})\right] \\
&= \sum_{t=0}^{T-1} \left[\nabla_\theta \log \delta_{\rho(\boldsymbol{\sigma}_t^{(i)})}(\mathbf{a}_t^{(i)}) + \nabla_\theta \log \psi_\theta(\boldsymbol{\sigma}_t^{(i)}|\mathbf{s}_t^{(i)})\right] \\
&= \sum_{t=0}^{T-1} \nabla_\theta \log \psi_\theta(\boldsymbol{\sigma}_t^{(i)}|\mathbf{s}_t^{(i)})
\end{aligned}$$

Training Perceptors

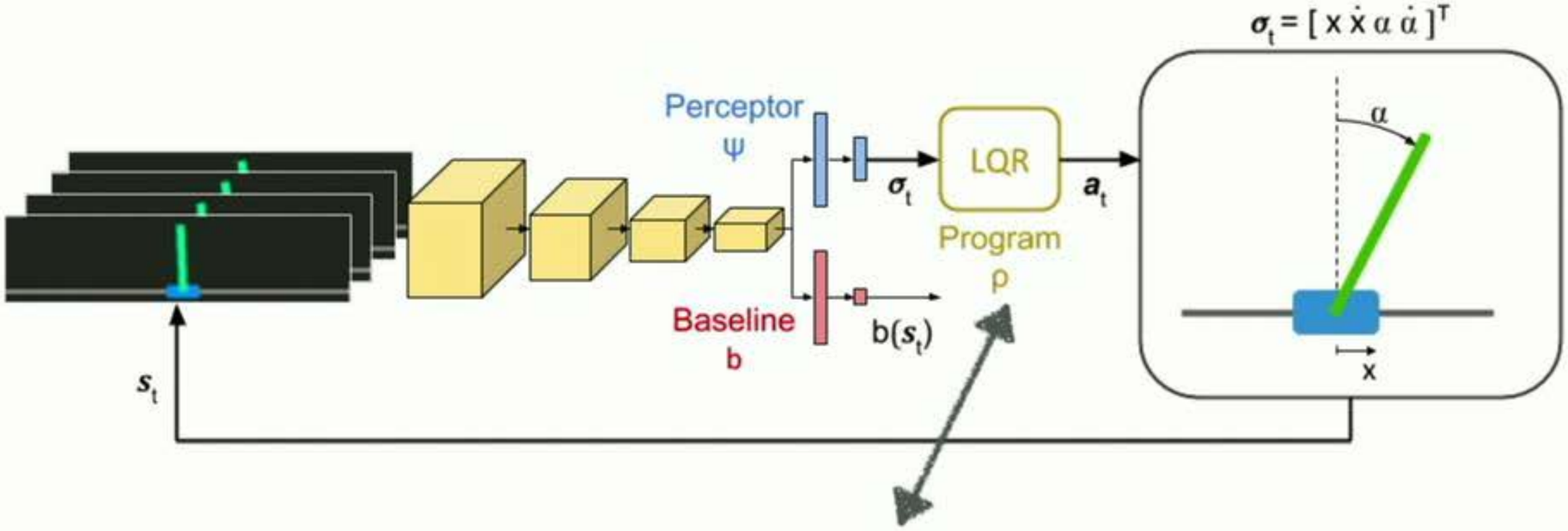
A loss function for a feedforward preceptor can be defined as:

$$\mathcal{L}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} \{\mathcal{L}_{\psi}(\tau^{(i)}, \theta) + \mathcal{L}_{b}(\tau^{(i)}, \phi)\}$$

$$\mathcal{L}_{\psi}(\tau^{(i)}, \theta) = \sum_{t=0}^{T-1} \log \psi_{\theta}(\sigma_t^{(i)} | s_t^{(i)})(R_t(\tau^{(i)}) - b_{\phi}(s_t^{(i)}))$$

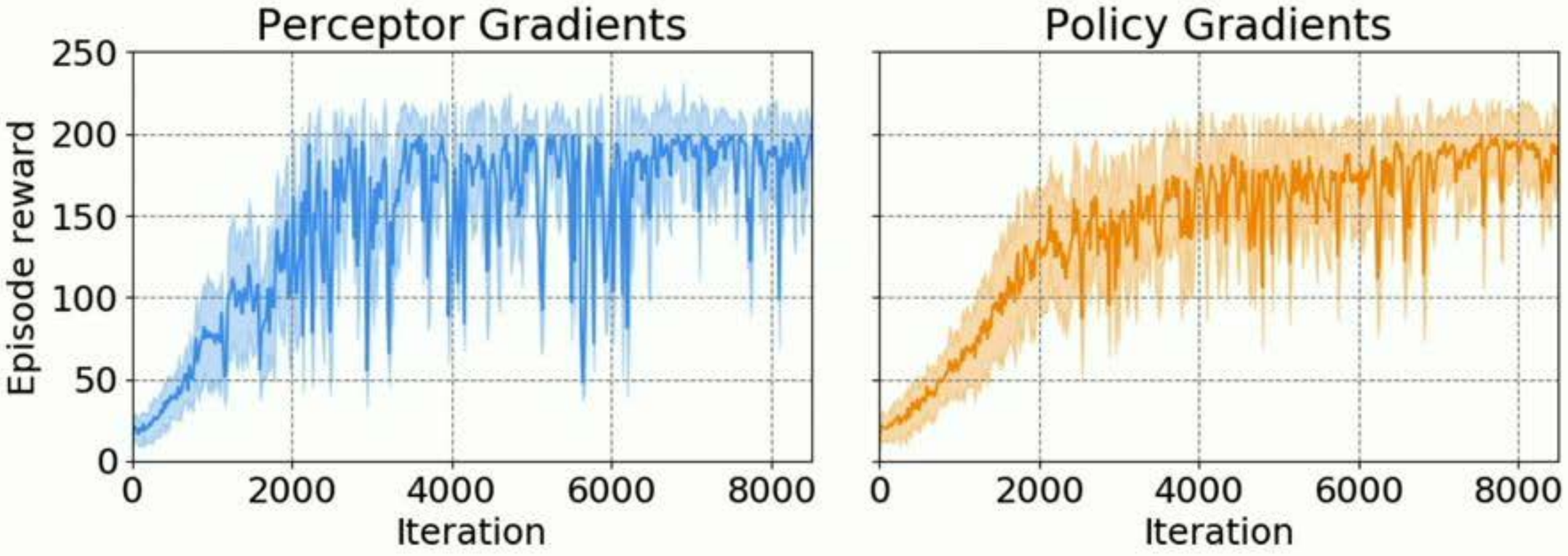
$$\mathcal{L}_{b}(\tau^{(i)}, \phi) = \sum_{t=0}^{T-1} (R_t(\tau^{(i)}) - b_{\phi}(s_t^{(i)}))^2$$

Example: Cart-pole Control; LQR structure

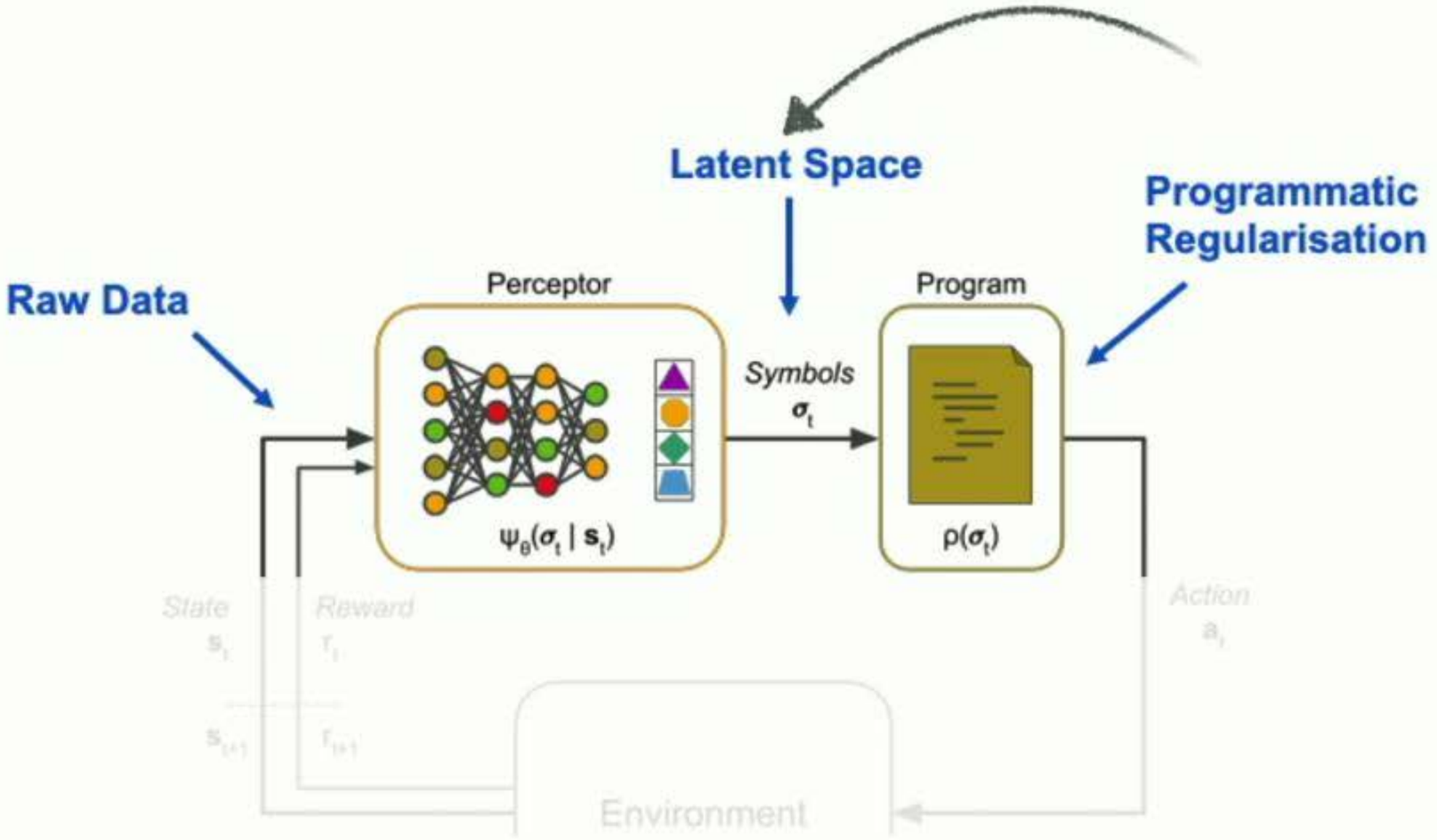


$u = -K\sigma$ where K minimizes $J = \int_0^\infty \sigma(\mathbf{t})^{\mathbf{T}}\mathbf{Q}\sigma(\mathbf{t}) + \mathbf{u}(\mathbf{t})^{\mathbf{T}}\mathbf{R}\mathbf{u}(\mathbf{t})\mathbf{dt}$

Learning performance:
Perceptor vs Policy Gradients



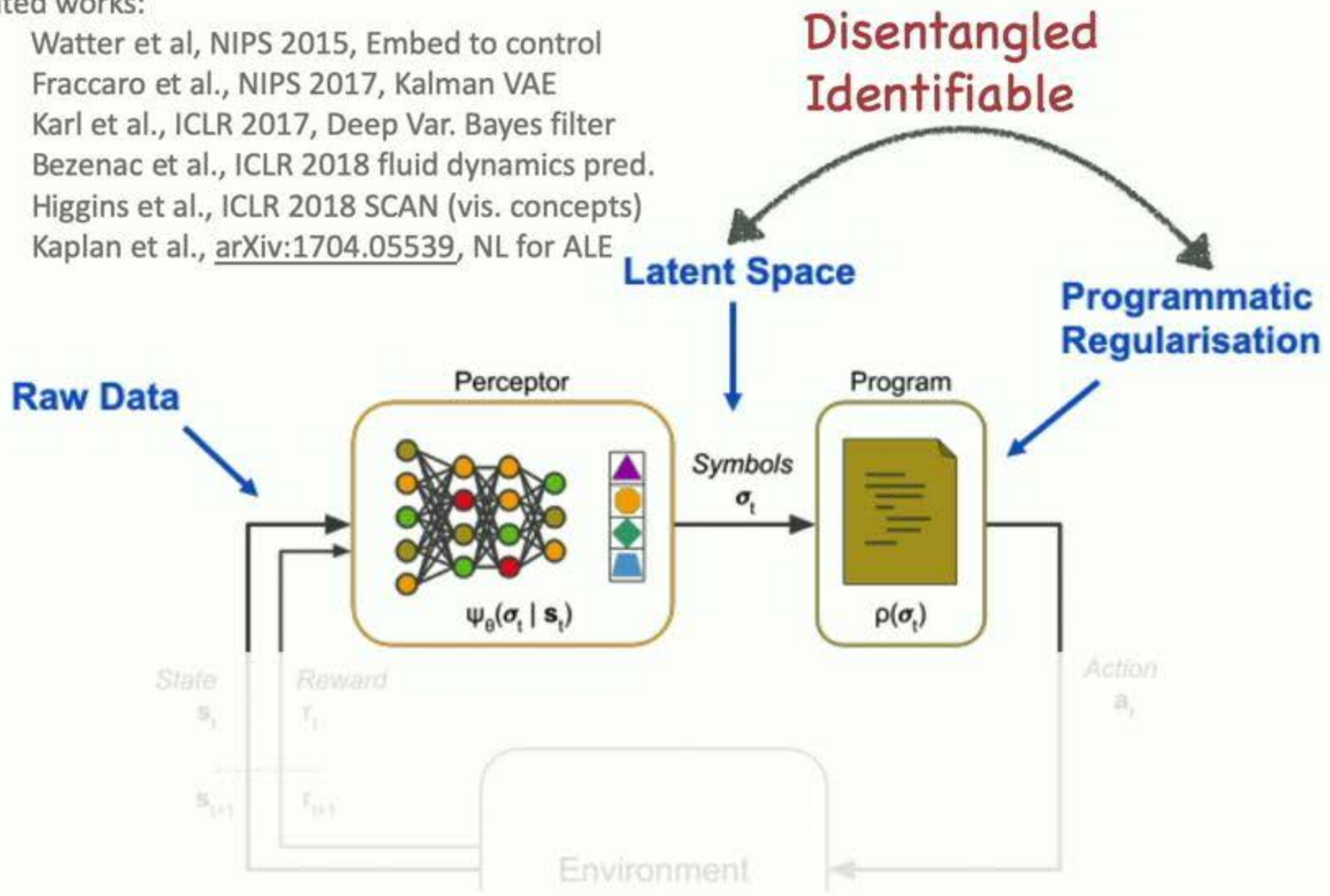
What is going on here?



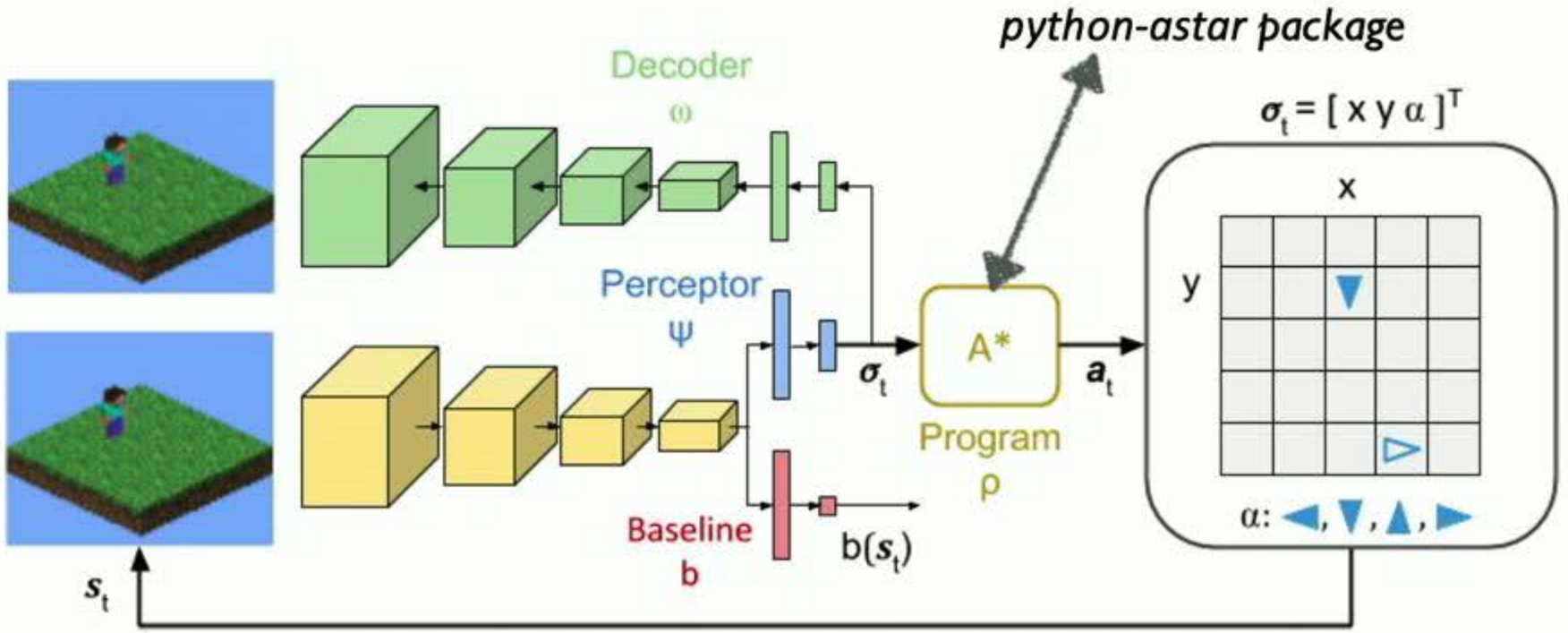
What is going on here?

Related works:

Watter et al, NIPS 2015, Embed to control
Fraccaro et al., NIPS 2017, Kalman VAE
Karl et al., ICLR 2017, Deep Var. Bayes filter
Bezenac et al., ICLR 2018 fluid dynamics pred.
Higgins et al., ICLR 2018 SCAN (vis. concepts)
Kaplan et al., arXiv:1704.05539, NL for ALE

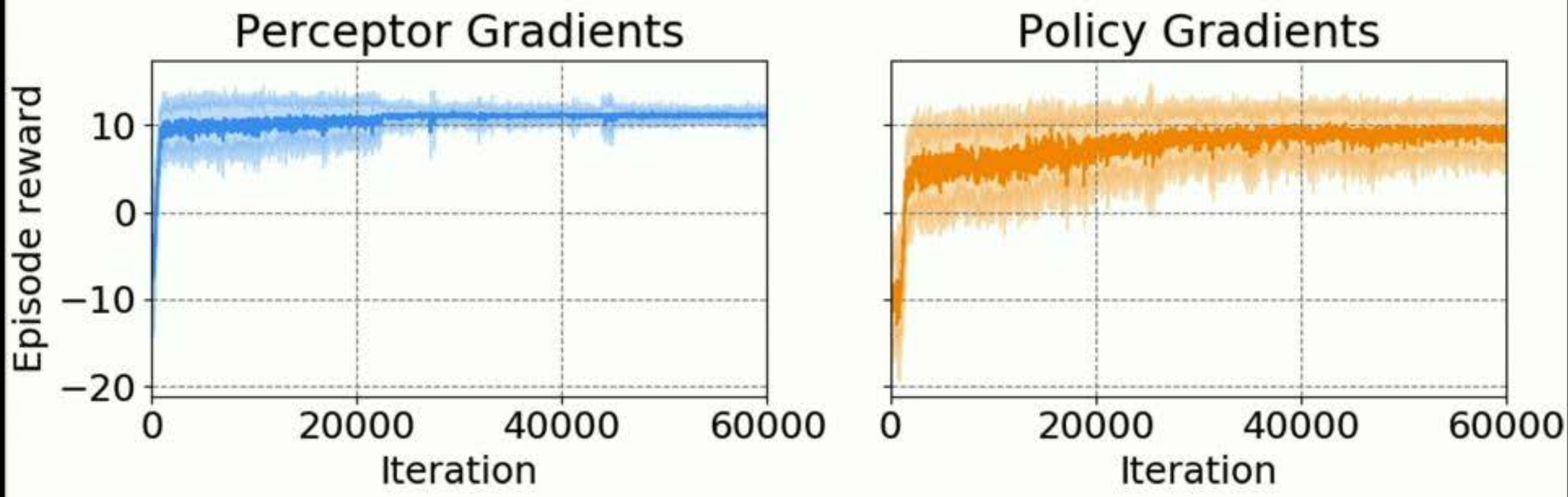


go-to-pose in Minecraft

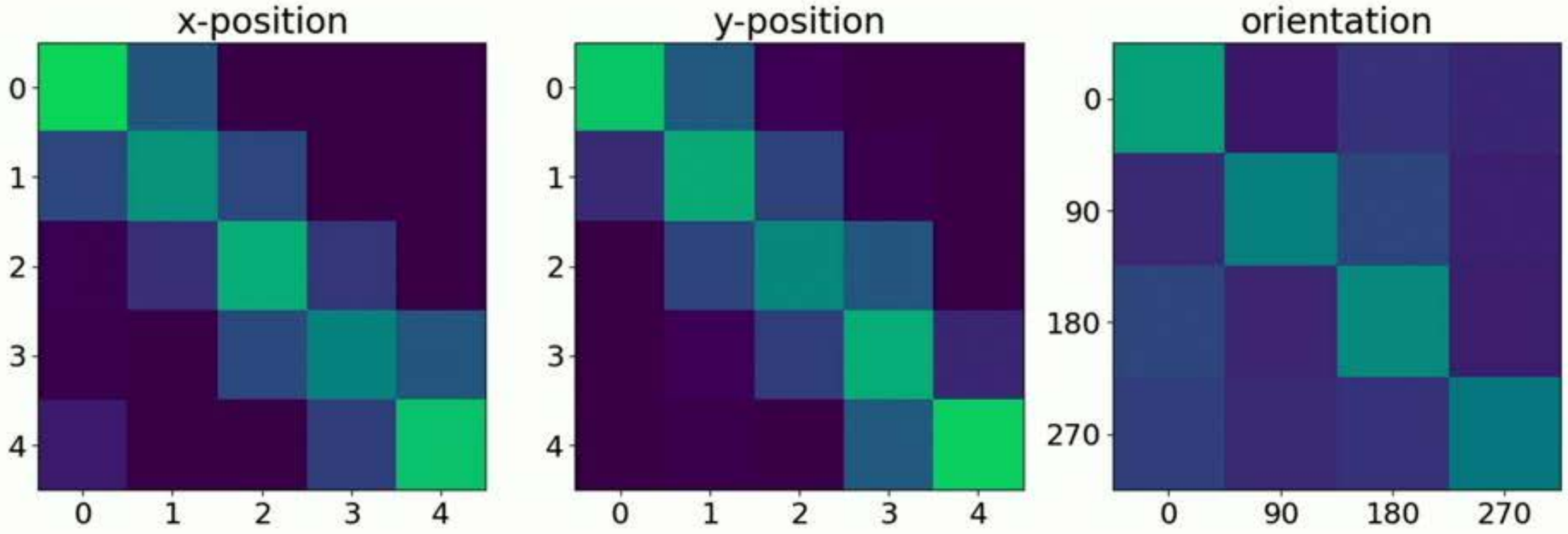


Loss function augmented with reconstruction term: $\mathcal{L}_\omega(\tau^{(i)}, \theta) = \sum_{t=0}^{T-1} \log \omega_\nu(s_t^{(i)} | \sigma_t^{(i)})$

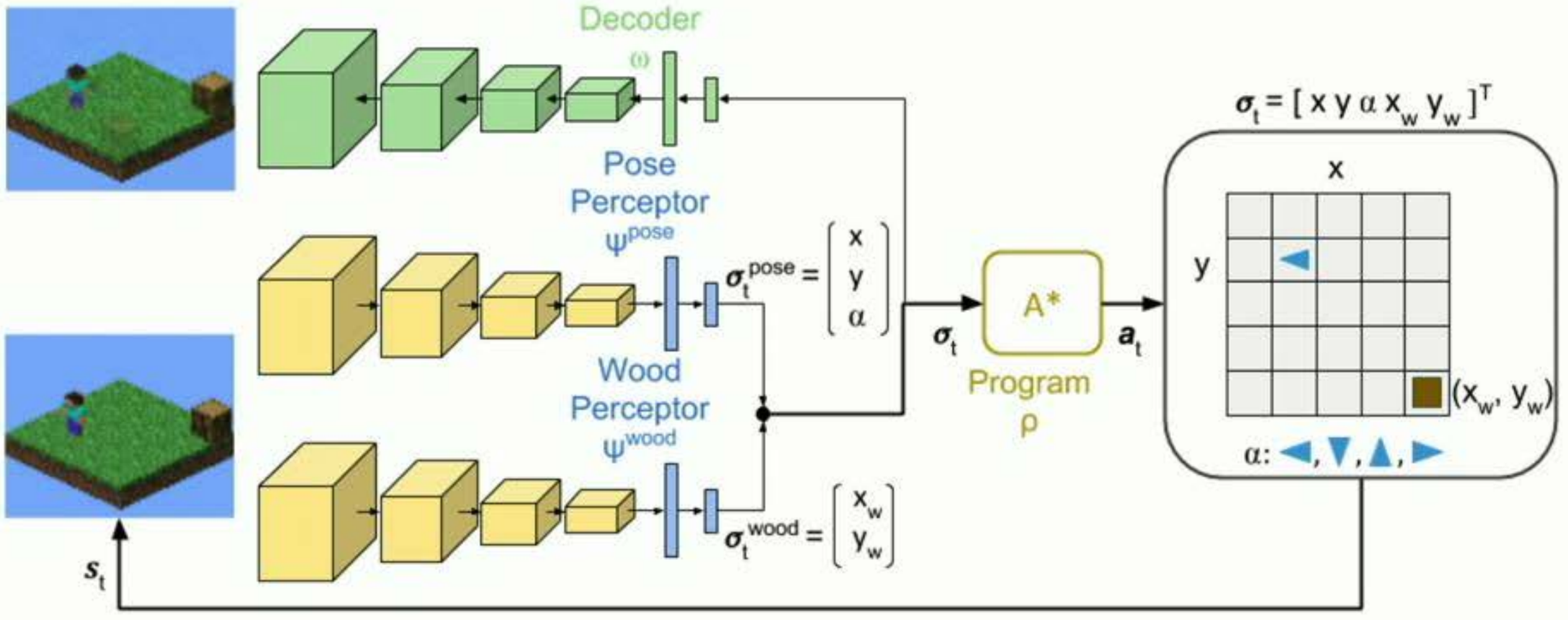
Learning performance for go-to-pose



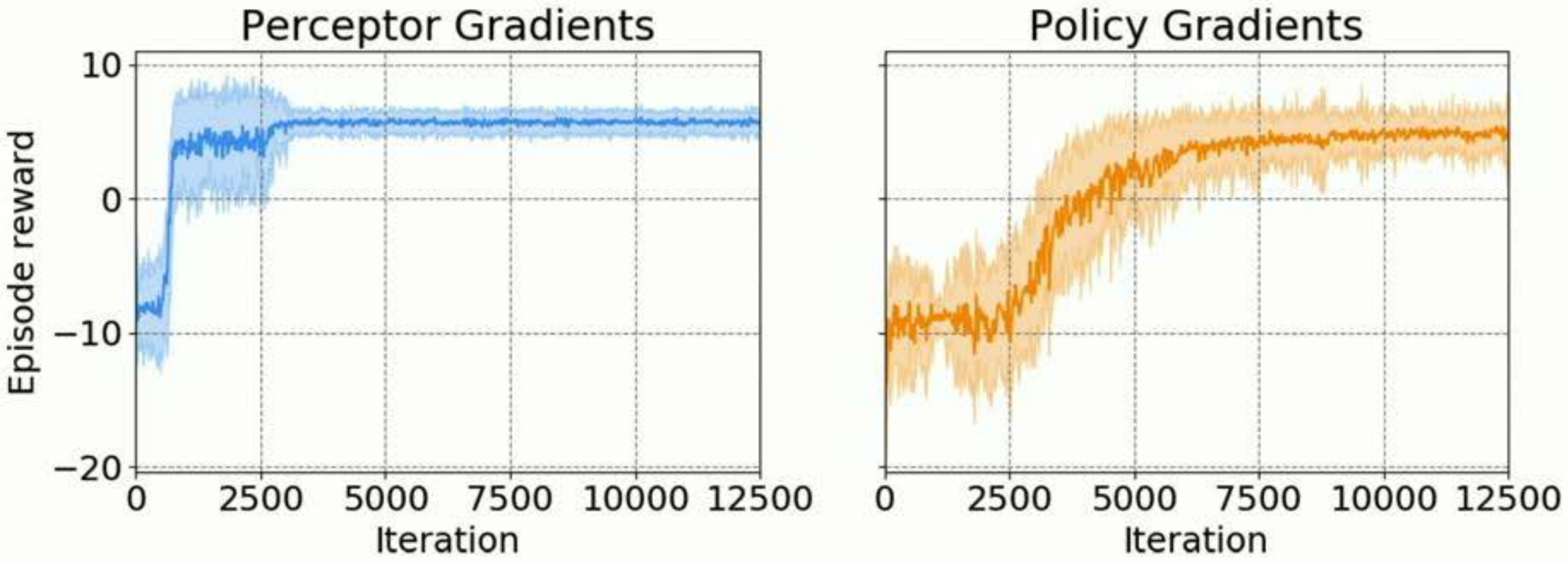
Inspecting the Latent Space



Stacked Perceptors: get wood in Minecraft

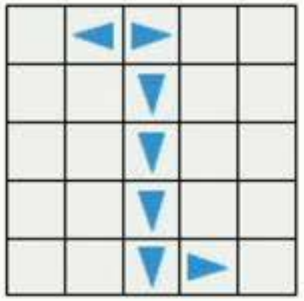


Learning performance for ‘get wood’



Generating states from symbols

Symbolic Trajectory



Generated States Along Trajectory

t = 1

t = 2

t = 3

t = 4

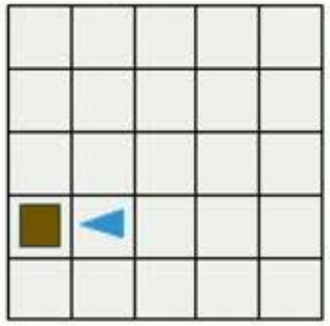
t = 5

t = 6

t = 7

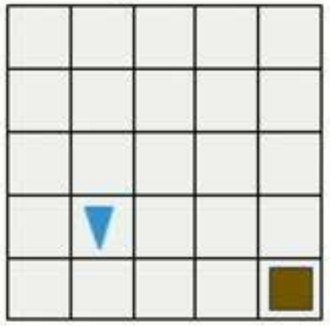
Generating states with relational structure

Symbols



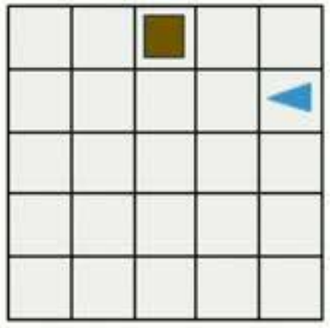
Sample

Symbols



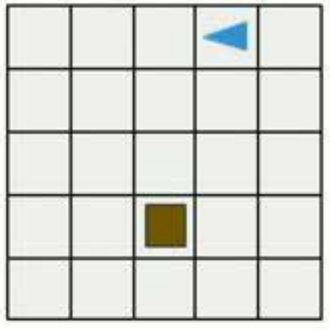
Sample

Symbols



Sample

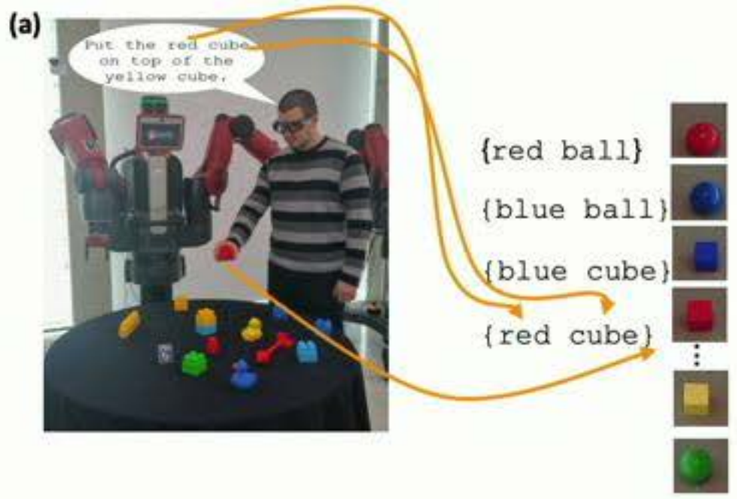
Symbols

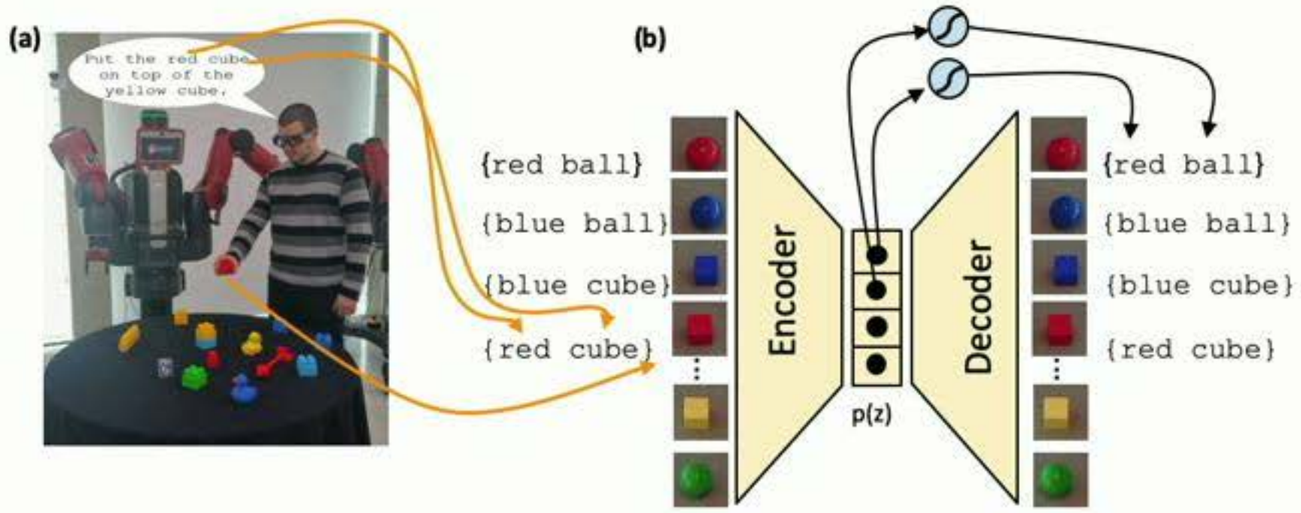


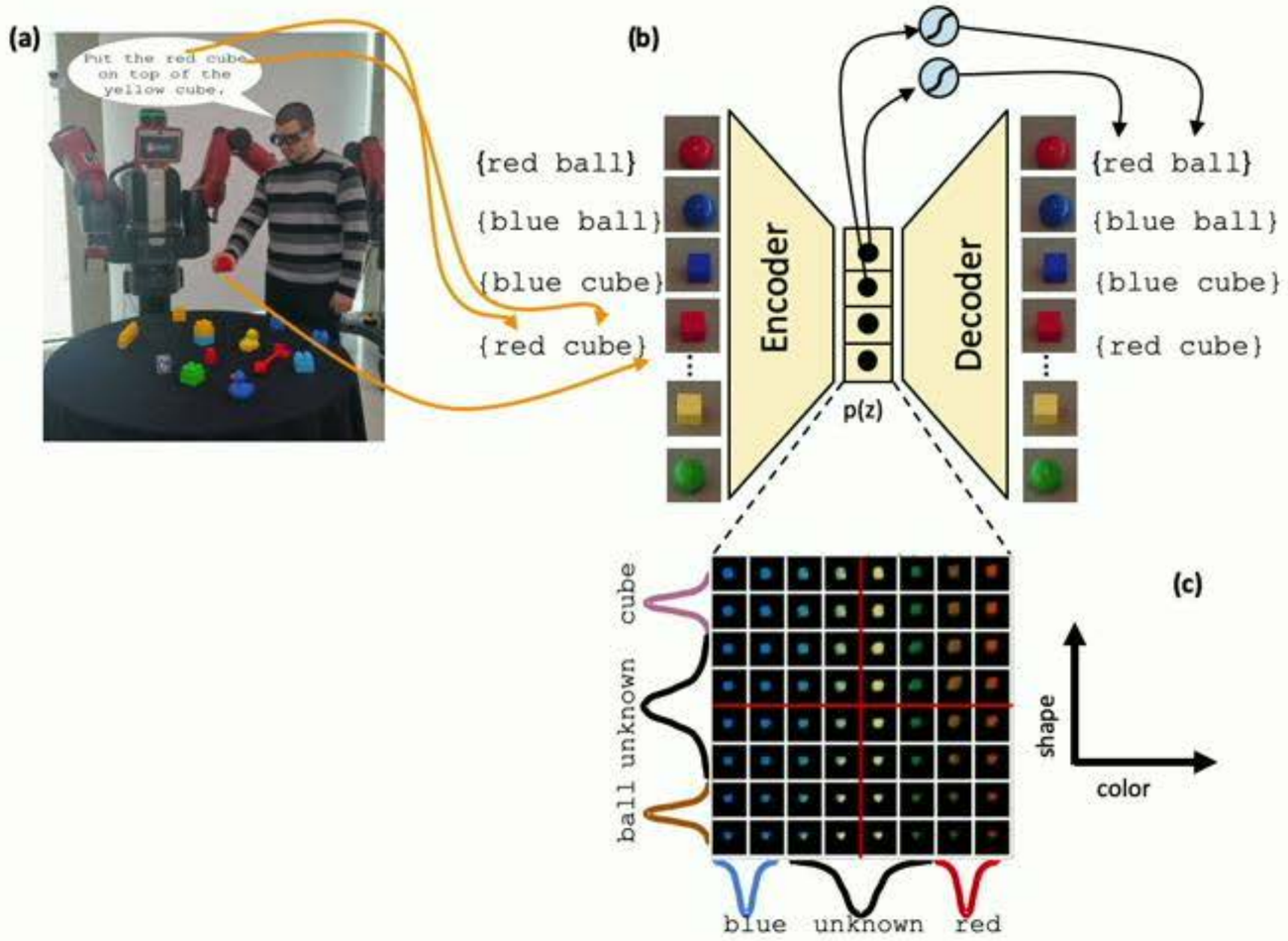
Sample

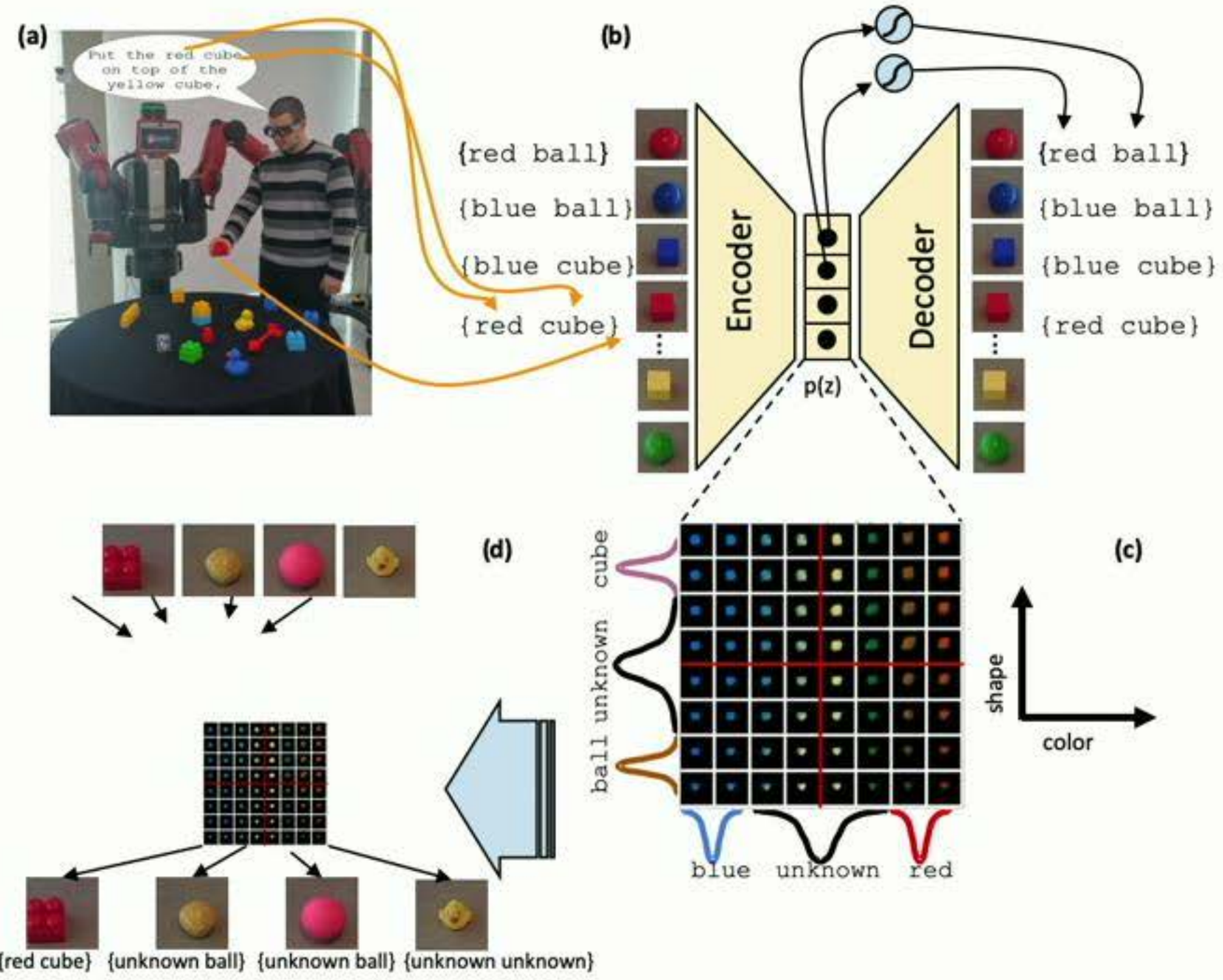
Learning and Using Interpretable Latent Spaces

[with Y. Hristov, A. Lascarides, et al. CoRL 2018, CoRL 2019]

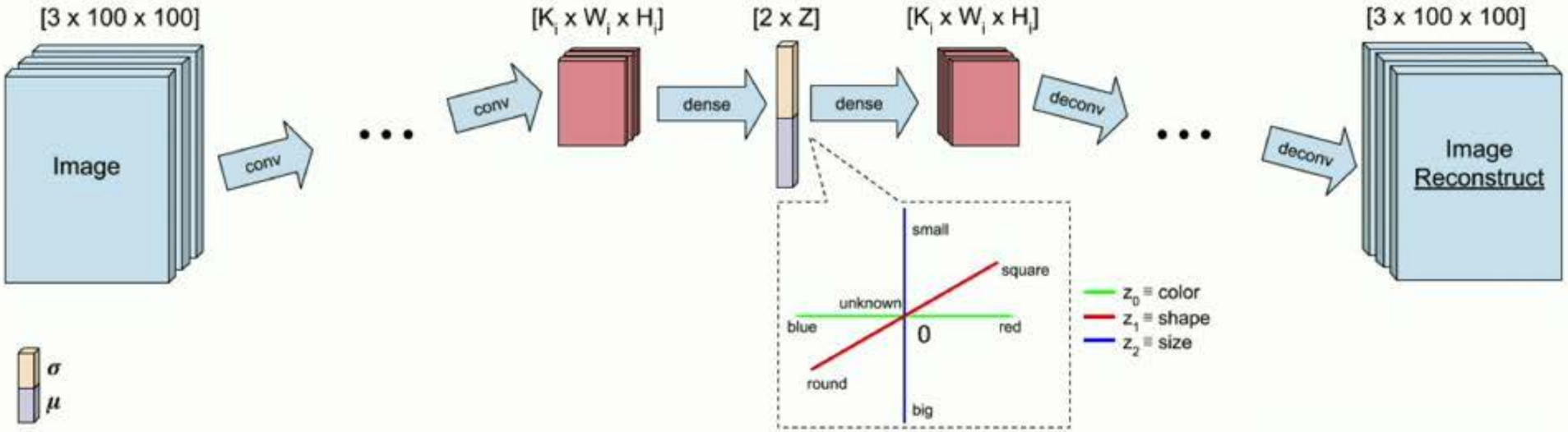

(a)
Put the red cube on top of the yellow cube.
{red ball}
{blue ball}
{blue cube}
{red cube}


(a)
Put the red cube on top of the yellow cube.
{red ball}
{blue ball}
{blue cube}
{red cube}
(b)
Encoder
p(z)
Decoder
{red ball}
{blue ball}
{blue cube}
{red cube}


(a)
Put the red cube on top of the yellow cube.
(b)
{red ball}
{blue ball}
{blue cube}
{red cube}
Encoder
p(z)
Decoder
{red ball}
{blue ball}
{blue cube}
{red cube}
(c)
cube
ball unknown
blue
unknown
red
shape
color


(a)
Put the red cube on top of the yellow cube.
(b)
{red ball}
{blue ball}
{blue cube}
{red cube}
Encoder
p(z)
Decoder
{red ball}
{blue ball}
{blue cube}
{red cube}
(c)
cube
ball unknown
shape
color
blue
unknown
red
(d)
{red cube}
{unknown ball}
{unknown ball}
{unknown unknown}

Model Architecture



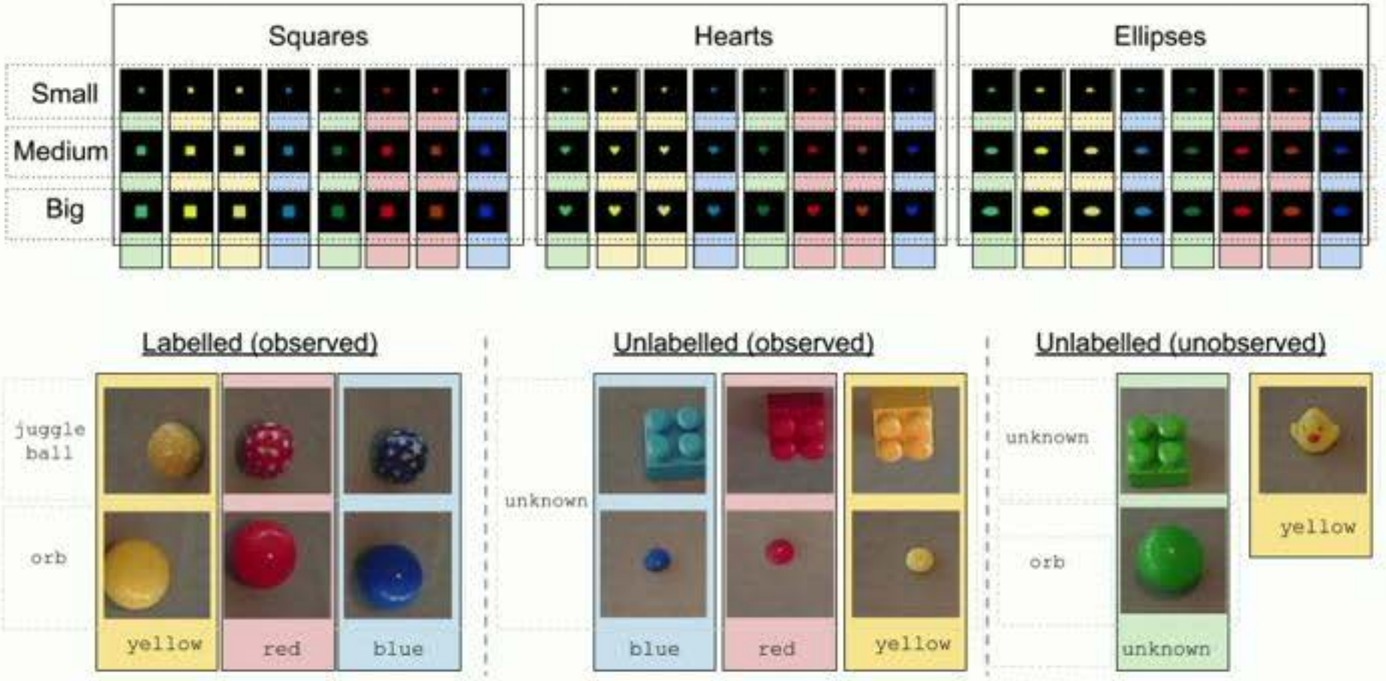
$$\min_{\theta,\phi,\mathbf{W}} \mathcal{L}(\mathbf{x},\mathbf{y},\theta,\phi) = \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) - \alpha\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}(\log p_\theta(\mathbf{x}|\mathbf{z})) + \gamma \sum_i^{|\mathcal{G}|} H(z_i \mathbf{w}_i^T, \mathbf{y}_i)$$

- Kullback-Leibler divergence term
- Reconstruction term
- Classification term

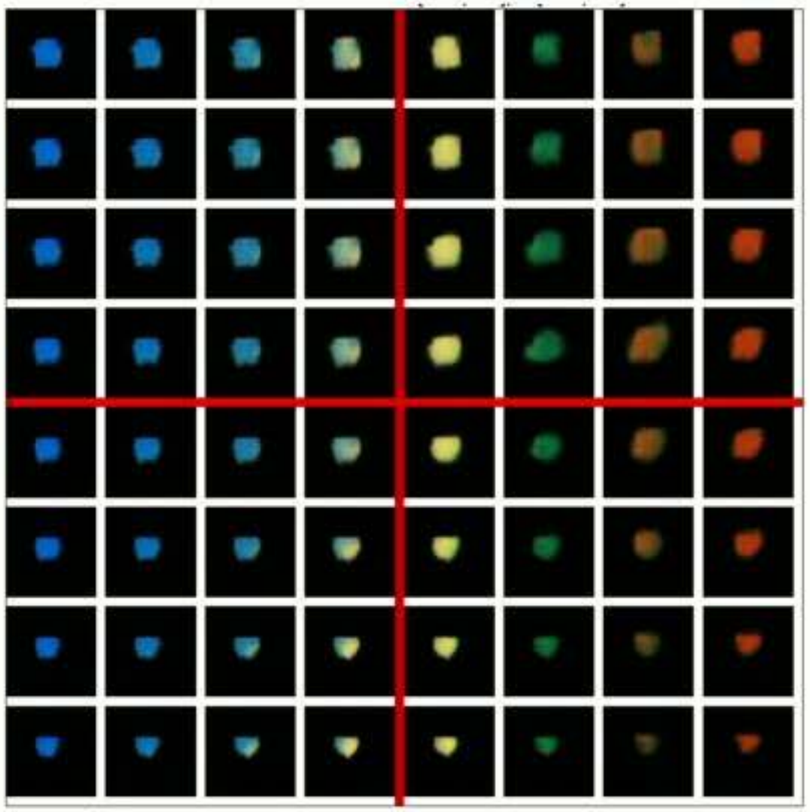
Baselines and Data

- Full Model (Ours)
- Vanilla $\boldsymbol{\beta}$-VAE ($\boldsymbol{\gamma} = 0$)
- Conv Classifier Network (CCN, $\boldsymbol{\alpha} = 0$)

- Synthetic colored d-Sprites
- Real-world tabletop dataset



Evaluation Criteria

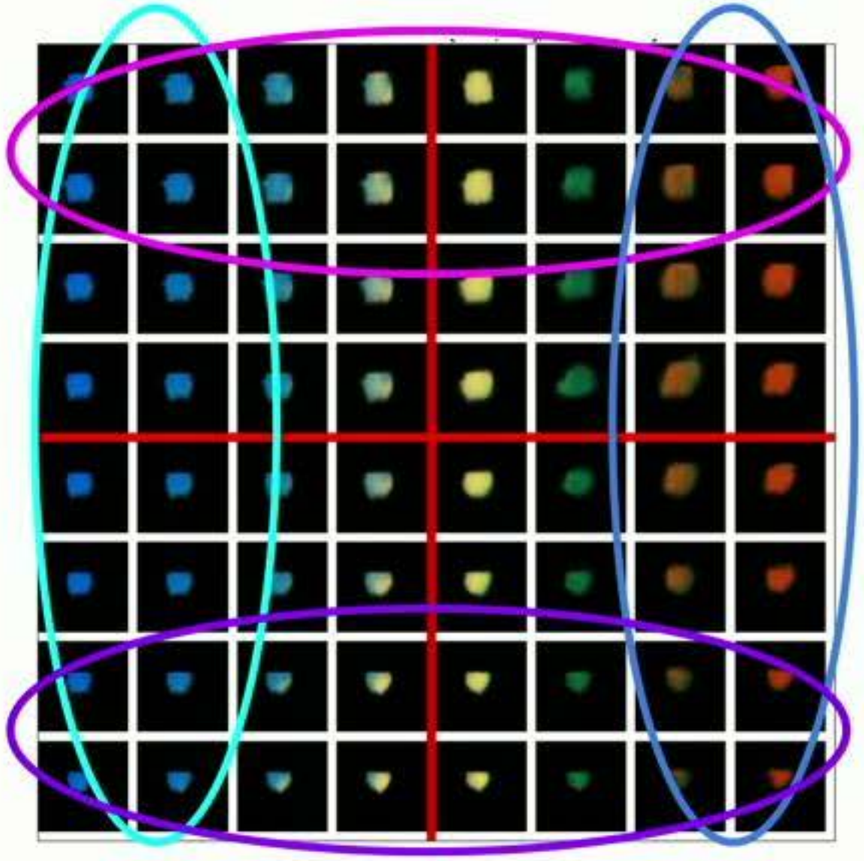


Factors of variation should be:

- Axes-aligned
- Linearly Separable

in the latent space.

Evaluation Criteria

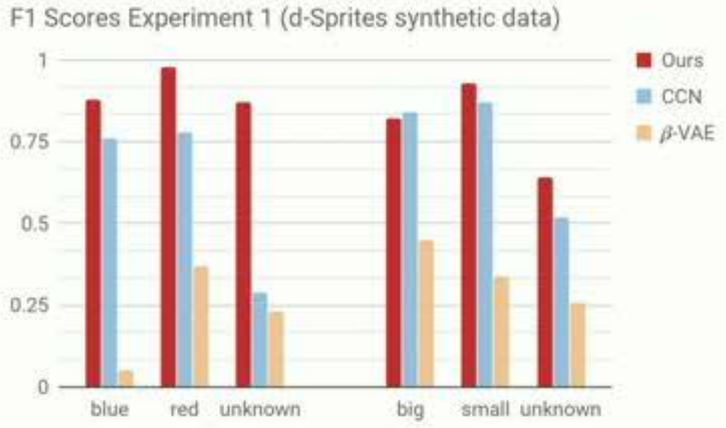


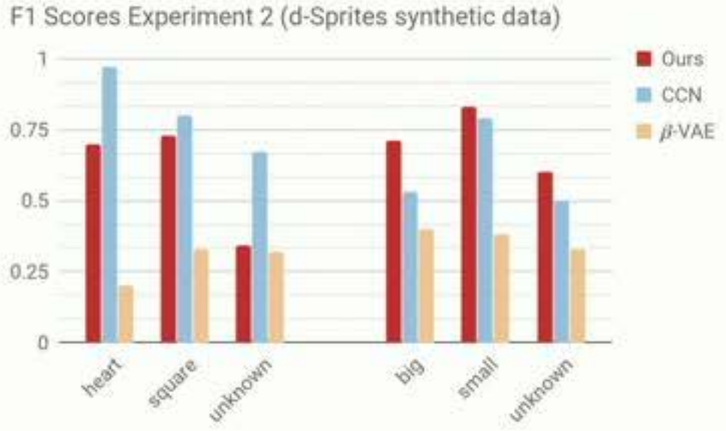
Factors of variation should be:

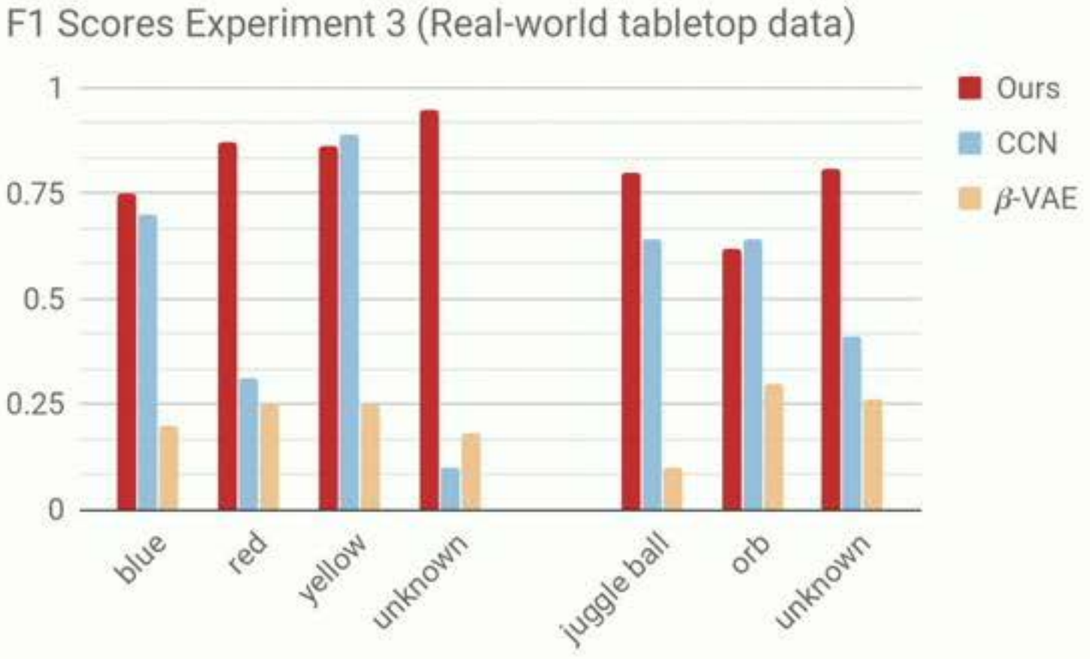
- Axes-aligned
- Linearly Separable

in the latent space.

Linear Separability Evaluation - F1 Scores







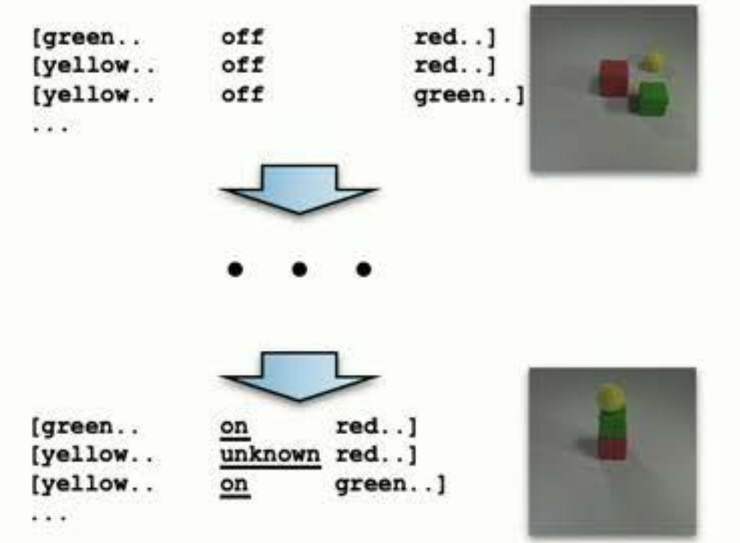
Disentangled Relational Representations for LfD

- Can we have useful *shared task representations?*
- Useful = *interpretable* + usable in models for planning
- Our framework allows human demonstrators to teach how to *ground high-level spatial concepts in their sensory input*

Claim: If we explicitly optimize for *disentanglement*, the learned latent space is also *useful in tasks downstream*.

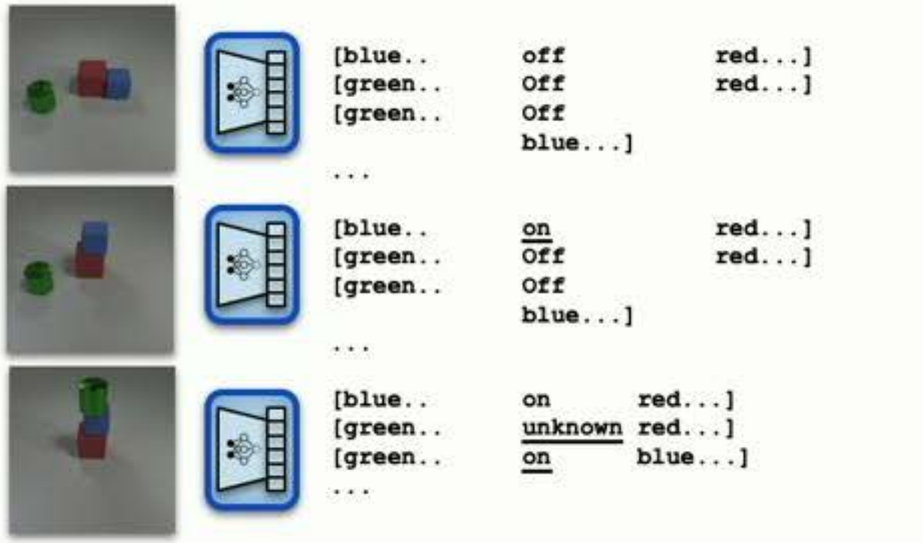
Overall Setup - Explain and Repeat for Spatially-grounded Plans

(a) **Capture partially-labelled training pairs**



Learn representations in which *object* and *relational symbols* (above) can be grounded

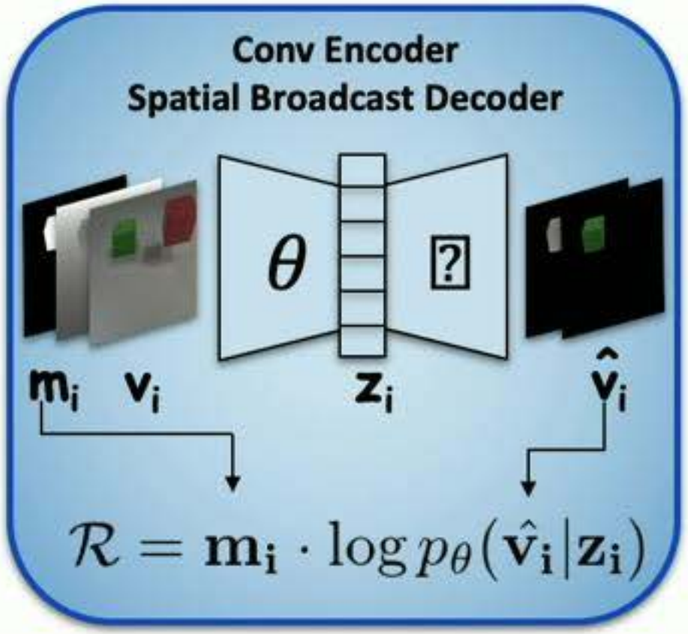
(b) **Test-time demonstrations for a task with extracted invariant plan**

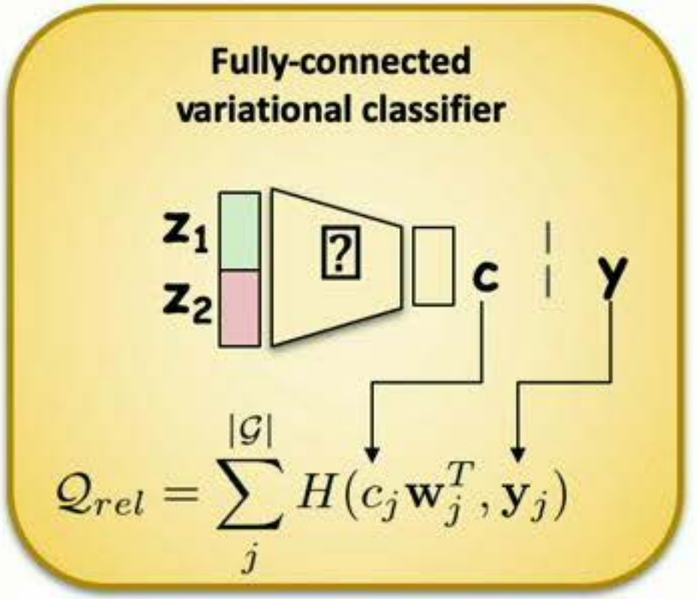


Project a set of new demonstrations in the learned representation space and using the grounded symbols infer an invariant (wrt all demos) *plan*

```
Plan:
[1] put blue on red
[2] put green on blue
```

Neural Models - Partially Inspired by MONet (Burgess et al.)

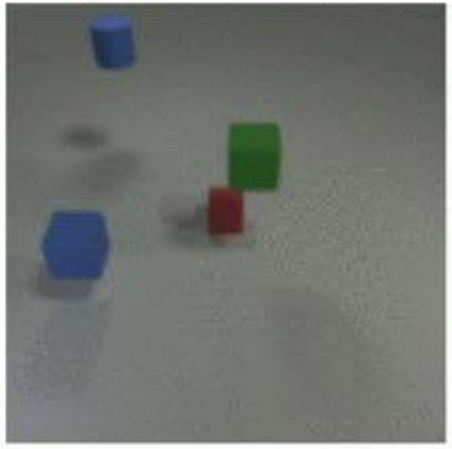




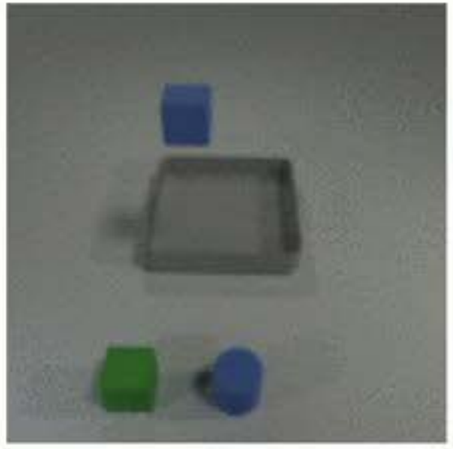
$$\min_{\theta,\phi,\psi,\mathbf{W},\mathbf{W}^o} \mathcal{L}(\mathbf{x},\mathbf{y},\mathbf{o},\theta,\phi,\psi) = \beta\mathcal{KL}(\mathbf{C}||\mathbf{Z}) + \alpha\mathcal{R} + \gamma(\mathcal{Q}_{obj} + \mathcal{Q}_{rel}),$$

$$\mathcal{Q} = \mathcal{Q}_{obj} + \mathcal{Q}_{rel} = \sum_i^2 \sum_o^{|\mathcal{O}|} H(z_{io}\mathbf{w}_o^T, \mathbf{o}_o) + \sum_j^{|\mathcal{G}|} H(q_\psi(c_j|\mathbf{z}_1,\mathbf{z}_2)\mathbf{w}_j^T, \mathbf{y}_j) \quad (1)$$

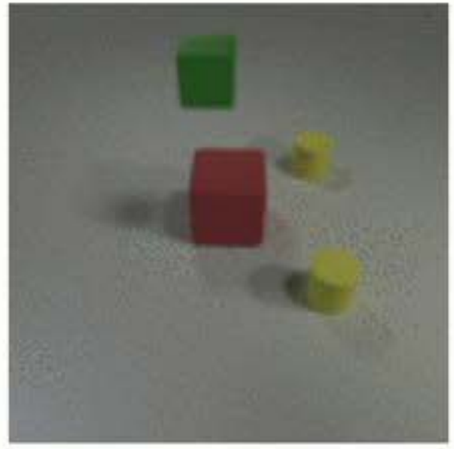
Testing on Repetitive Motions (Synthetic)



repetitive left-right



repetitive out-in



repetitive off-on

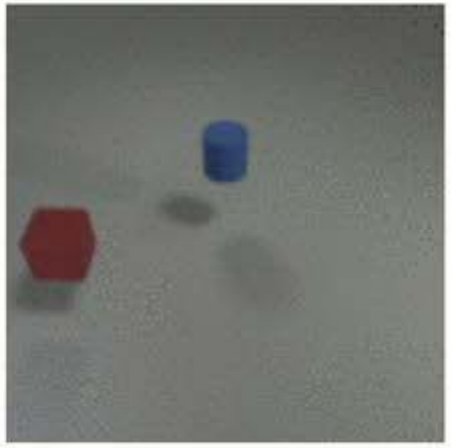
• • •

Model	left-right	front-behind	below-above	far-close	off-on	out-in
No $\mathcal{R}$, No $\mathcal{Q}_{obj}$	0.50	0.64	0.54	0.56	0.49	0.66
No $\mathcal{R}$, With $\mathcal{Q}_{obj}$	0.53	0.68	0.68	0.63	0.65	0.62
With $\mathcal{R}$, No $\mathcal{Q}_{obj}$	0.70	0.73	0.69	0.68	0.64	**0.78**
With $\mathcal{R}$, With $\mathcal{Q}_{obj}$	**0.80**	**0.88**	**0.91**	**0.86**	**0.76**	0.56

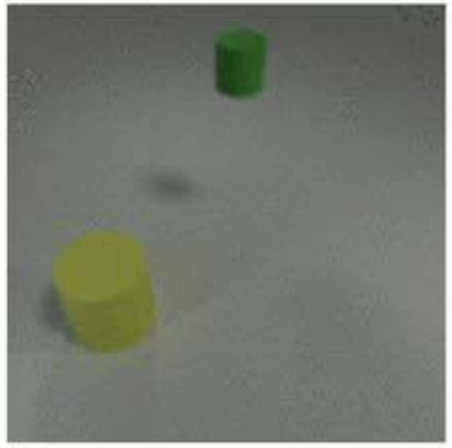
Plan segmentation Acc - *what moves when* - for repetitive demos

Testing on Chained Motions (Synthetic)

C-shape

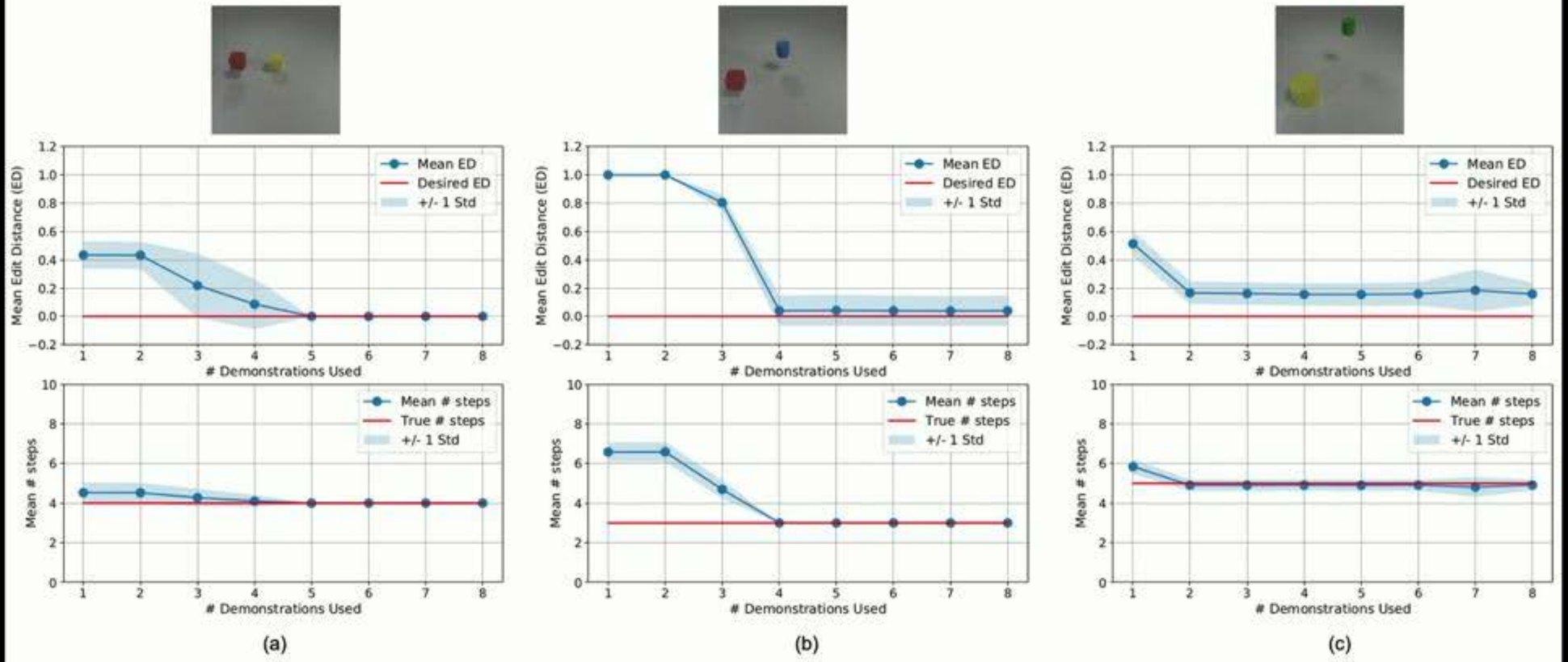


Jump over



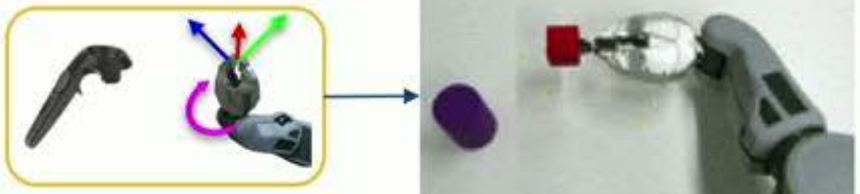
off-on-off

Results - Chained Motions



(top): edit distance statistics as a function of how many demonstrations the agent has seen. **(bottom)** plan length statistics for the inferred plans as a function of how many demonstrations the agent has seen for all three chained behaviours—**(a)** C-shape, **(b)** off-on-off and **(c)** jump over;

Training/Testing Data
(Robot Tabletop Manipulation)

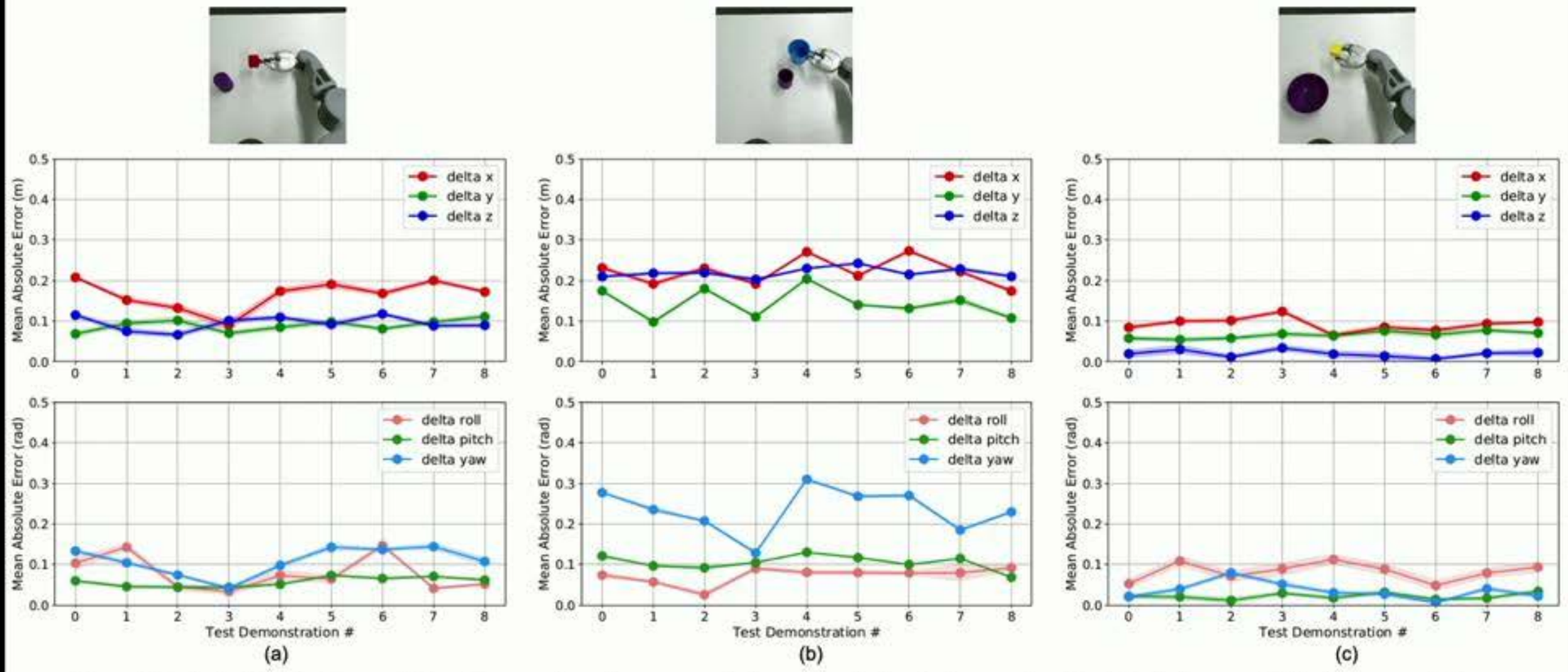


Place a cube ***on*** a cylinder

Make 2 cups ***face*** each other

Put a cube ***in*** a bowl

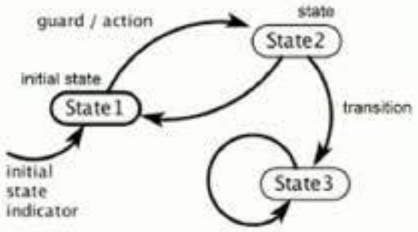
- Single-step plans
- 20/10 (train/test) demonstrations per task
- Table-top dataset gathered while teleoperating a PR2 robot
- Pretrained Mask R-CNN for segmenting the objects
- Kinect2 as RGBD sensor



Mean Absolute Error between inferred poses $\hat{\mathbf{p}}$ and commanded poses $\mathbf{p}$ during teleoperation for (a) placing on, (b) facing cups, (c) placing in. The reported error values are across 10 demonstrations (X-axis) not seen during training.

Turing Project: Safe AI for Surgical Assistance

PIs: S Ramamoorthy (Informatics @ UoE), P Brennan (NHS Lothian, UoE Clinical Brain Sciences)



Specifications :
learnt from data +
extracted from codes of practice
(Amenable to safety verification)

$$
\begin{array}{lcl}
(\mathbf{x}, t) \models \mu & \Leftrightarrow & f(x_1[t], \ldots, x_n[t]) > 0 \\
(\mathbf{x}, t) \models \varphi \wedge \psi & \Leftrightarrow & (x, t) \models \varphi \wedge (x, t) \models \psi \\
(\mathbf{x}, t) \models \neg\varphi & \Leftrightarrow & \neg((x, t) \models \varphi) \\
(\mathbf{x}, t) \models \varphi\, \mathcal{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t' \in [t+a, t+b] \text{ such that } (x, t') \models \psi \wedge \\
 & & \quad \forall t'' \in [t, t'],\ (x, t'') \models \varphi\}
\end{array}
$$

Programming
by discussion &
model induction

Guaranteed
control synthesis
at Levels 3-4

Surfing on an uncertain edge: Precision cutting of soft tissue using torque-based medium classification

Artūras Straižys, Michael Burke
and Subramanian Ramamoorthy

Robust Autonomy and Decisions Group
University of Edinburgh
http://rad.inf.ed.ac.uk

We learn probabilistic classification of sensor readings associated with knife's operation in either peel or pulp region

and construct a control scheme in which the movement is corrected according to the probability of the knife being inserted into either of the two mediums.


PR2
Online movement correction (closed-loop)

Concluding Remarks

- Most robotic systems are hybrid control systems
  - Modelling them should involve careful combination of compositional reasoning that drives sensorimotor control
- Programmatic representations, carefully combined with flexible probabilistic predicates, are useful in this way
- We propose novel tools for learning from sensorimotor experience using such representations

- Such representations also enable connections to other tools for:
  - Analysis of causality
  - Guided sampling and analysis of coverage
  - Verification of behavioural properties