Carnegie Mellon

School of Computer Science

Deep Reinforcement Learning and Control

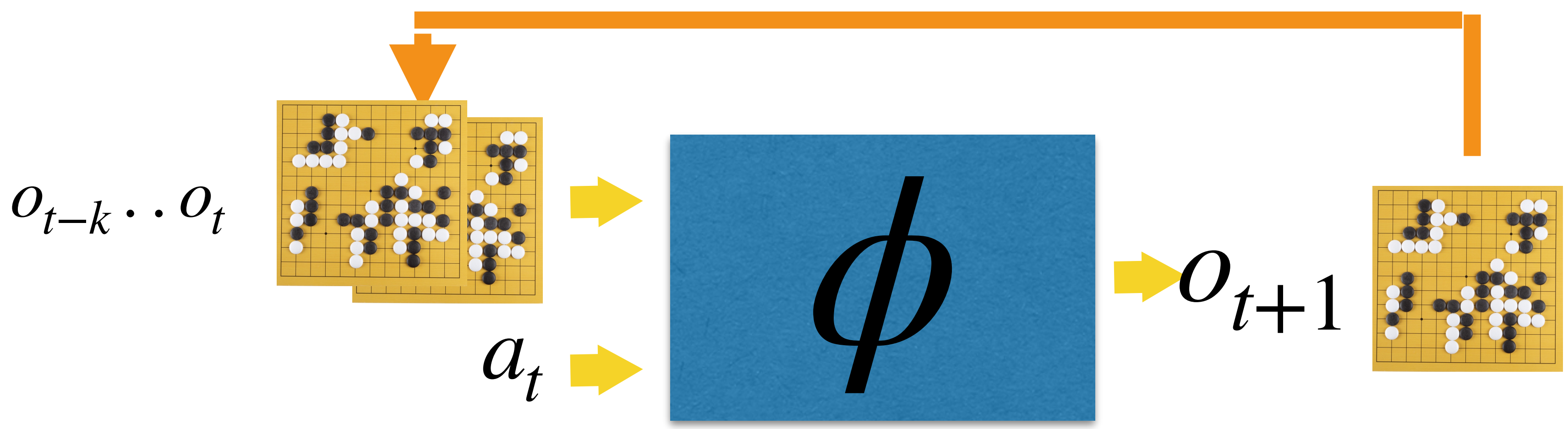
MBRL (cont.): Holistic and graph-based world models

Fall 2021, CMU 10-703

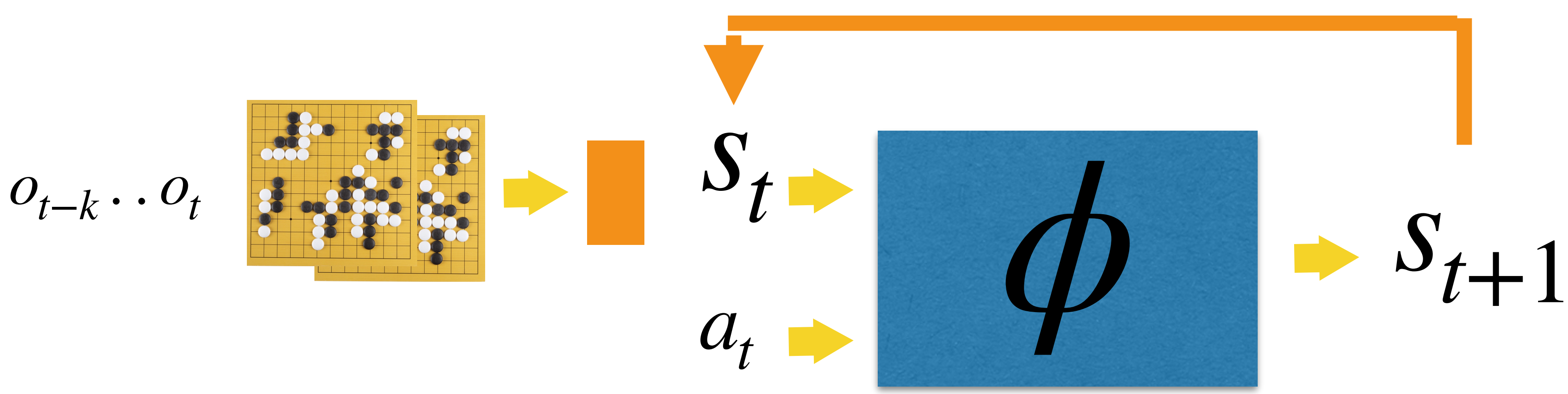
Instructors
Katerina Fragkiadaki
Ruslan Salakhutdinov

Model learning from sensory input

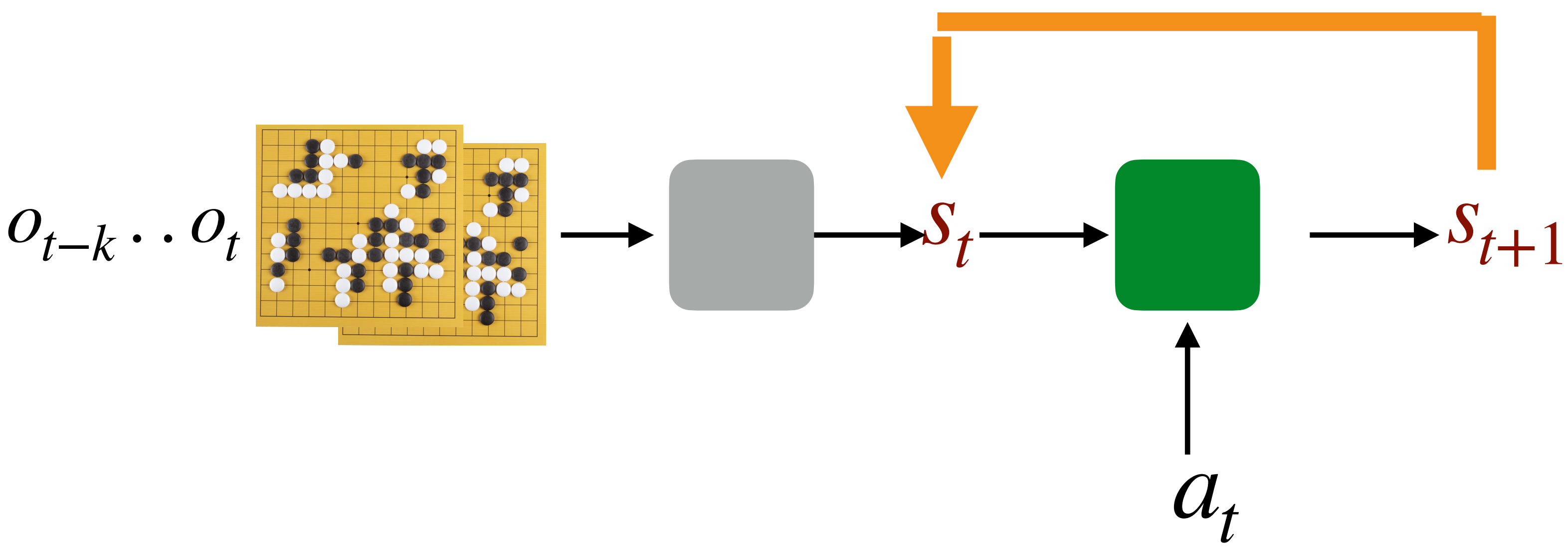
Unrolling in the observation space:



Unrolling in a latent space:



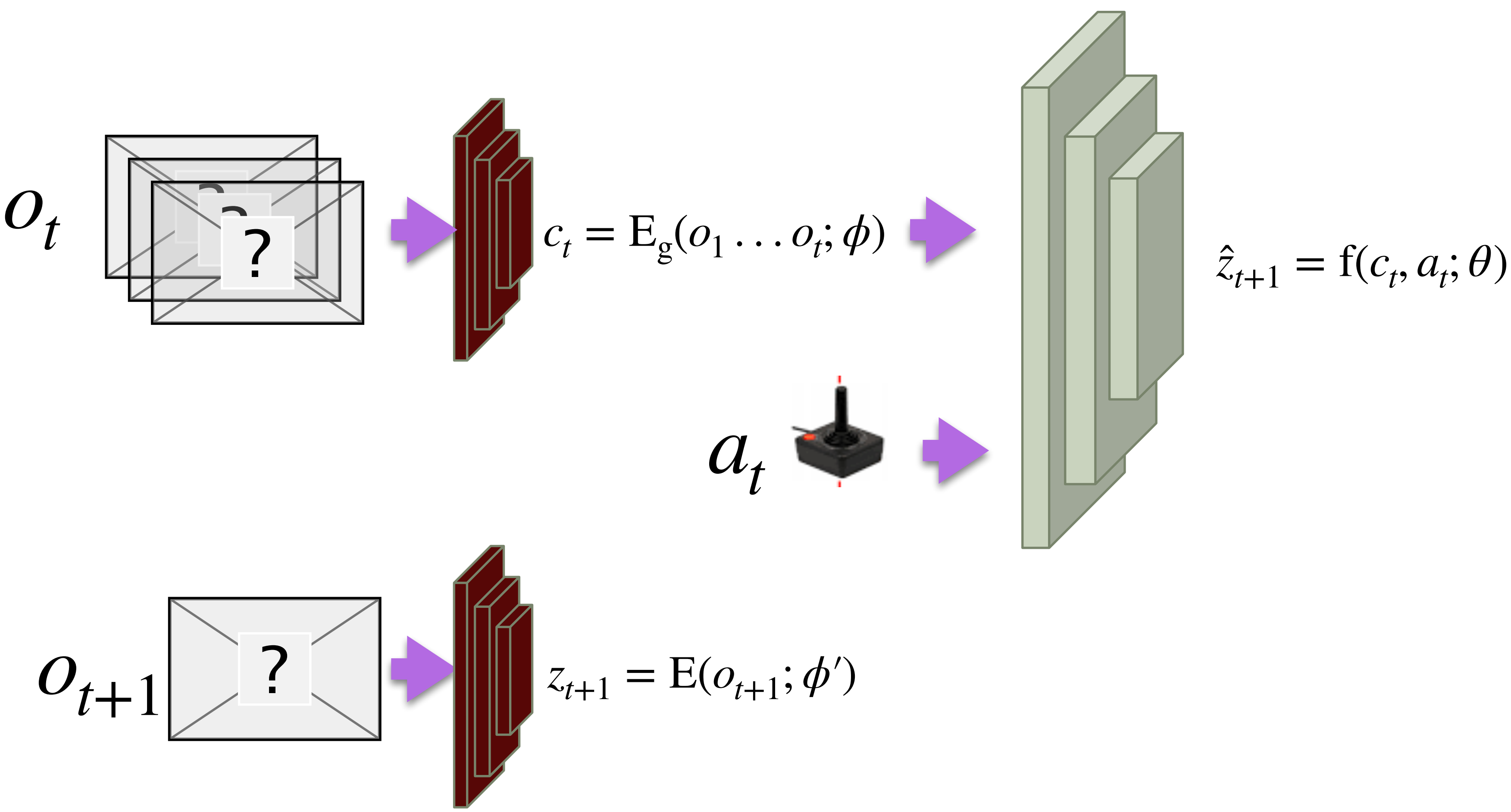
Unrolling in a latent state space



Deterministic transition model: $s_{t+1} = g(s_t, a_t)$

MuZero learns such observation to latent space mapping by considering value functions and policies under a specific reward function.

Prediction in a latent space



$$\min_{\theta,\phi,\phi'} . \quad \|\mathrm{f}(c_t, a_t; \theta) - z_{t+1}\|$$

Q:What is the problem with this optimization problem?

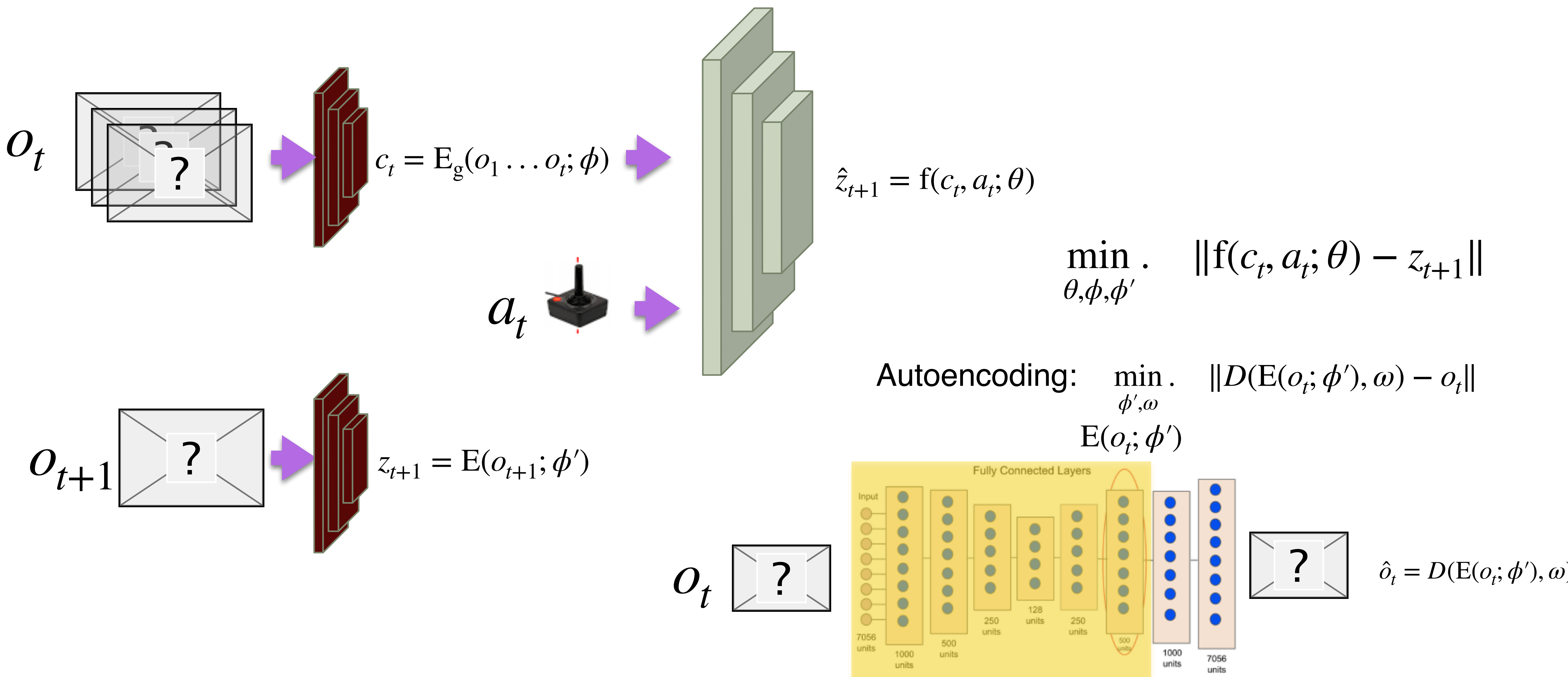
A:There is a trivial solution :-(

Q:Would the problem go away if instead we had: $\hat{z}_{t+1} = z_t + \mathrm{f}(c_t, a_t; \theta)$

A:No, it’s exactly the same problem.

We need to predict additional information from the encodings to avoid the trivial solution

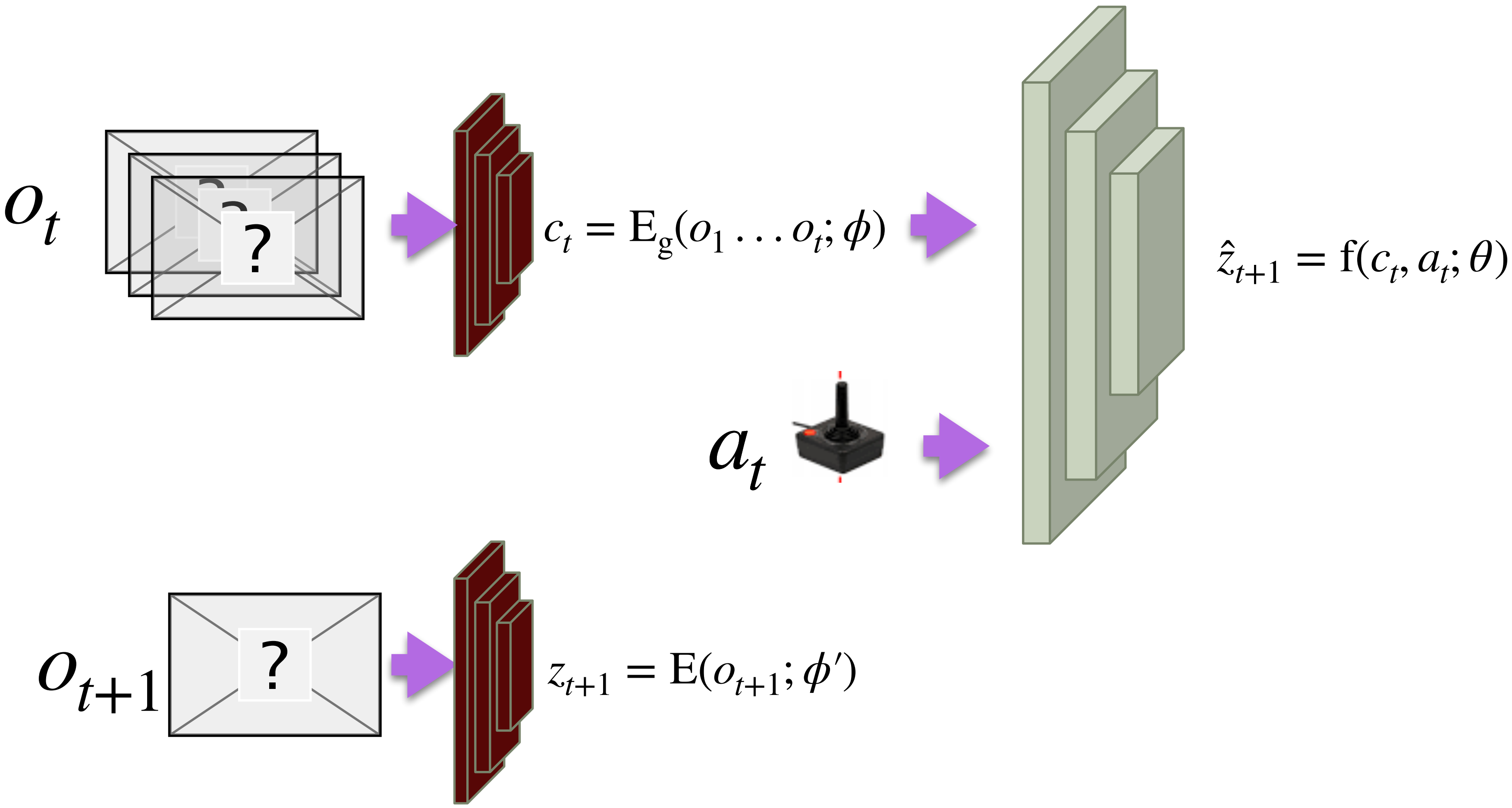
Prediction in a latent space - autoencoding



- Predict the image from the latent encoding
- …and suffer the problems of autoencoding reconstruction loss that has little to do with our task

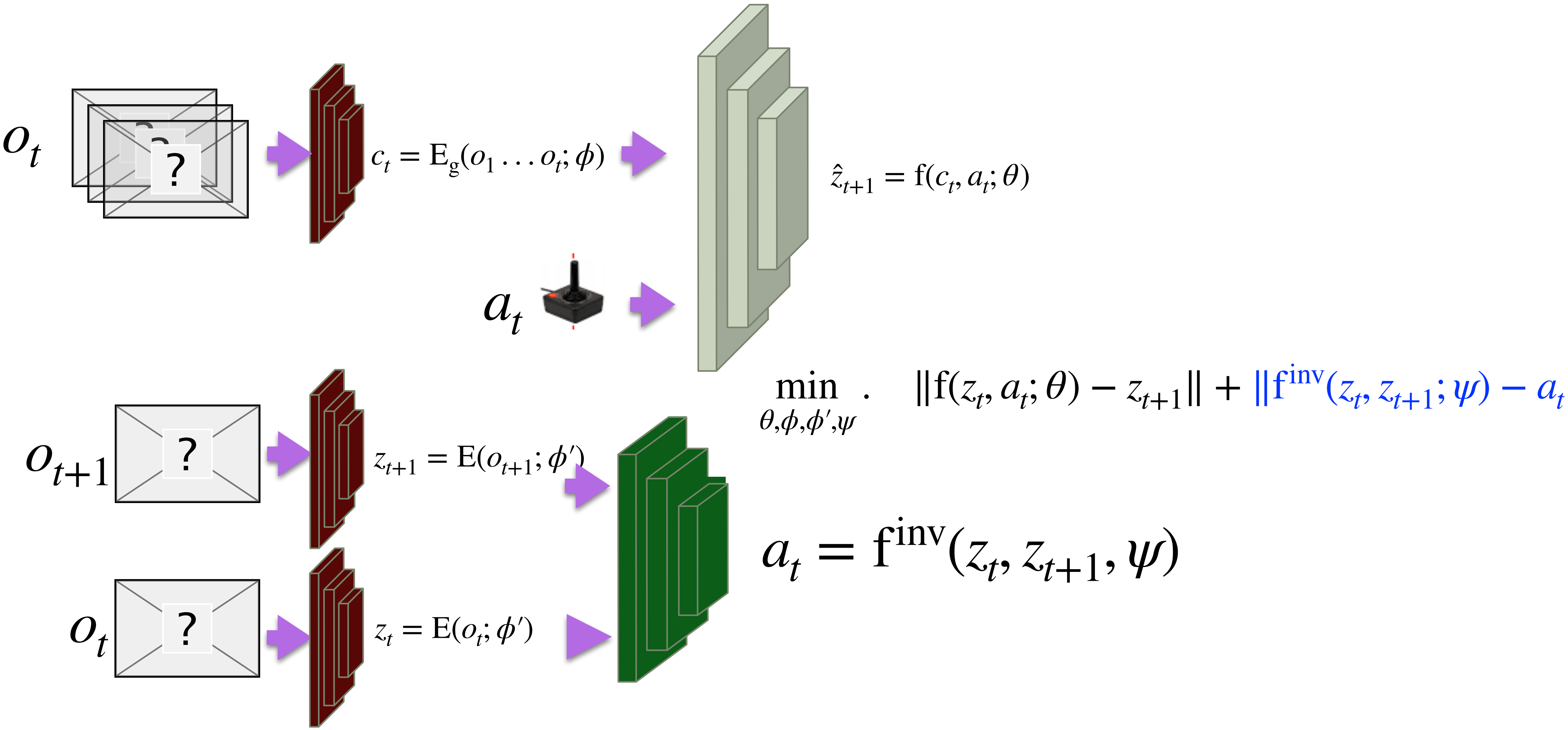
Incentivizing exploration in RL with deep predictive models, Stadie et al.

Prediction in a latent space



$$\min_{\theta,\phi,\phi'} . \quad \|\mathrm{f}(c_t, a_t; \theta) - \hat{z}_{t+1}\|$$

Prediction in a latent space - inverse models

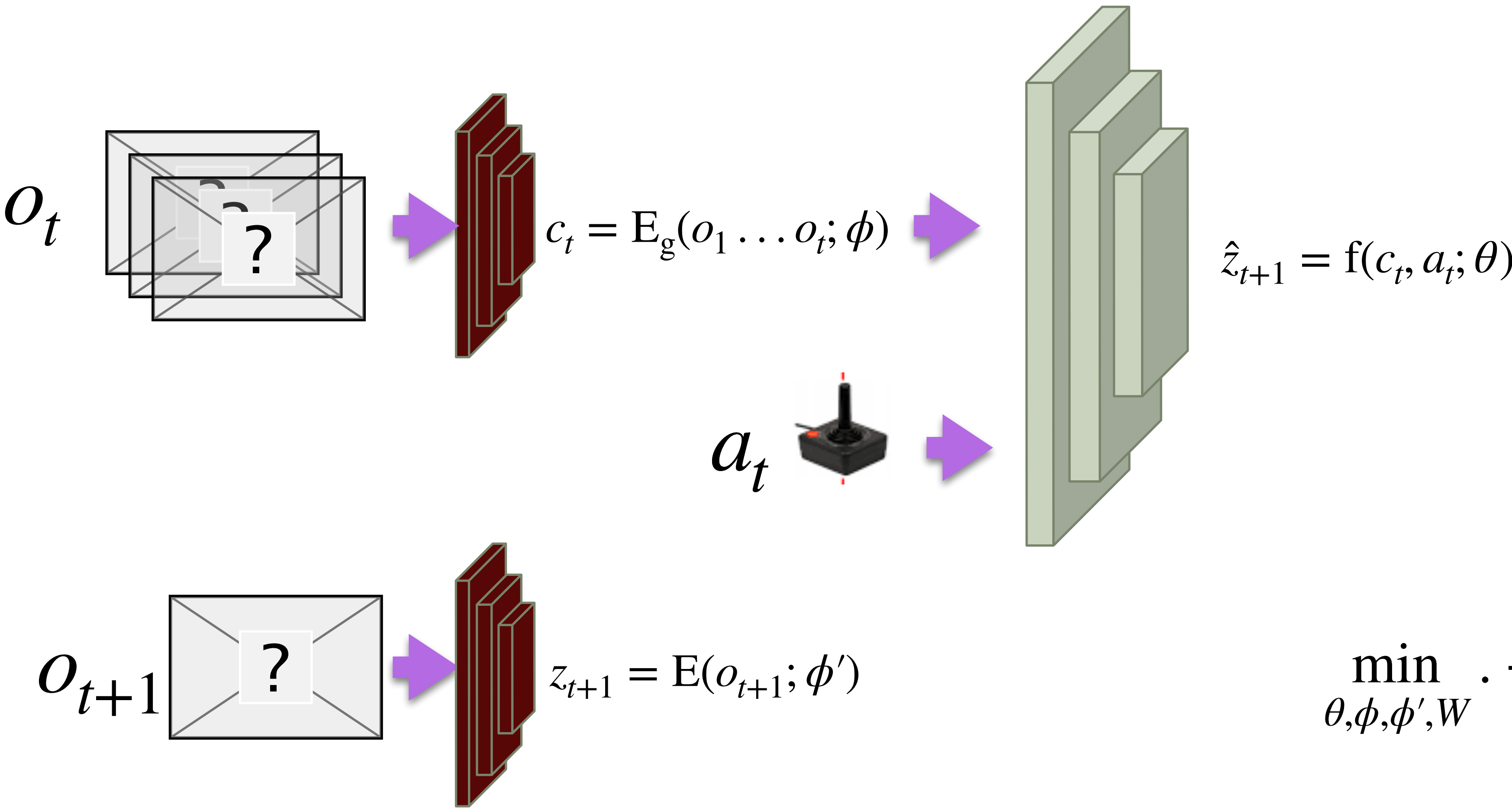


- Let’s couple forward and inverse models (to avoid the trivial solution)
- …then we will only predict things that the agent can control

Learning to poke by poking, *Agrawal et al. 2016*

Prediction in a latent space - contrastive prediction

$$P(d = i \mid c) = \frac{\exp(c^\top W z^i)}{\sum_j \exp(c^\top W z^j)}$$

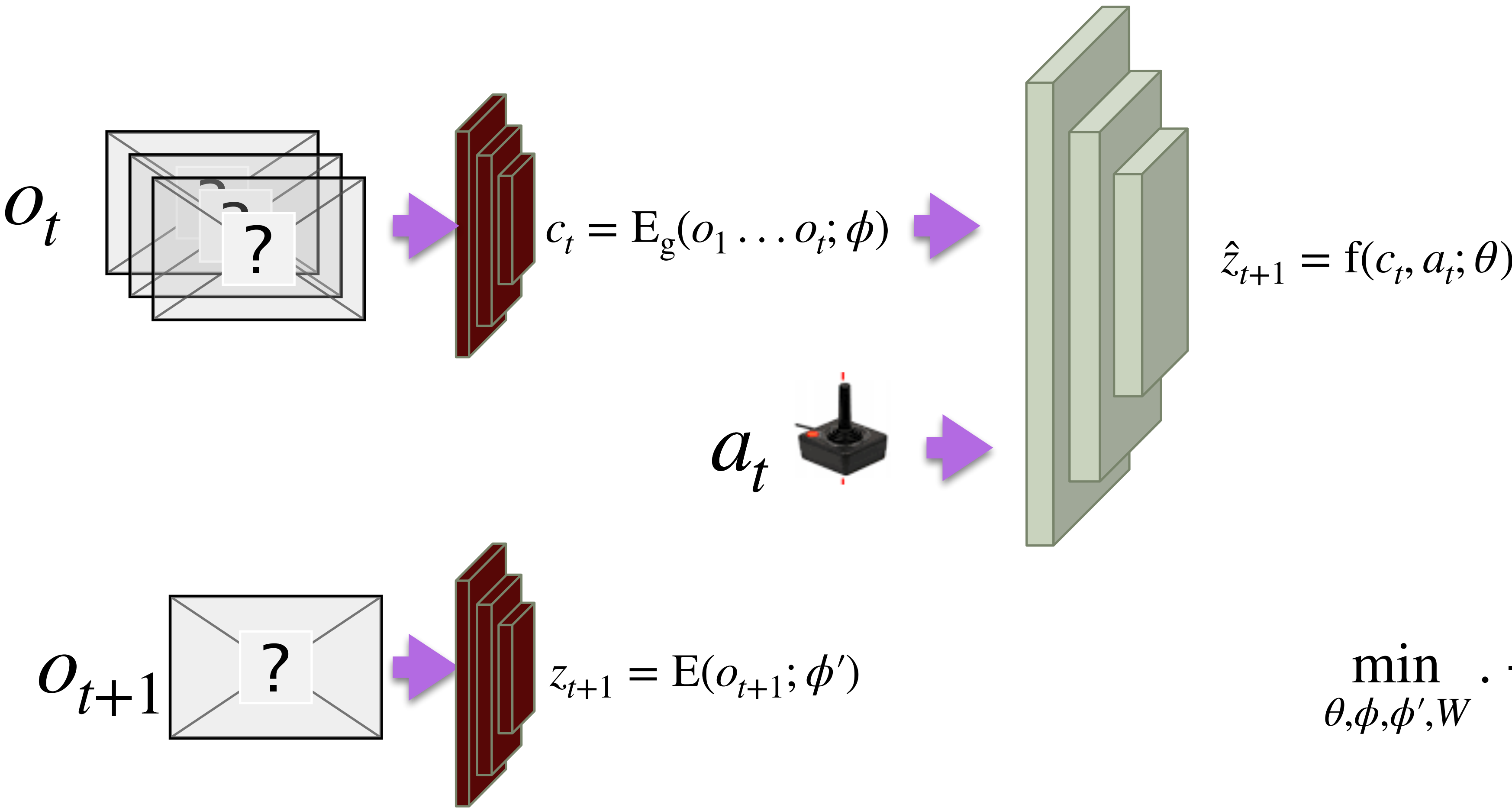


$$\min_{\theta,\phi,\phi',W} . -\log \frac{\exp(\hat{z}_{t+1}{}^\top W z_{t+1})}{\sum_{j \in Neg} \exp(\hat{z}_{t+1}{}^\top W z^j)}$$

- Generative: model the distribution of future observations/embeddings
- Discriminative: model how much closer you can match the future observations than other alternatives.
- Imagine we could discretize all the possible future: then we would just need to predict the right probability distribution over all (discrete set of) possibilities. Then, we want to maximize the probability of the correct outcome.

Prediction in a latent space - contrastive prediction

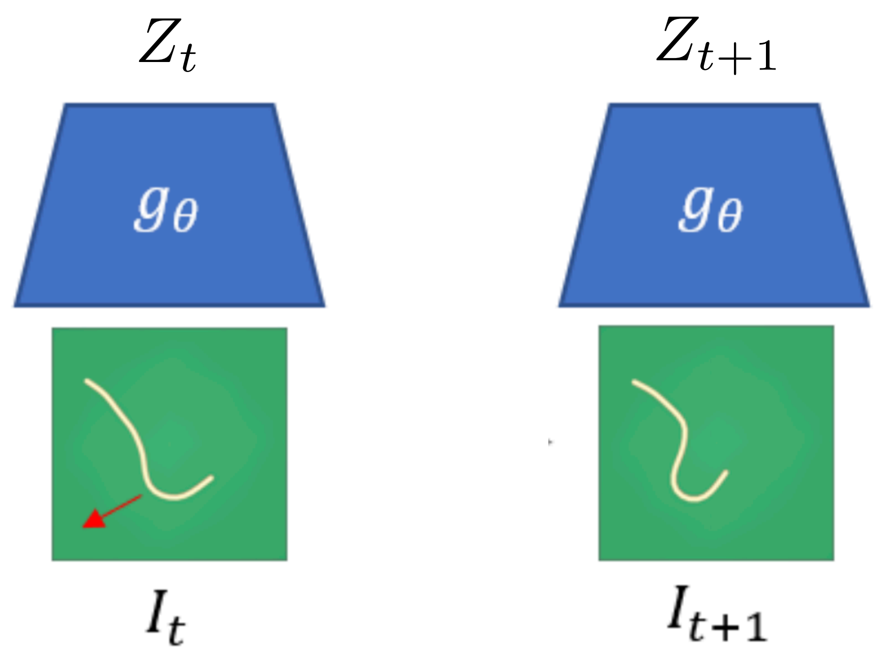
$$P(d = i \mid c) = \frac{\exp(c^\top W z^i)}{\sum_j \exp(c^\top W z^j)}$$



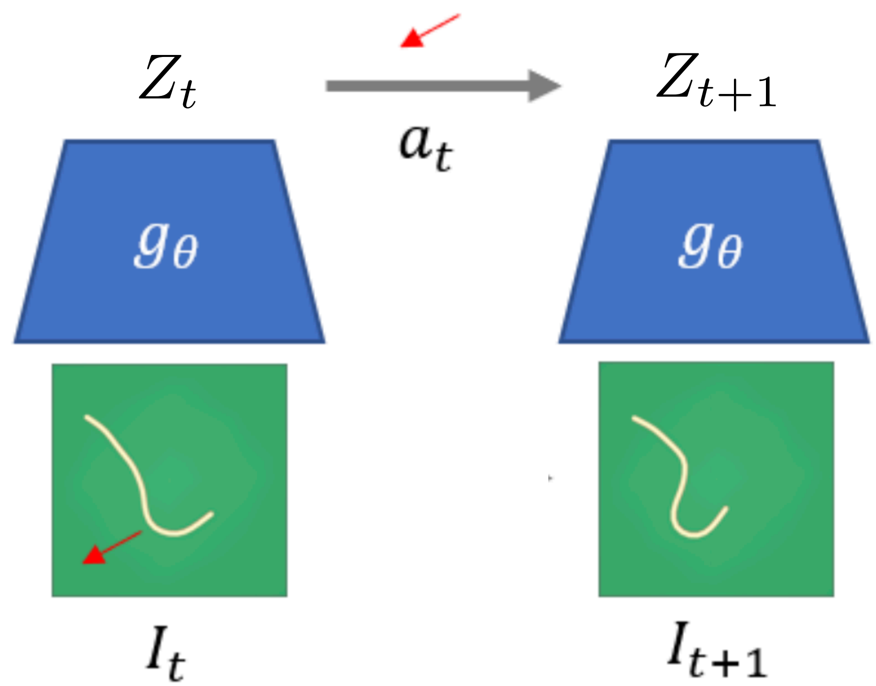
$$\min_{\theta,\phi,\phi',W} . - \log \frac{\exp(\hat{z}_{t+1}{}^\top W z_{t+1})}{\sum_{j \in Neg} \exp(\hat{z}_{t+1}{}^\top W z^j)}$$

- Q:Since we do not directly predict the future z_{t+1}, how can we unroll this model forward in time?
- A: Through ranking. Consider a set of possibilities and rank them

Contrastive forward models for deformable object manipulation



Contrastive forward models for deformable object manipulation



Contrastive forward models for deformable object manipulation

Contrastive loss:

$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}}\left[\log \frac{h(\hat{z}, z_{\text{pos}})}{\sum_{i=1}^{k} h(\hat{z}, z_{\text{neg}})}\right]$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation

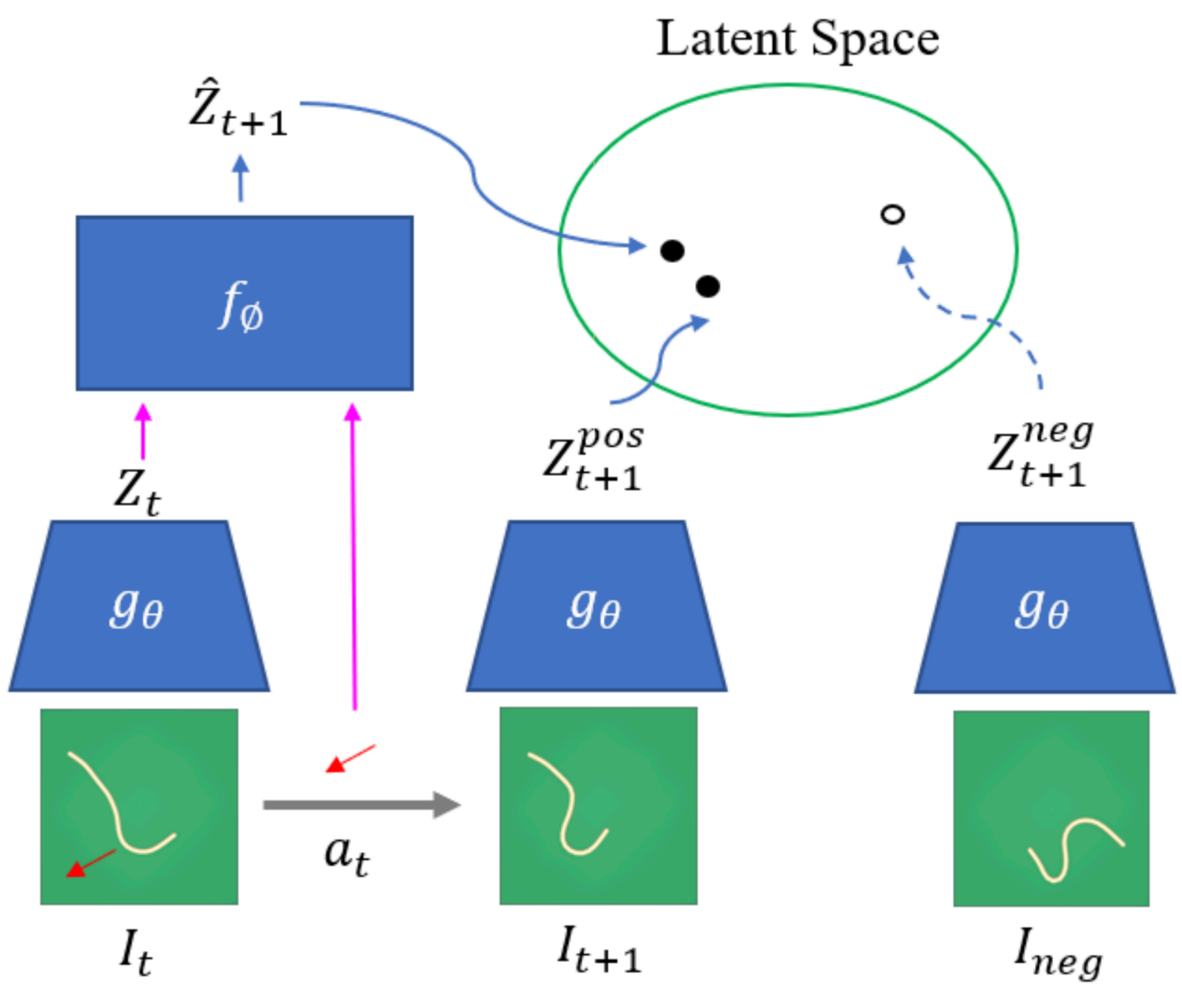
Contrastive loss:

$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}}\left[\log \frac{h(\hat{z}, z_{\text{pos}})}{\sum_{i=1}^{k} h(\hat{z}, z_{\text{neg}})}\right]$$

Similarity function:

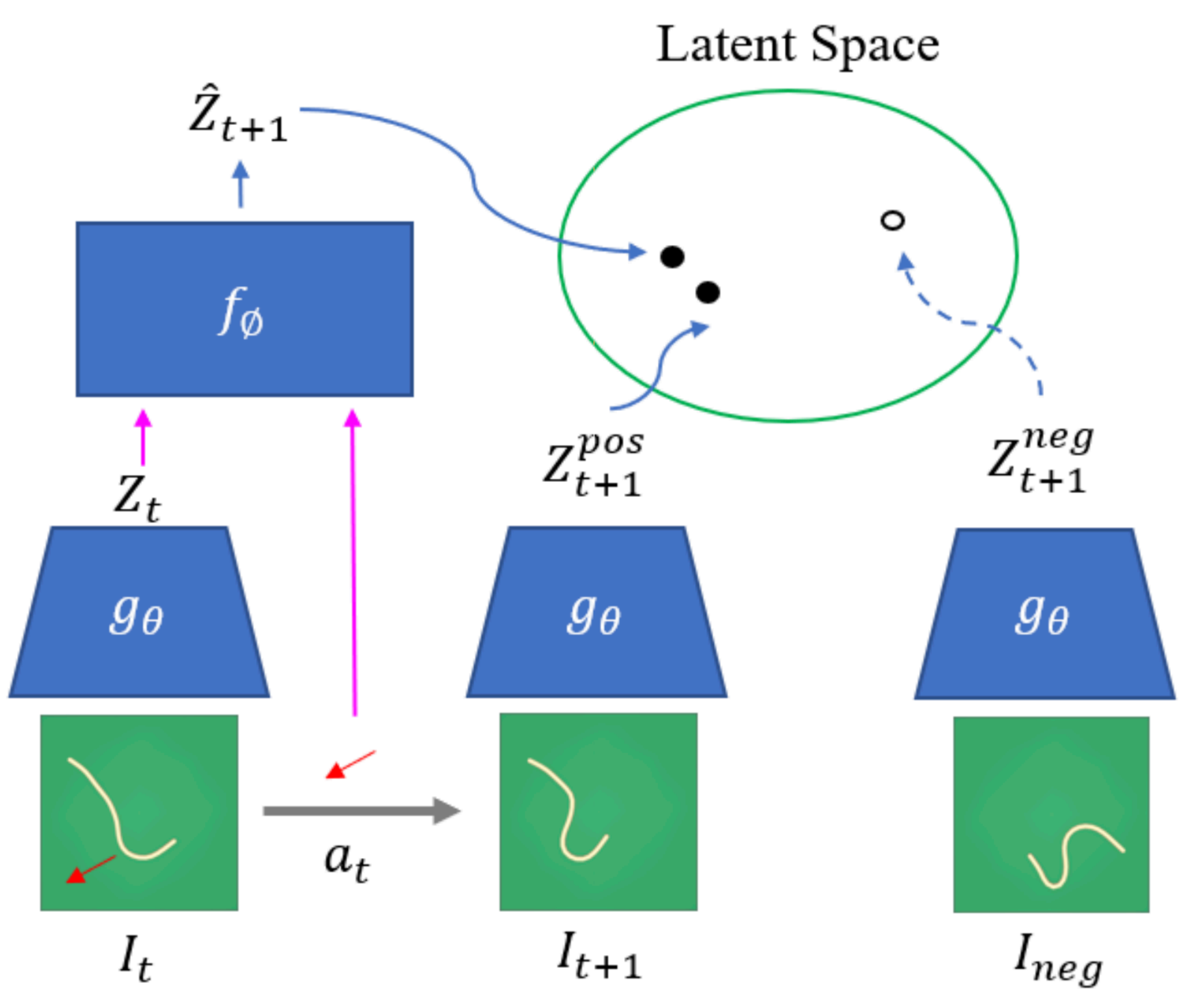
$$h(z_1, z_2) = \exp(z_1^T z_2)$$

Contrastive forward models for deformable object manipulation



Learning Predictive Representations for Deformable Objects using Contrastive Estimation

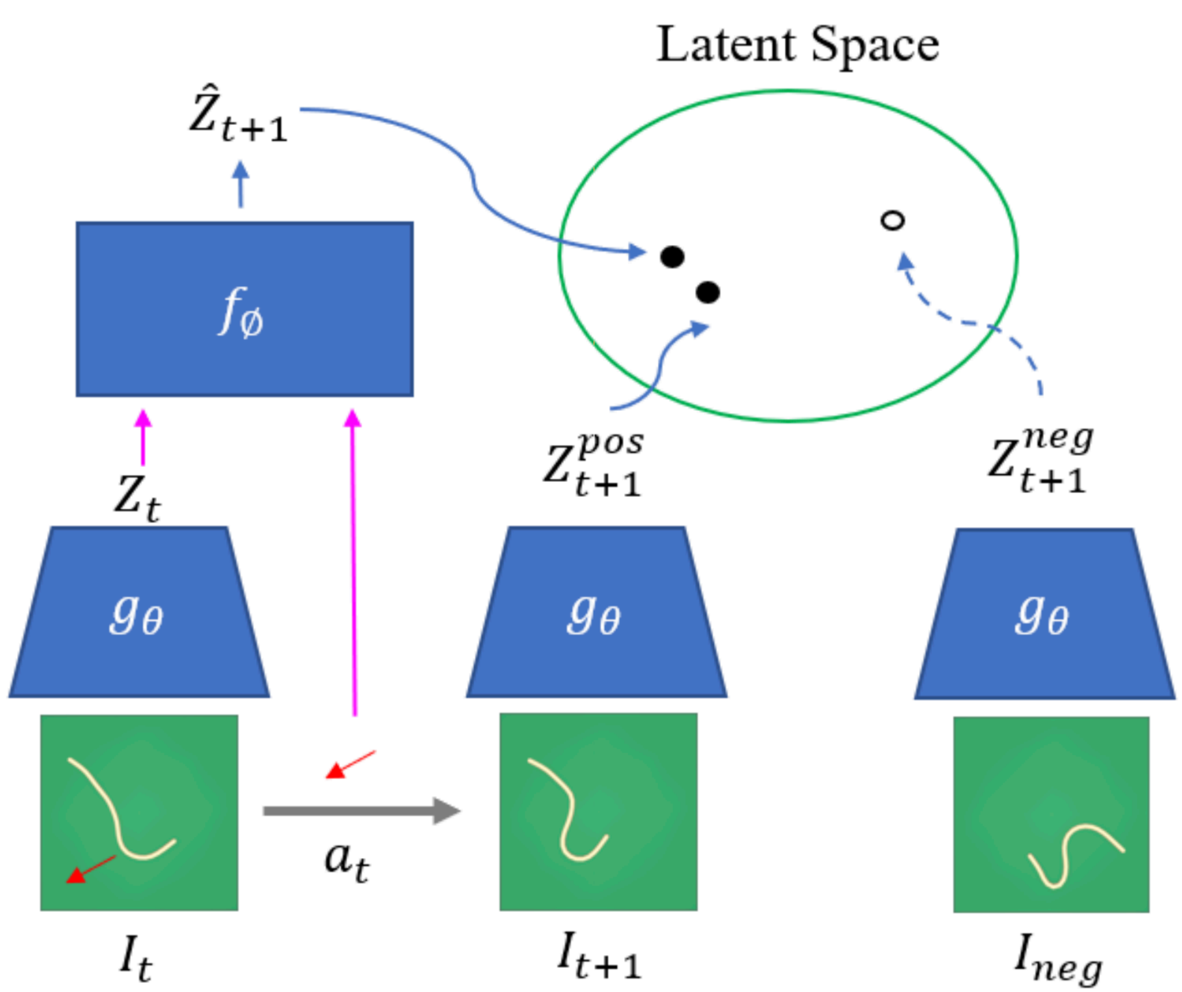
Contrastive forward models for deformable object manipulation



$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}}\left[\log \frac{h(\hat{z}_{t+1}, z_{t+1})}{\sum_{i=1}^{k} h(\hat{z}_{t+1}, \tilde{z}_i)}\right]$$

Learning Predictive Representations for Deformable Objects using Contrastive Estimation

Contrastive forward models for deformable object manipulation

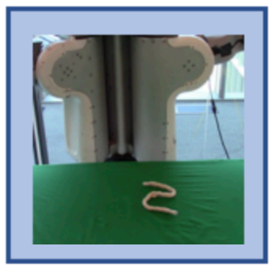


$$\mathcal{L} = -\mathbb{E}_{\mathcal{D}}\left[\log \frac{h(\hat{z}_{t+1}, z_{t+1})}{\sum_{i=1}^{k} h(\hat{z}_{t+1}, \tilde{z}_i)}\right]$$

$$h(z_1, z_2) = \exp(-\left\|z_1 - z_2\right\|^2)$$

One step Model-Predictive Control

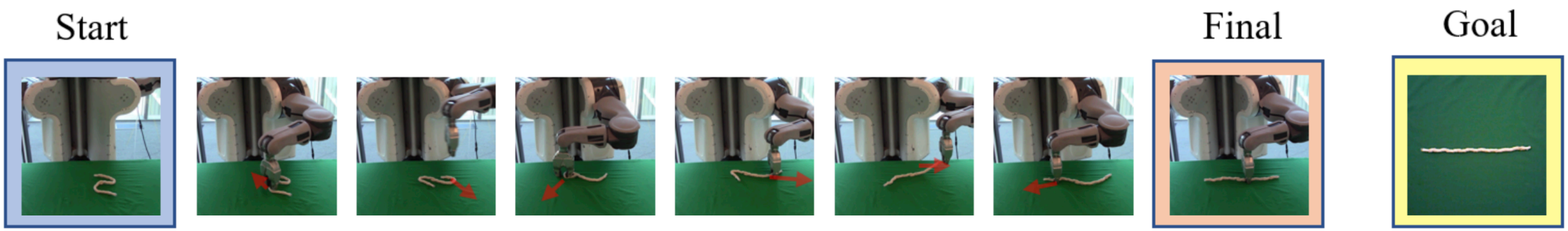
Start



Goal

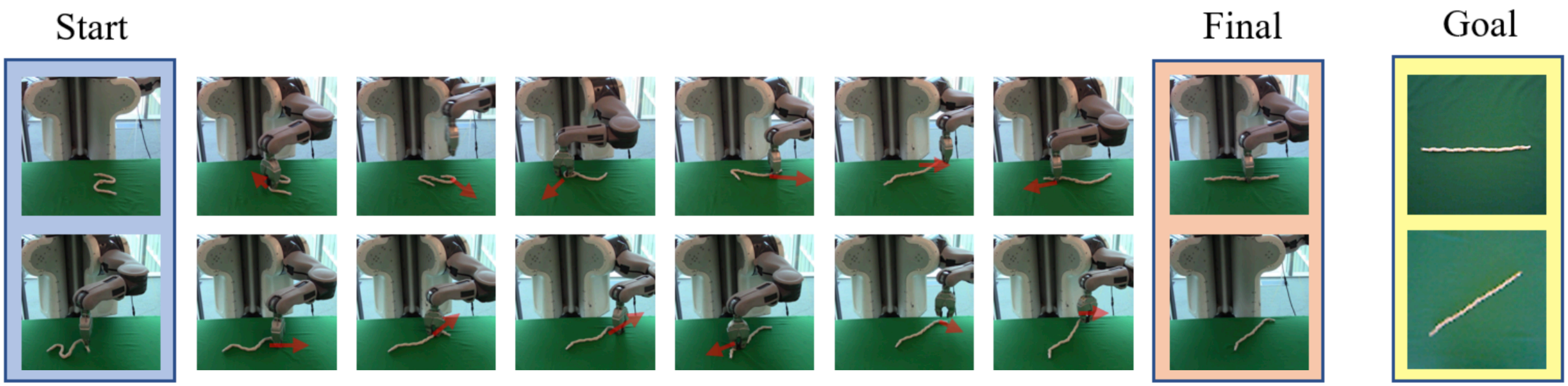
$$a_t = \max h(f_\phi(z_t, a), z_g)$$

One step Model-Predictive Control



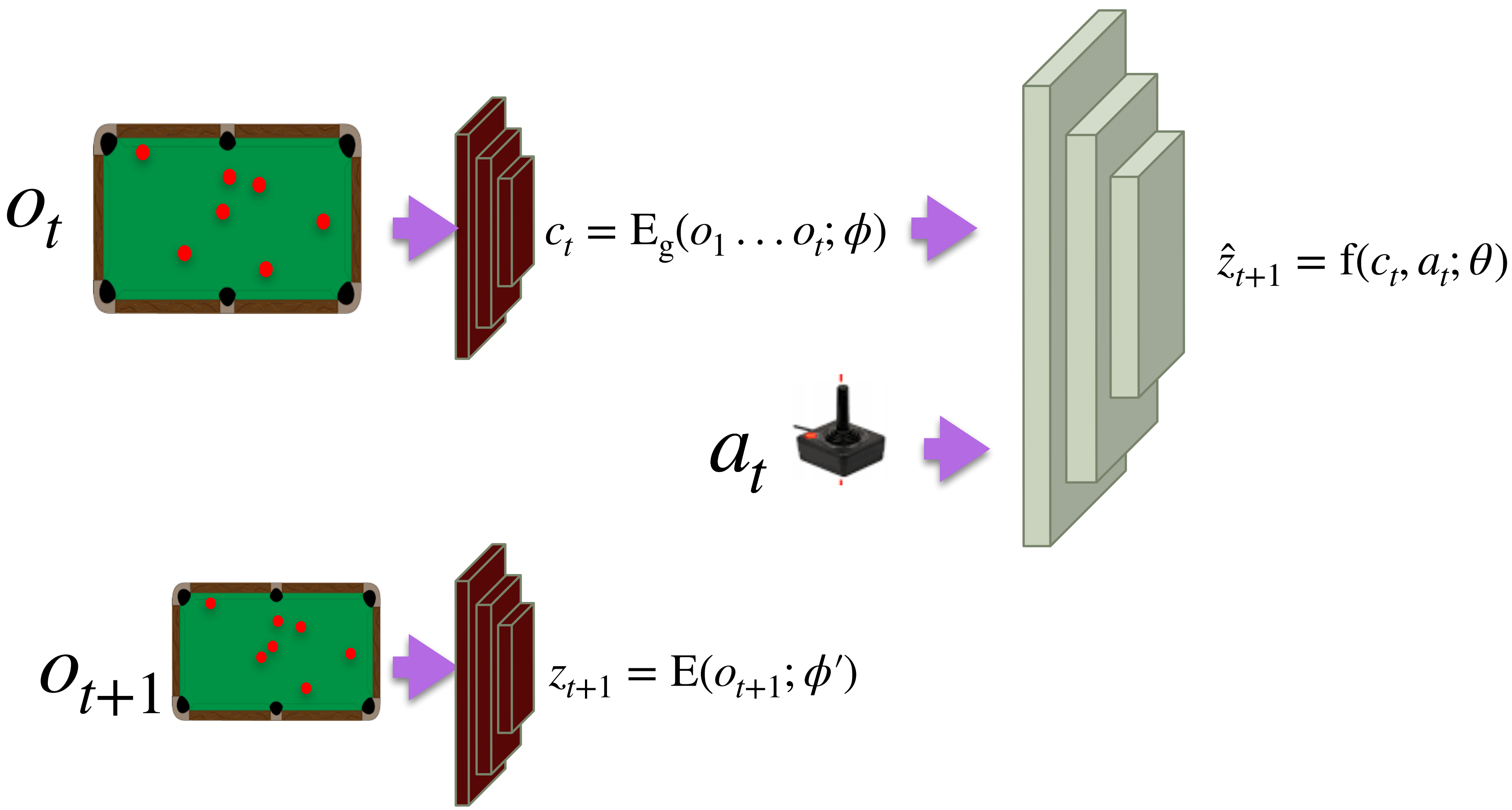
$$a_t = \max h(f_\phi(z_t, a), z_g)$$

One step Model-Predictive Control



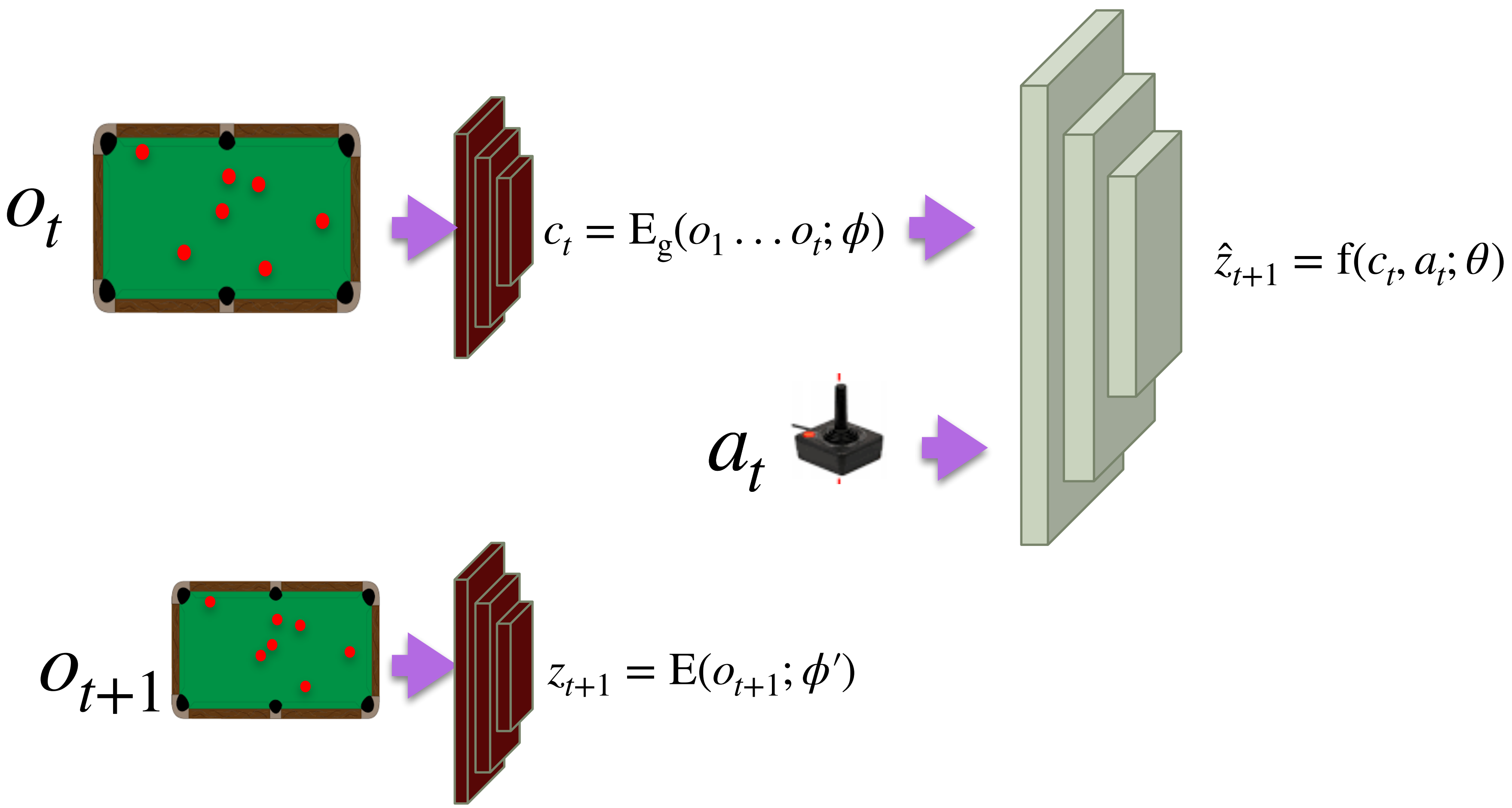
One step Model-Predictive Control

Problem with holistic models



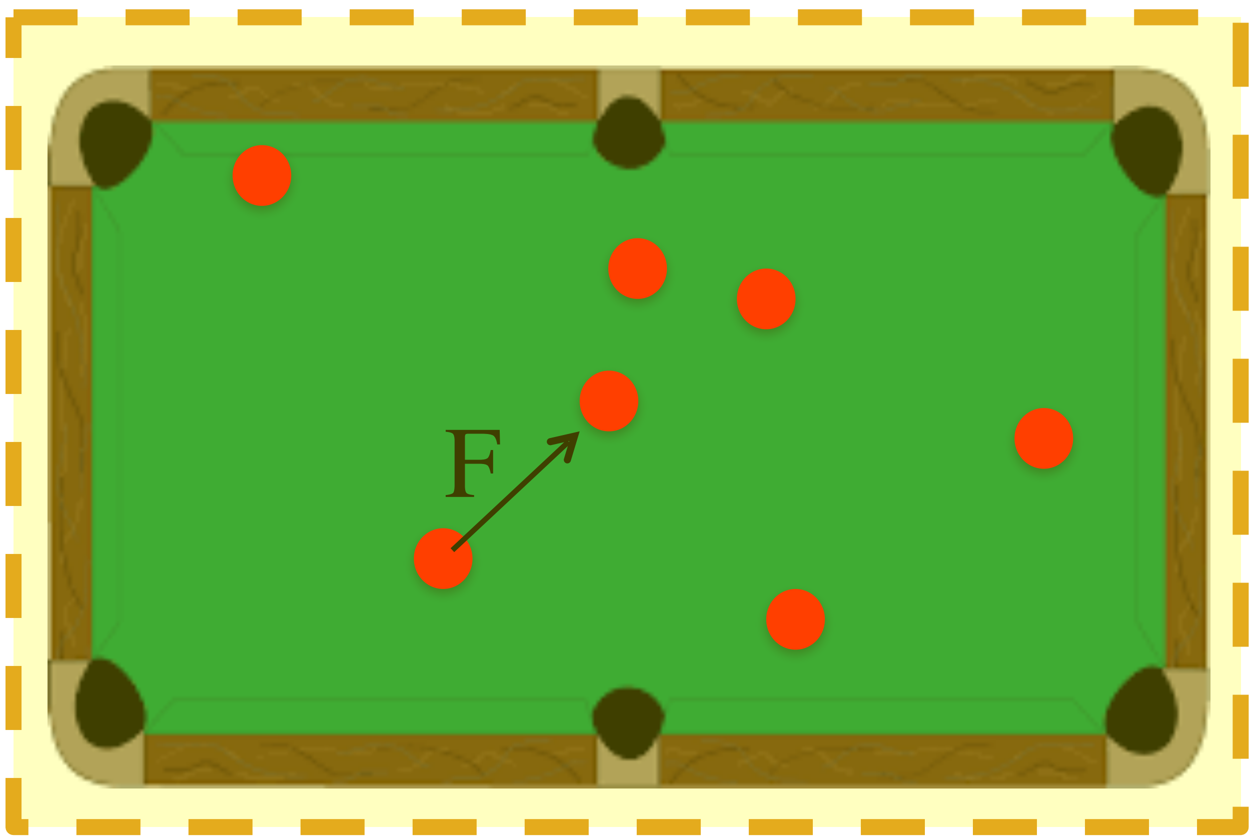
- The whole image is mapped to one vector, and the dynamics of that single vector are predicted over time.
- This means all objects together are predicted, and we do not exploit causality constraints: that objects often move independently!
- By making our representations causal and disentangled enough, we have the hope of generalization. If we entangle, we cannot generalize beyond training conditions.

Problem with holistic models



Q: Will our model be able to generalize across different number of balls present?

Frame-centric models

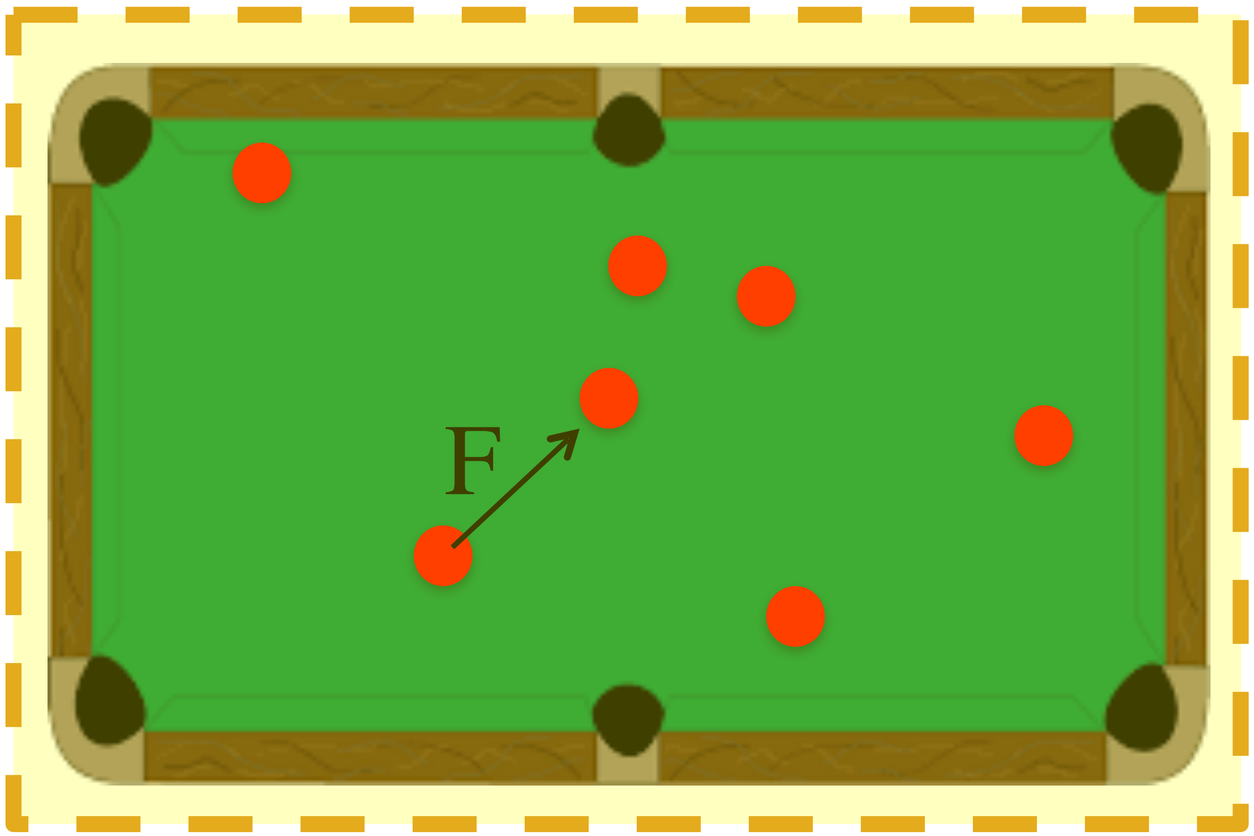


Frame-Centric Prediction

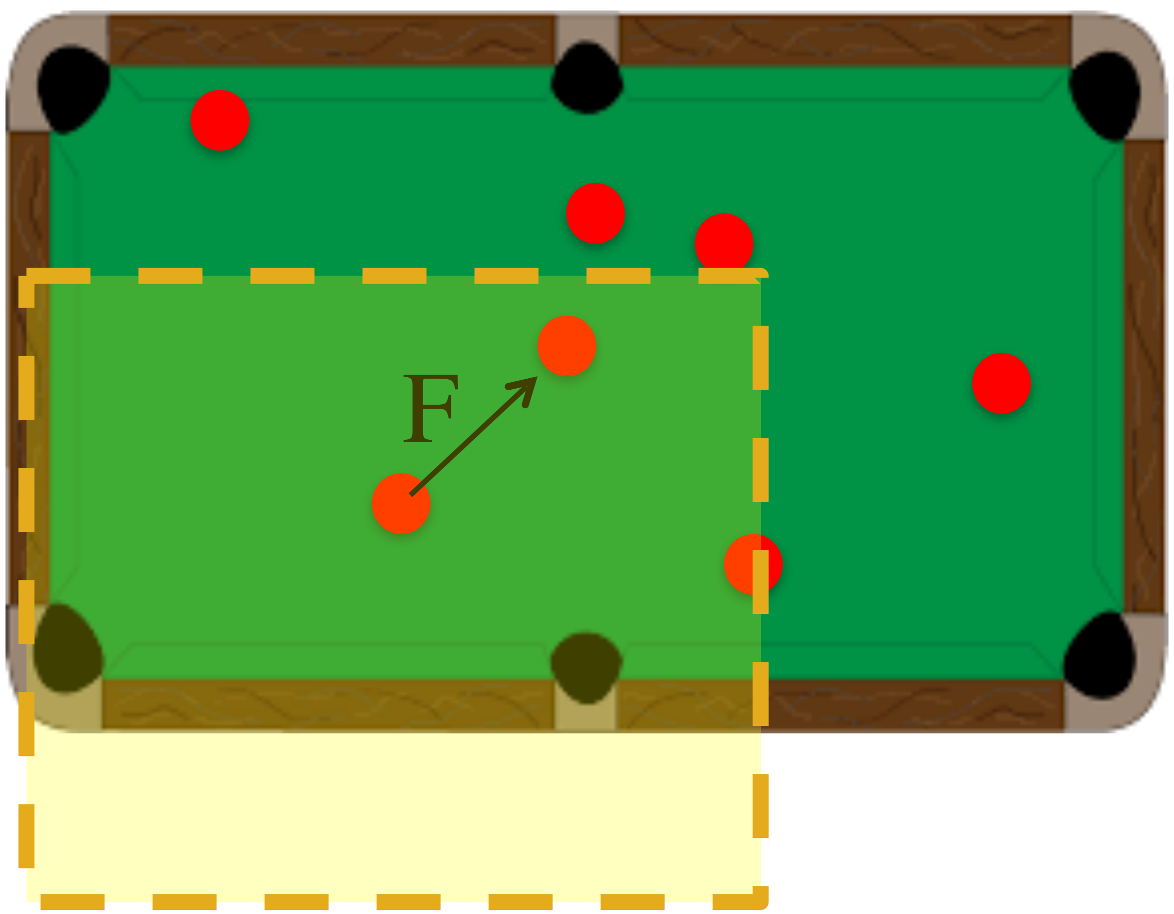
Q: Will our model be able to generalize across different number of balls present?

Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Entity-centric models



World-Centric Prediction

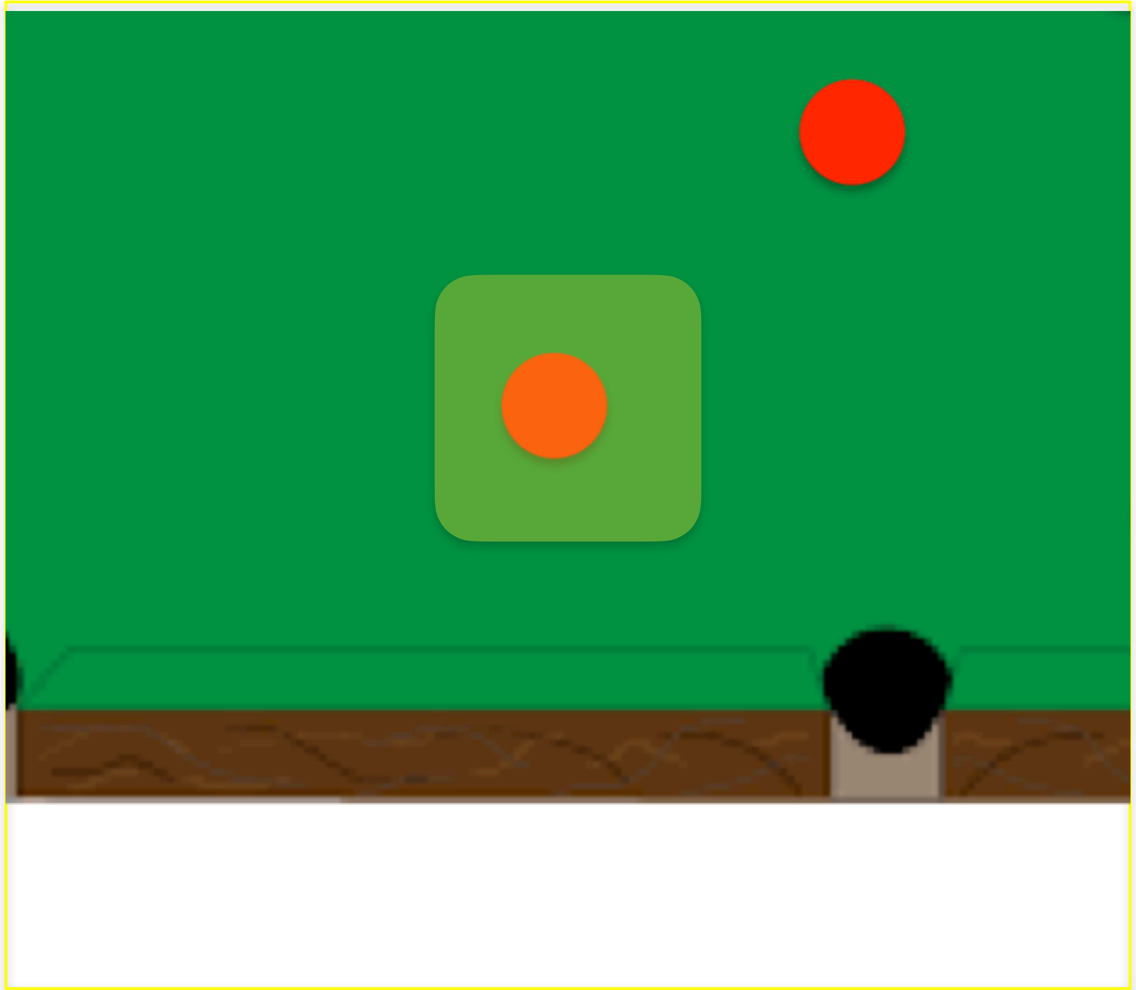


Object-Centric Prediction

The object-centric model will be applied to each object in the scene

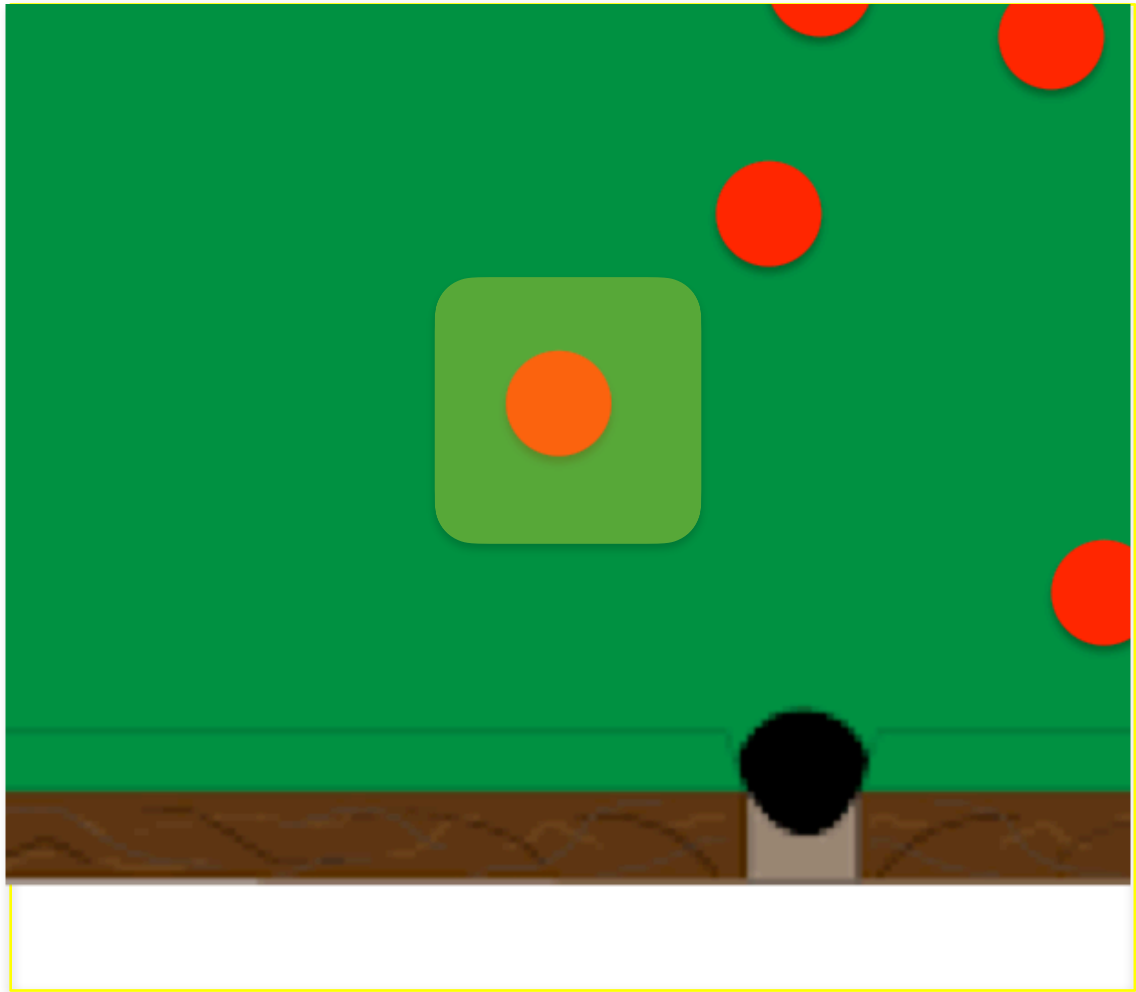
Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Context around the object is captured by using large windows around the object of interest



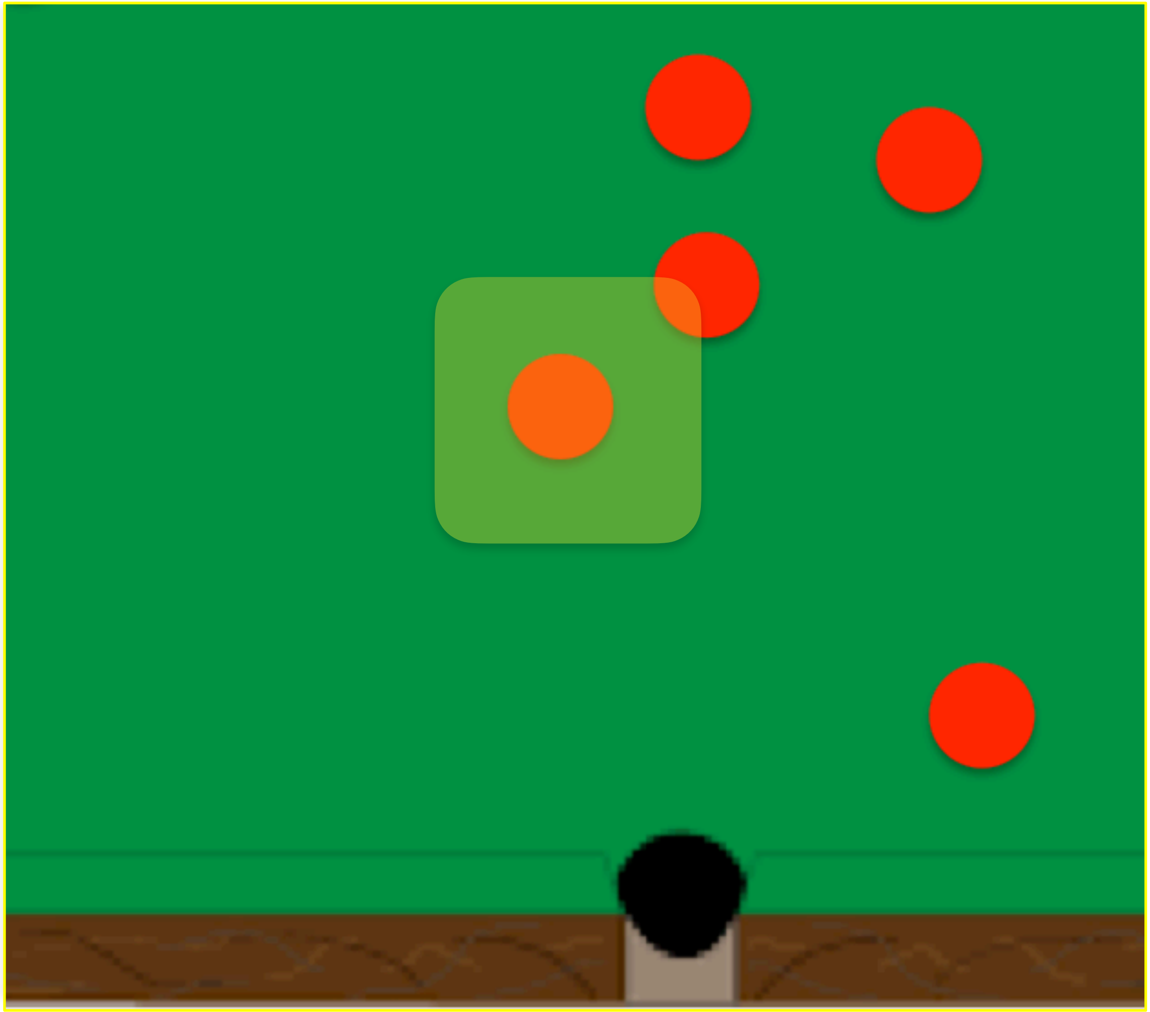
Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Context around the object is captured by using large windows around the object of interest



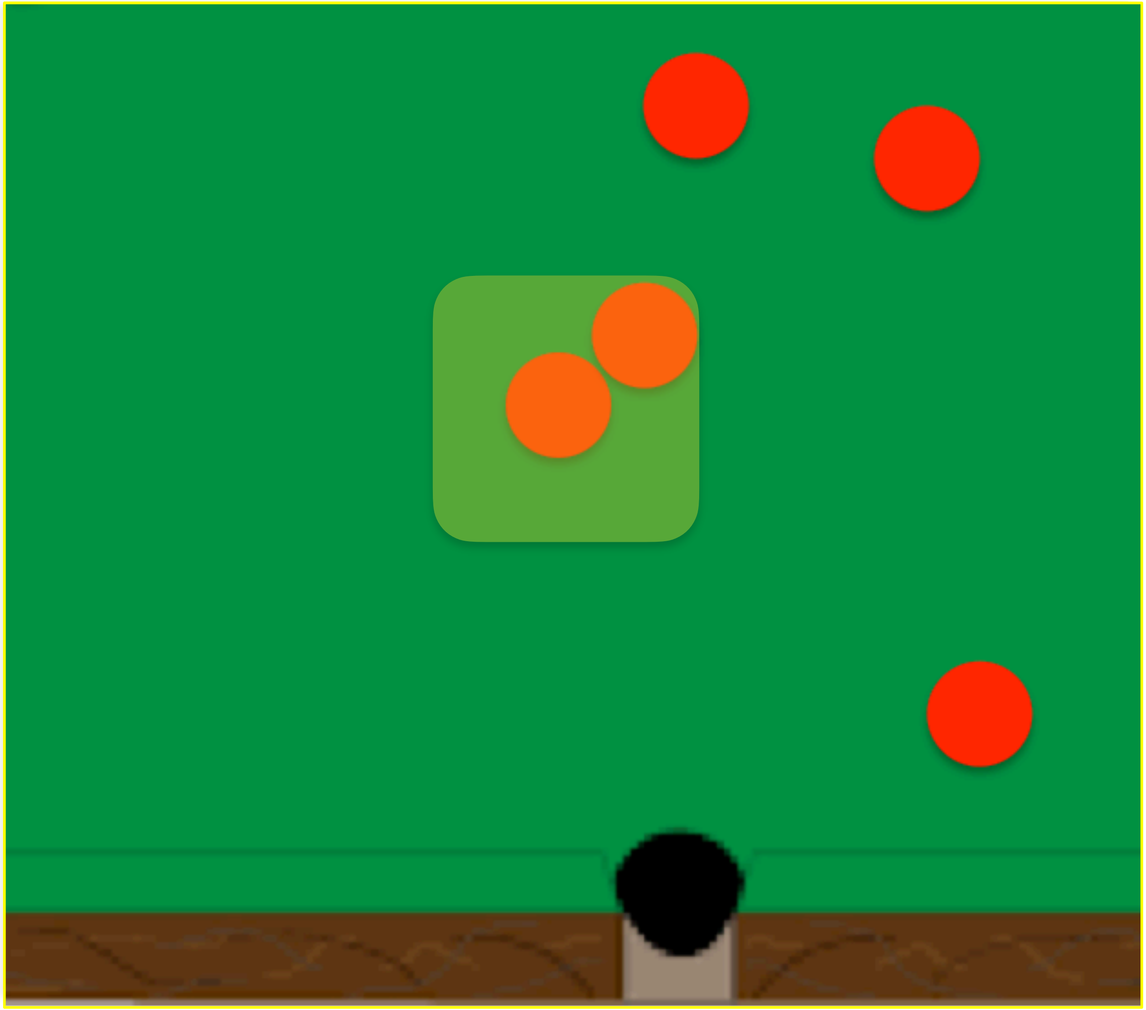
Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Context around the object is captured by using large windows around the object of interest



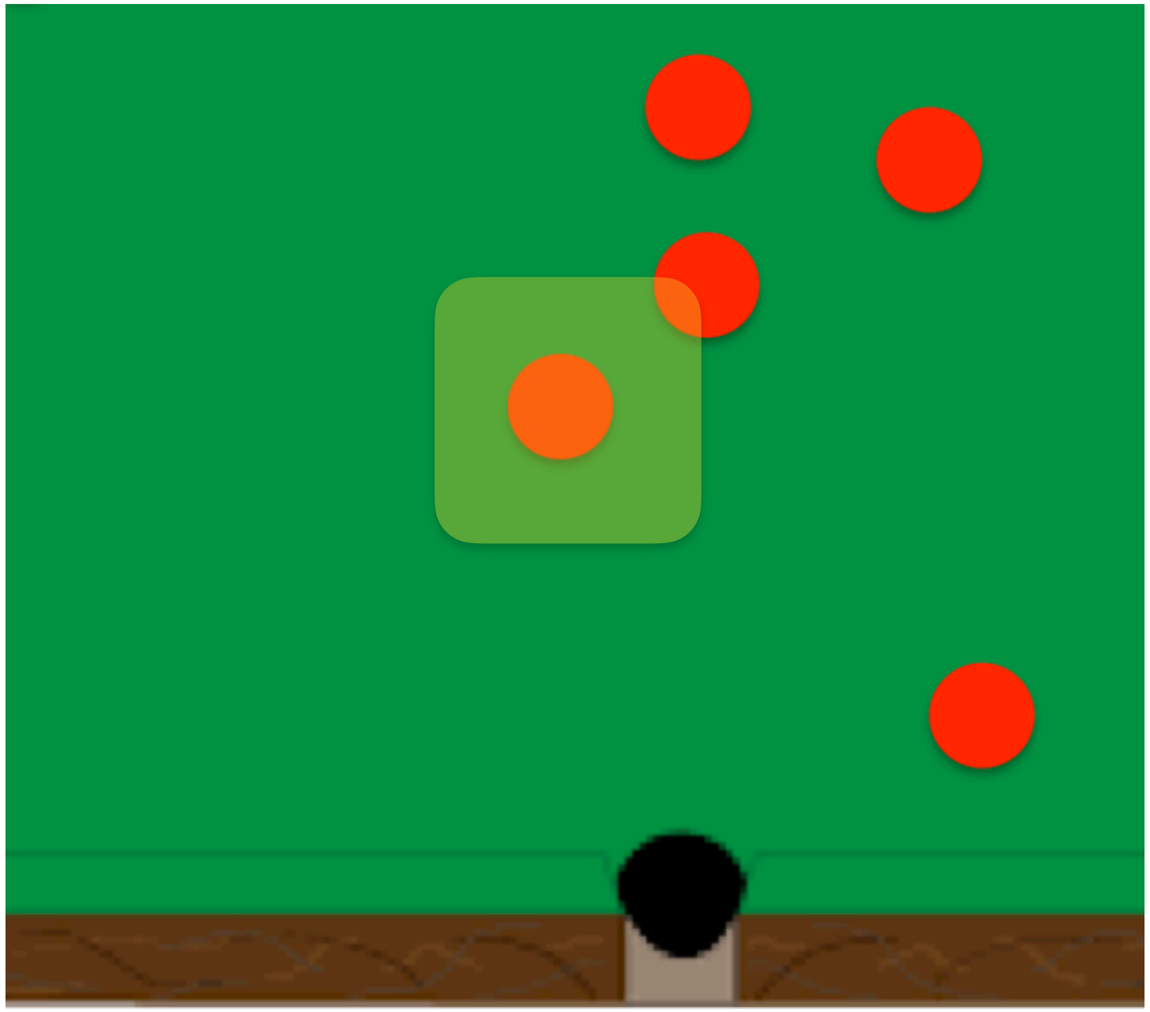
Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Context around the object is captured by using large windows around the object of interest



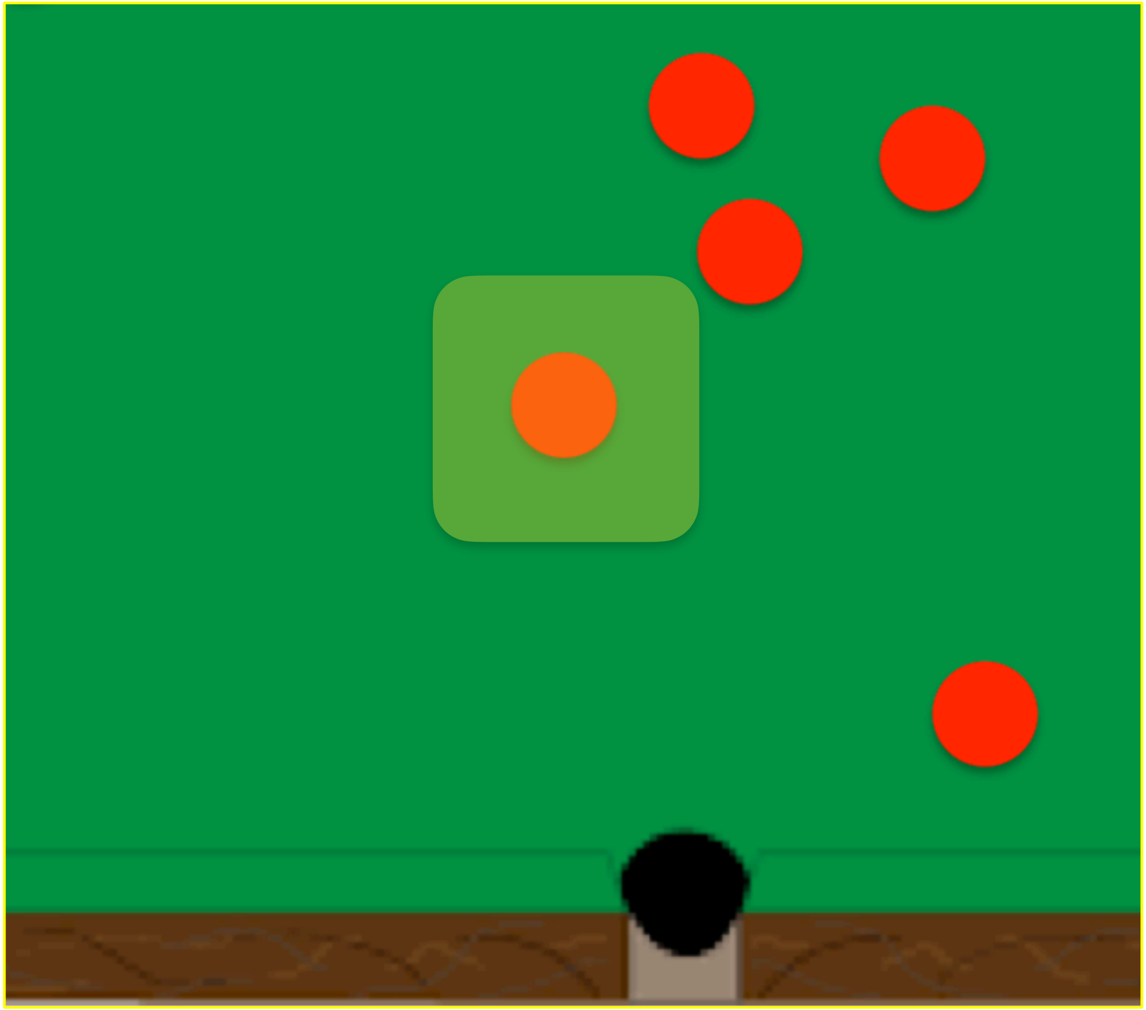
Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Context around the object is captured by using large windows around the object of interest



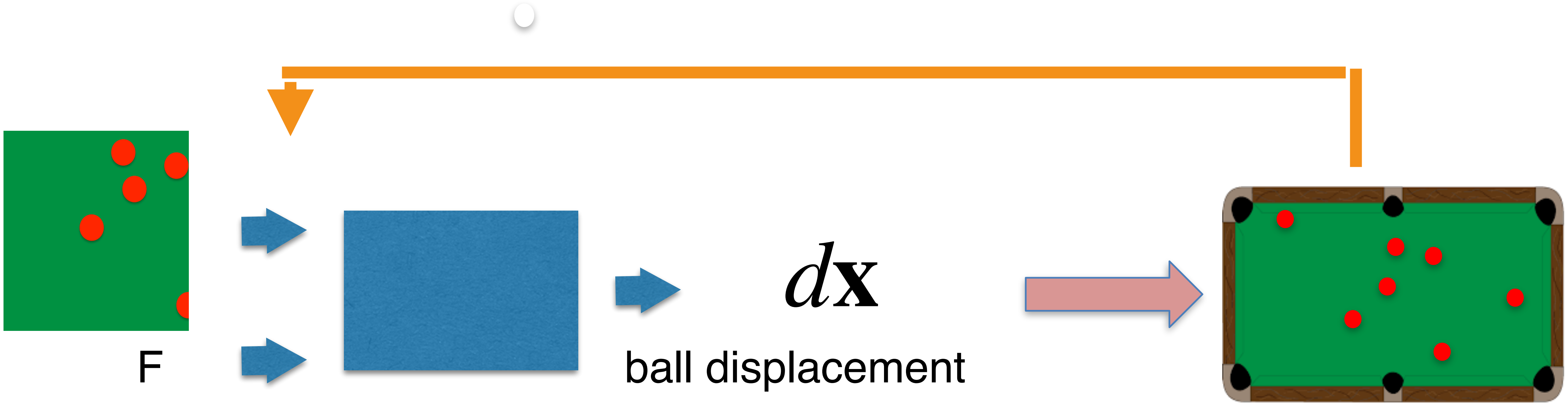
Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Context around the object is captured by using large windows around the object of interest



Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

Unrolling with entity-centric dynamics models



- The object-centric model is shared across all objects in the scene.
- We apply it one object at a time to predict the object's future displacement.
- We then copy paste the ball at the predicted location, and feed back as input.

Visual Predictive Models of Intuitive Physics for Playing Billiards, ICLR 2016

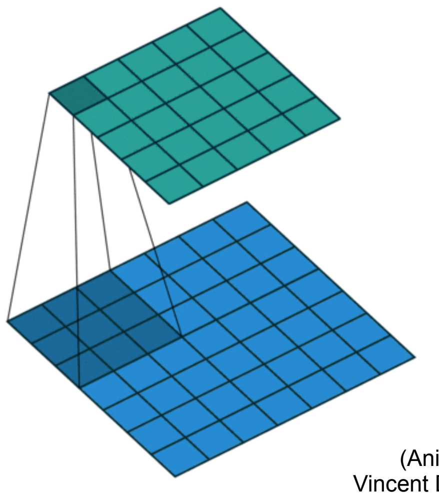
Cross-object interactions

How can we encode cross-object relations?

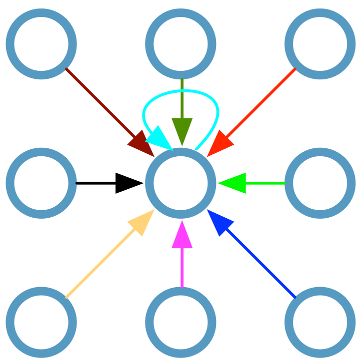
1. using large context windows around each object (this is what we just used)
2. using graph neural networks!

From CNNs to GNNs

Single CNN layer with 3x3 filter:

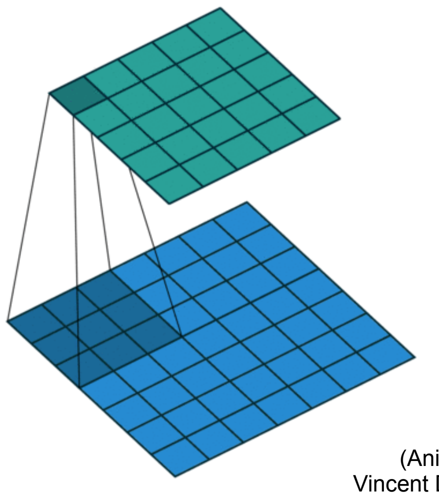


(Animation by Vincent Dumoulin)

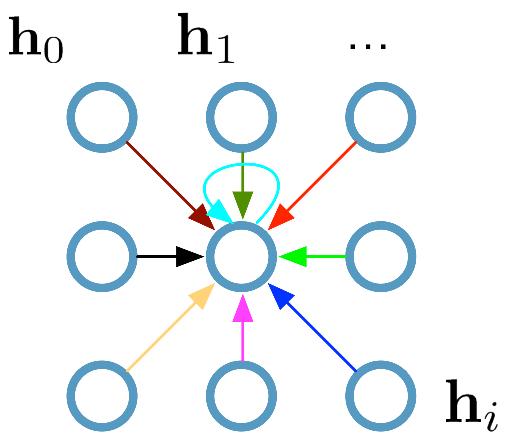


From CNNs to GNNs

Single CNN layer with 3x3 filter:

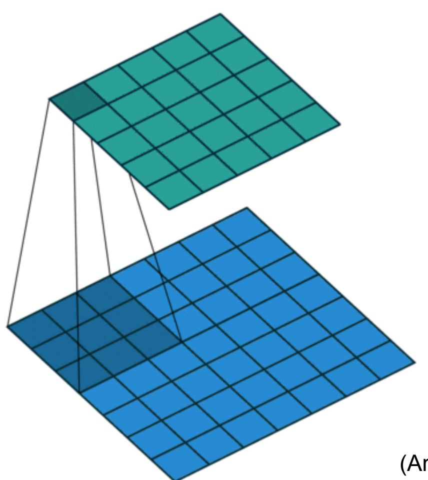


(Animation by Vincent Dumoulin)

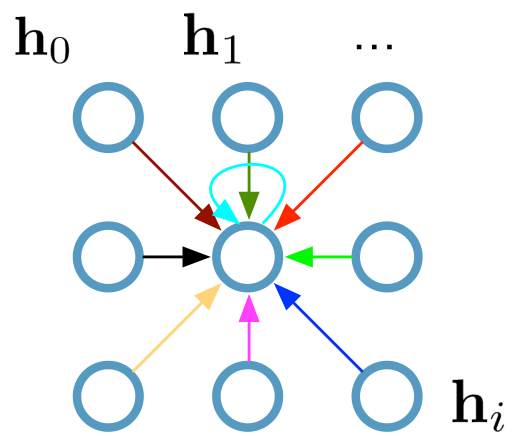


From CNNs to GNNs

Single CNN layer with 3x3 filter:



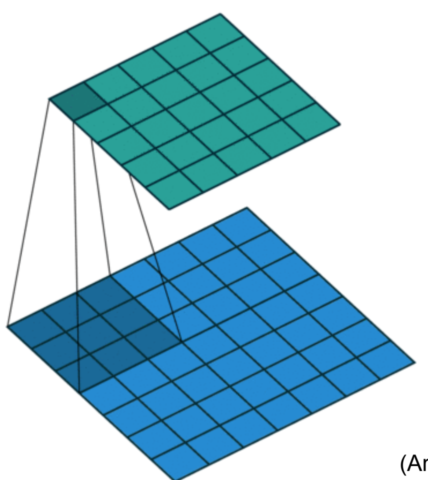
(Animation by Vincent Dumoulin)



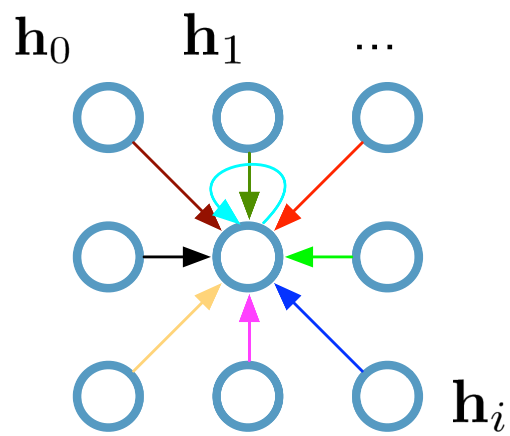
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

From CNNs to GNNs

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



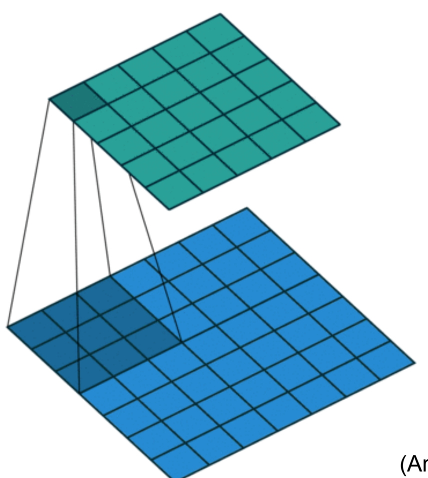
Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

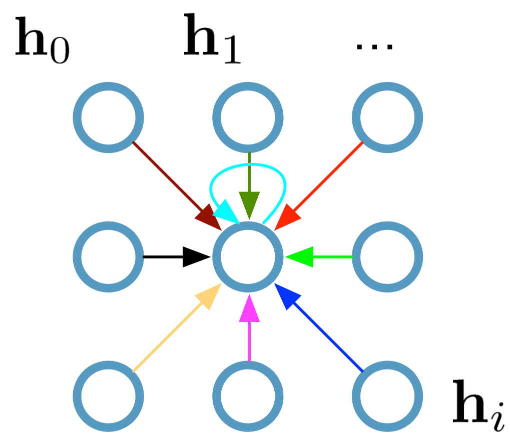
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

From CNNs to GNNs

Single CNN layer with 3x3 filter:



(Animation by Vincent Dumoulin)



Update for a single pixel:

- Transform messages individually $\mathbf{W}_i \mathbf{h}_i$
- Add everything up $\sum_i \mathbf{W}_i \mathbf{h}_i$

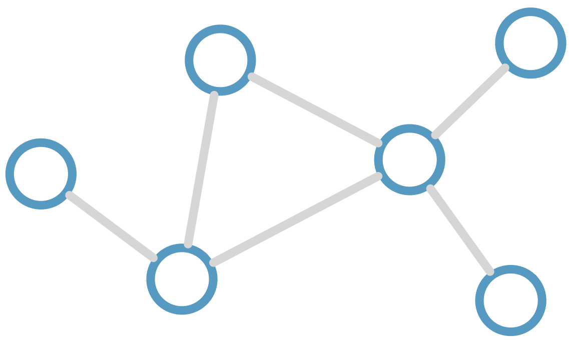
$\mathbf{h}_i \in \mathbb{R}^F$ are (hidden layer) activations of a pixel/node

Full update:

$$\mathbf{h}_4^{(l+1)} = \sigma\left(\mathbf{W}_0^{(l)}\mathbf{h}_0^{(l)} + \mathbf{W}_1^{(l)}\mathbf{h}_1^{(l)} + \cdots + \mathbf{W}_8^{(l)}\mathbf{h}_8^{(l)}\right)$$

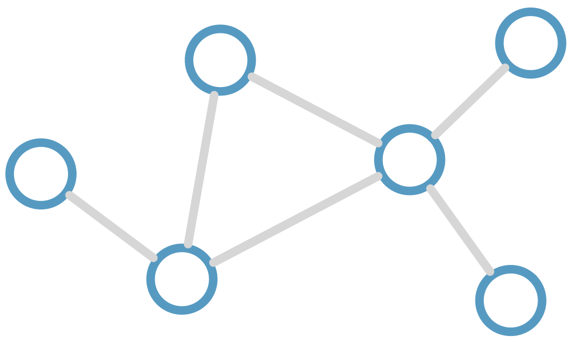
From CNNs to GNNs

Consider this
undirected graph:

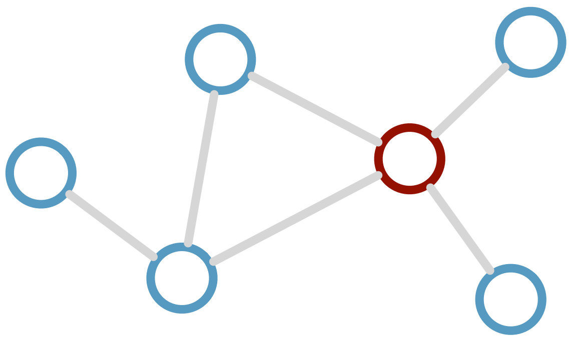


From CNNs to GNNs

Consider this undirected graph:

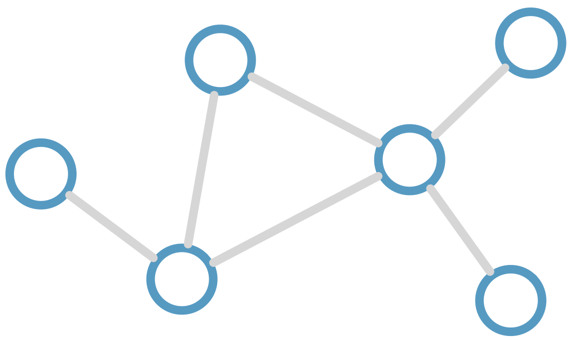


Calculate update for node in red:

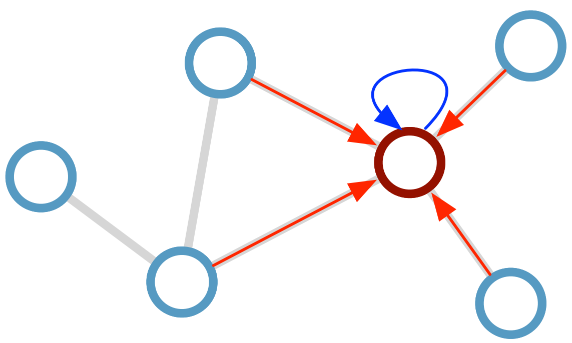


From CNNs to GNNs

Consider this undirected graph:

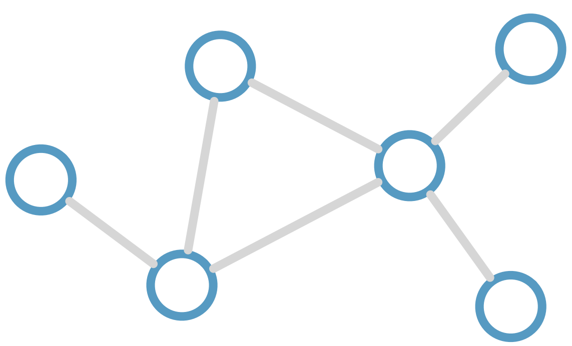


Calculate update for node in red:

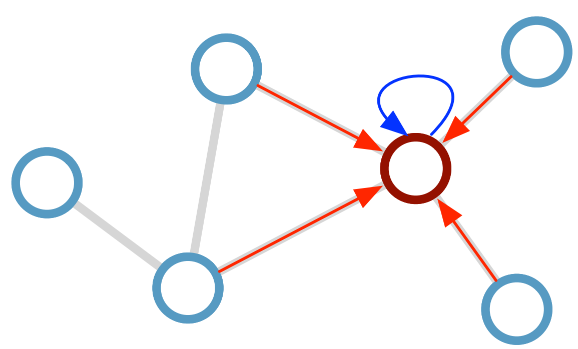


From CNNs to GNNs

Consider this undirected graph:



Calculate update for node in red:



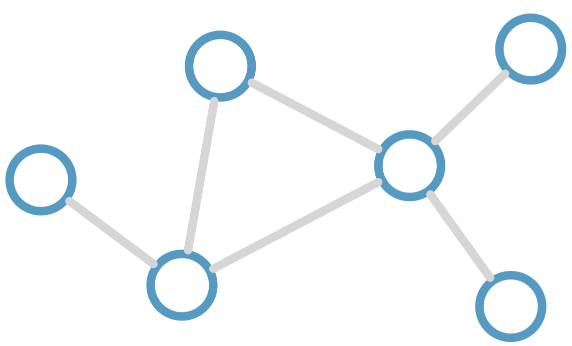
Update rule: $$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

$\mathcal{N}_i$: neighbor indices

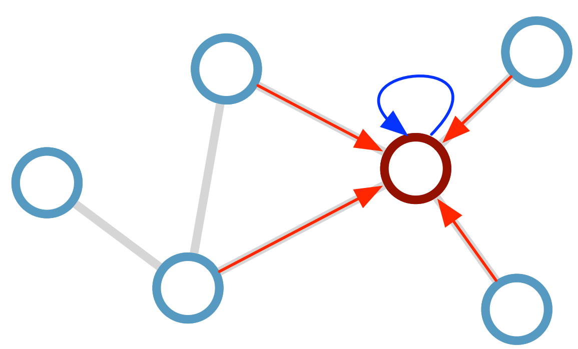
c_{ij} : norm. constant (fixed/trainable)

From CNNs to GNNs

Consider this undirected graph:



Calculate update for node in red:



Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j\in\mathcal{N}_i}\frac{1}{c_{ij}}\mathbf{h}_j^{(l)}\mathbf{W}_1^{(l)}\right)$$

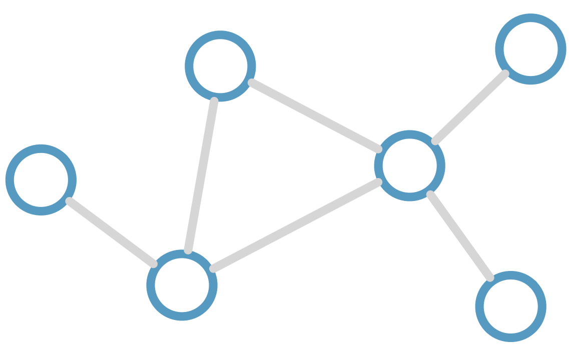
Scalability: subsample messages [Hamilton et al., NIPS 2017]

$\mathcal{N}_i$: neighbor indices

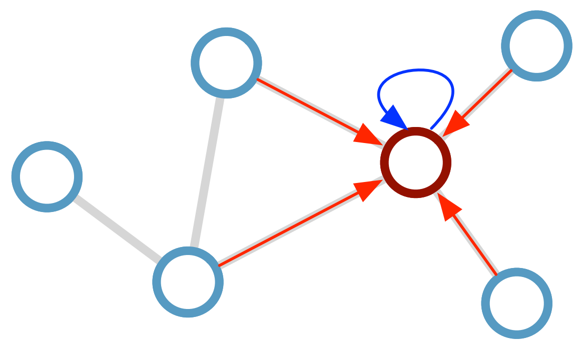
c_{ij} : norm. constant (fixed/trainable)

From CNNs to GNNs

Consider this undirected graph:



Calculate update for node in red:



Desirable properties:

- Weight sharing over all locations
- Invariance to permutations
- Linear complexity O(E)

Update rule:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{h}_i^{(l)}\mathbf{W}_0^{(l)} + \sum_{j \in \mathcal{N}_i} \frac{1}{c_{ij}} \mathbf{h}_j^{(l)} \mathbf{W}_1^{(l)}\right)$$

Scalability: subsample messages [Hamilton et al., NIPS 2017]

$\mathcal{N}_i$: neighbor indices

c_{ij} : norm. constant (fixed/trainable)

Interaction Networks for Learning about Objects, Relations and Physics

Peter W. Battaglia
Google DeepMind
London, UK N1C 4AG
peterbattaglia@google.com

Razvan Pascanu
Google DeepMind
London, UK N1C 4AG
razp@google.com

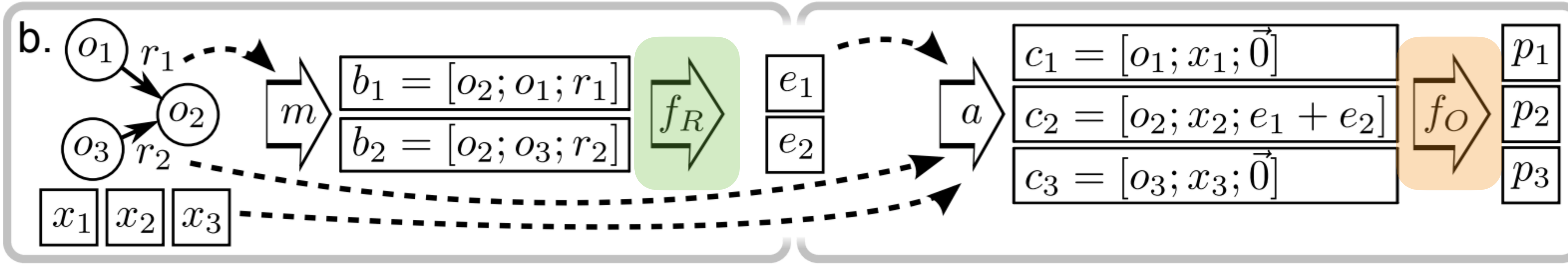
Matthew Lai
Google DeepMind
London, UK N1C 4AG
matthewlai@google.com

Danilo Rezende
Google DeepMind
London, UK N1C 4AG
danilor@google.com

Koray Kavukcuoglu
Google DeepMind
London, UK N1C 4AG
korayk@google.com

Main idea: Given a set of objects or object parts, use graph neural networks to predict their future velocities, given their physical properties and current positions and velocities

Interaction Networks for Learning about Objects, Relations and Physics, Battaglia et al., 2016



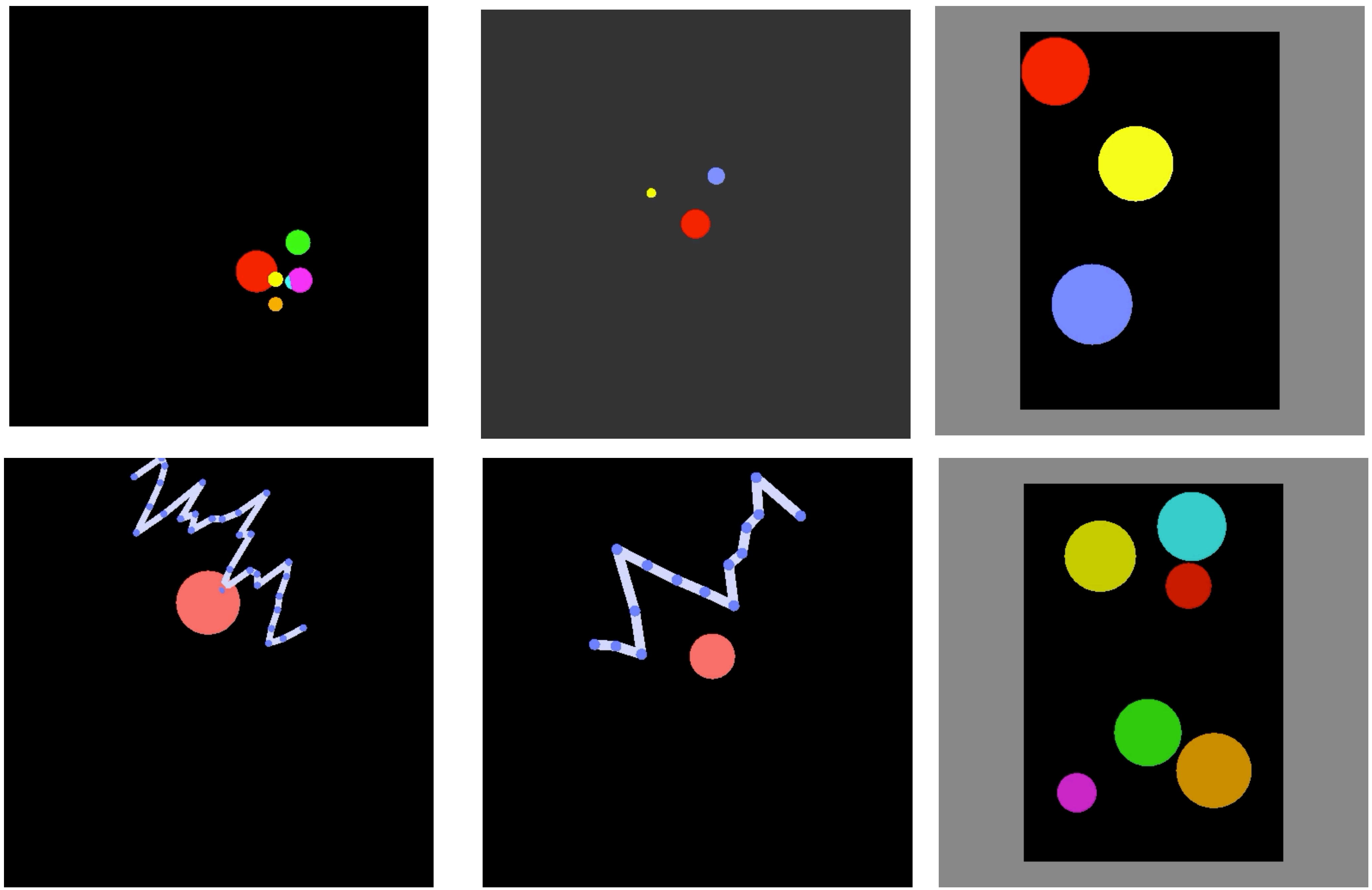
- Input:
  - Object state: dynamic (position/velocity), static(mass, size, shape)-> assumed given
  - Relation attributes: coefficient of restitution, spring constant
- Output: the velocities of the objects in the next time step.

- Relational (edge) neural net: takes two object states as input and relational attributes and predicts a feature vector
- Object (node) neural net: takes object states and summation of incoming edge messages and predicts future object velocity

- Can be used for unrolling by feeding the predictions back as input

Interaction Networks for Learning about Objects, Relations and Physics, Battaglia et al., 2016

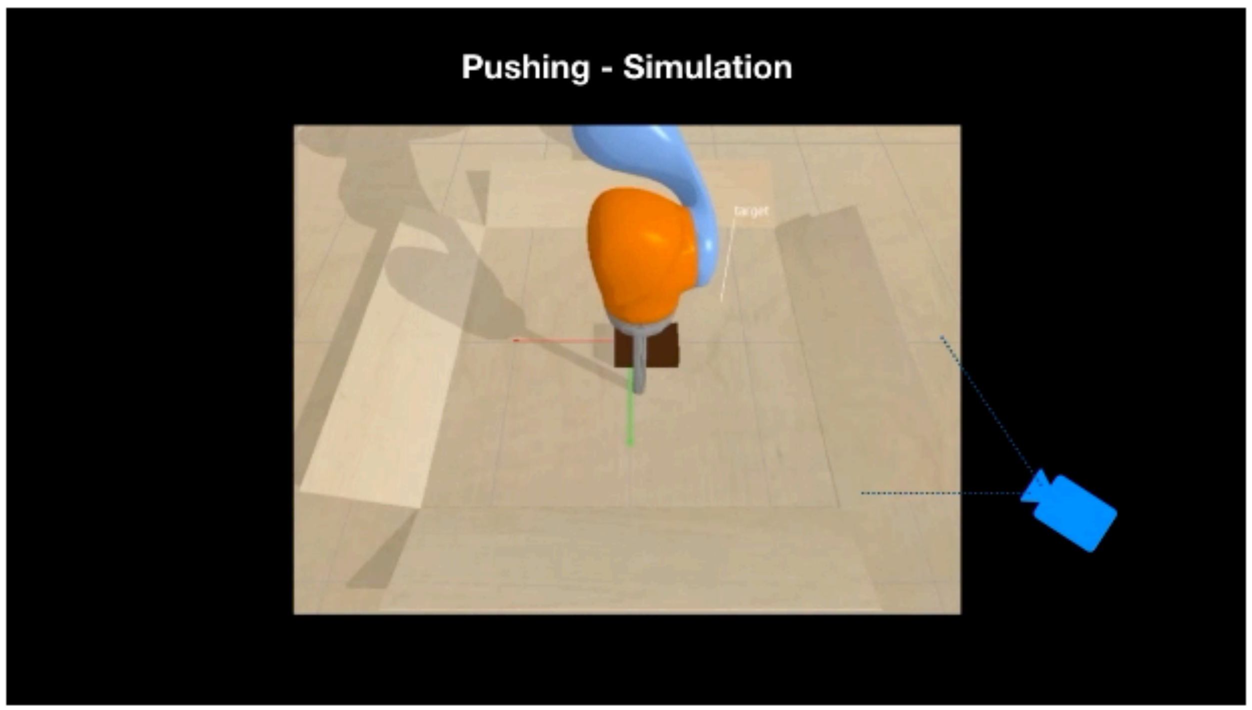
Unrolling results



Interaction Networks for Learning about Objects, Relations and Physics, Battaglia et al., 2016

Learning 3D object dynamics under

Manipulation under any viewpoint



Problems with 2D image centric representations

- No object permanence: objects disappear at occlusions
- Objects ``move” when the camera moves
- Objects change size when the camera zooms in/out

Camera motion is entangled with scene appearance in a 2D image.

SLAM

ORB-SLAM 2.0

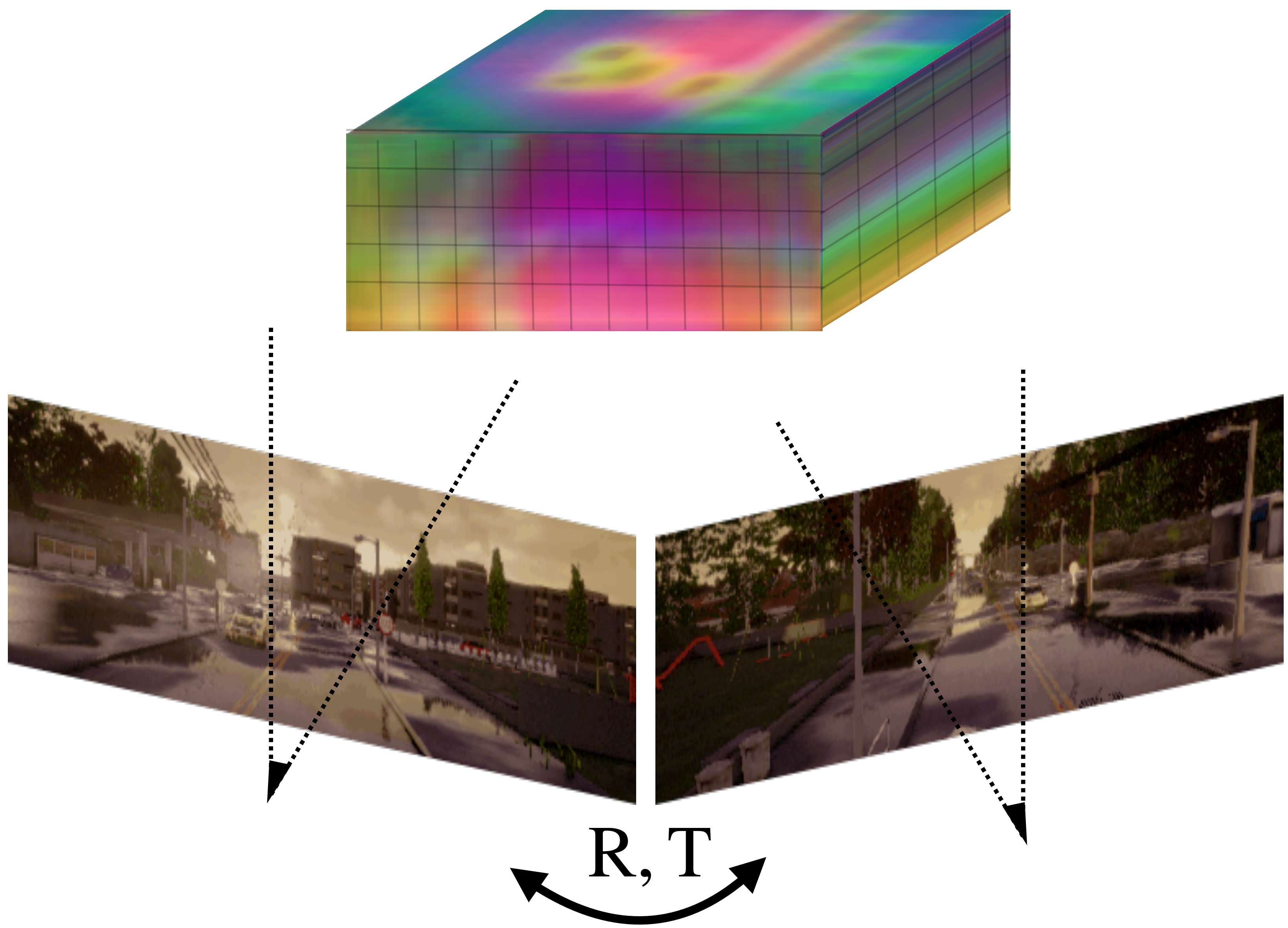
- SLAM disentangles a video into scene appearance (point cloud map) and camera motion
- Objects persist in the pointcloud map

but…

- SLAM cannot do amodal completion: it does not predict what the camera does not see.

- it may not optimize for the right end task (recognizing and acting in the world)

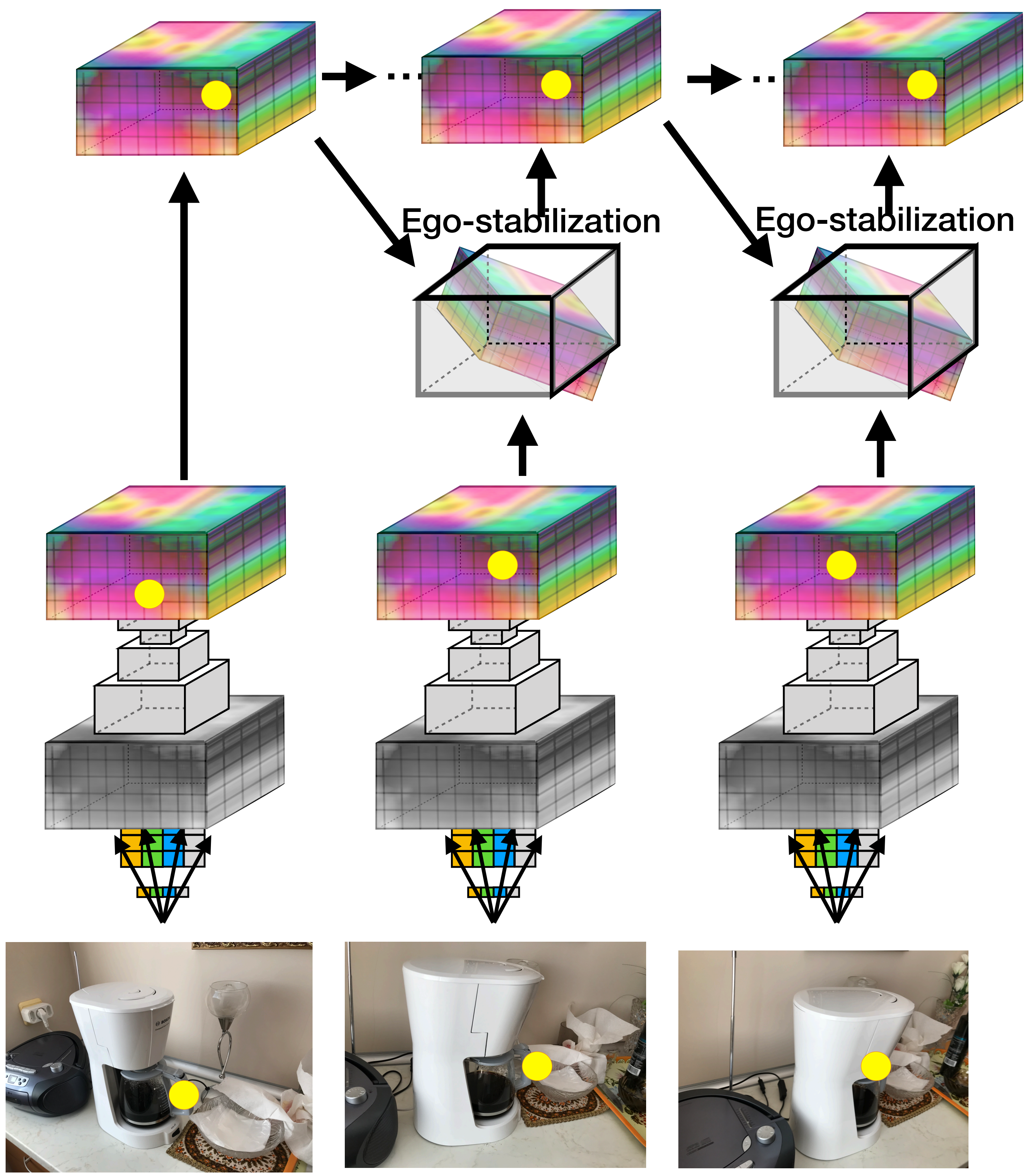
Geometry-Aware Recurrent Networks



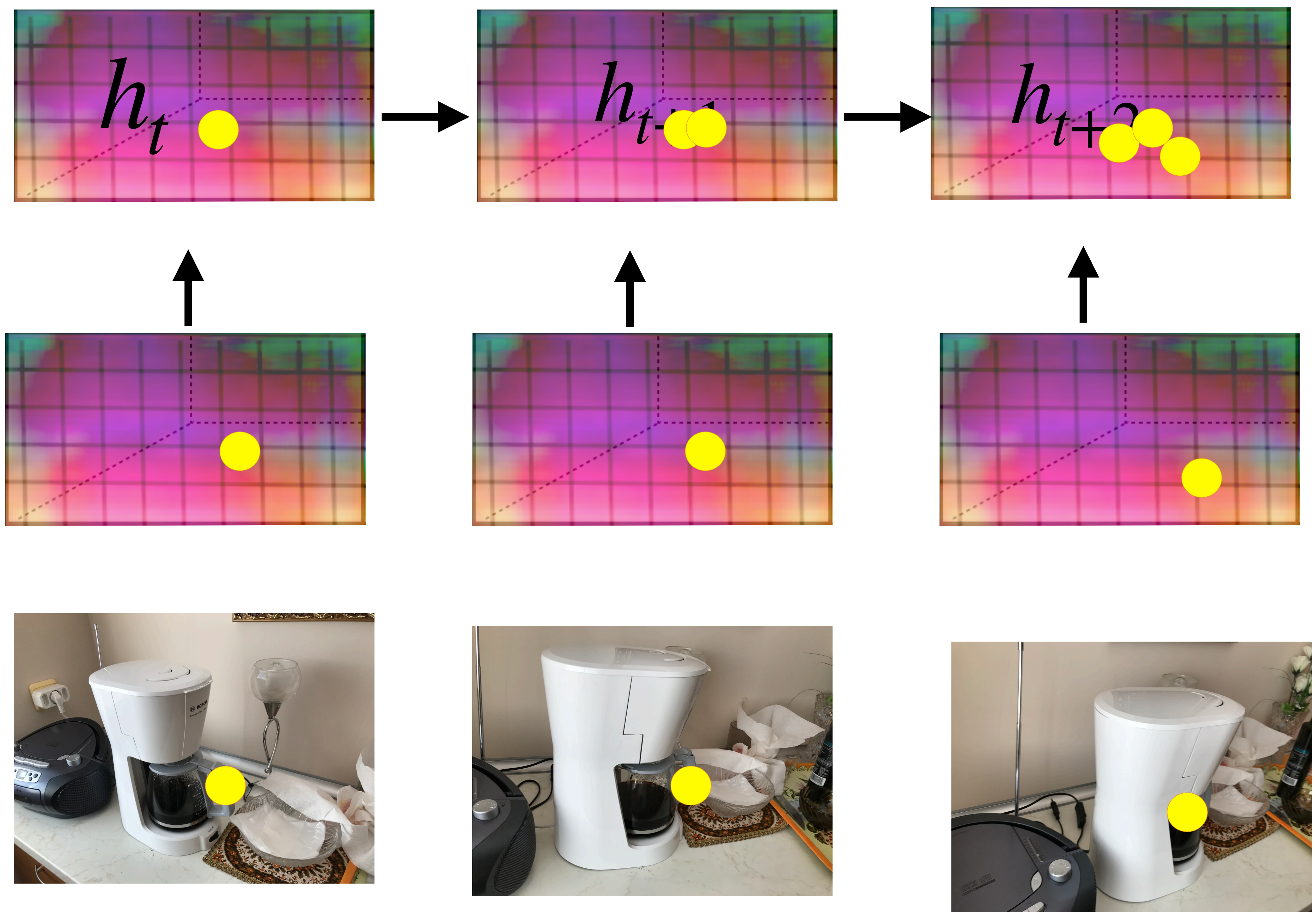
- 3-dimensional latent state
- Egomotion-stabilized latent state updates

Learning spatial common sense with geometry-aware recurrent networks, Tung et al. CVPR 2019

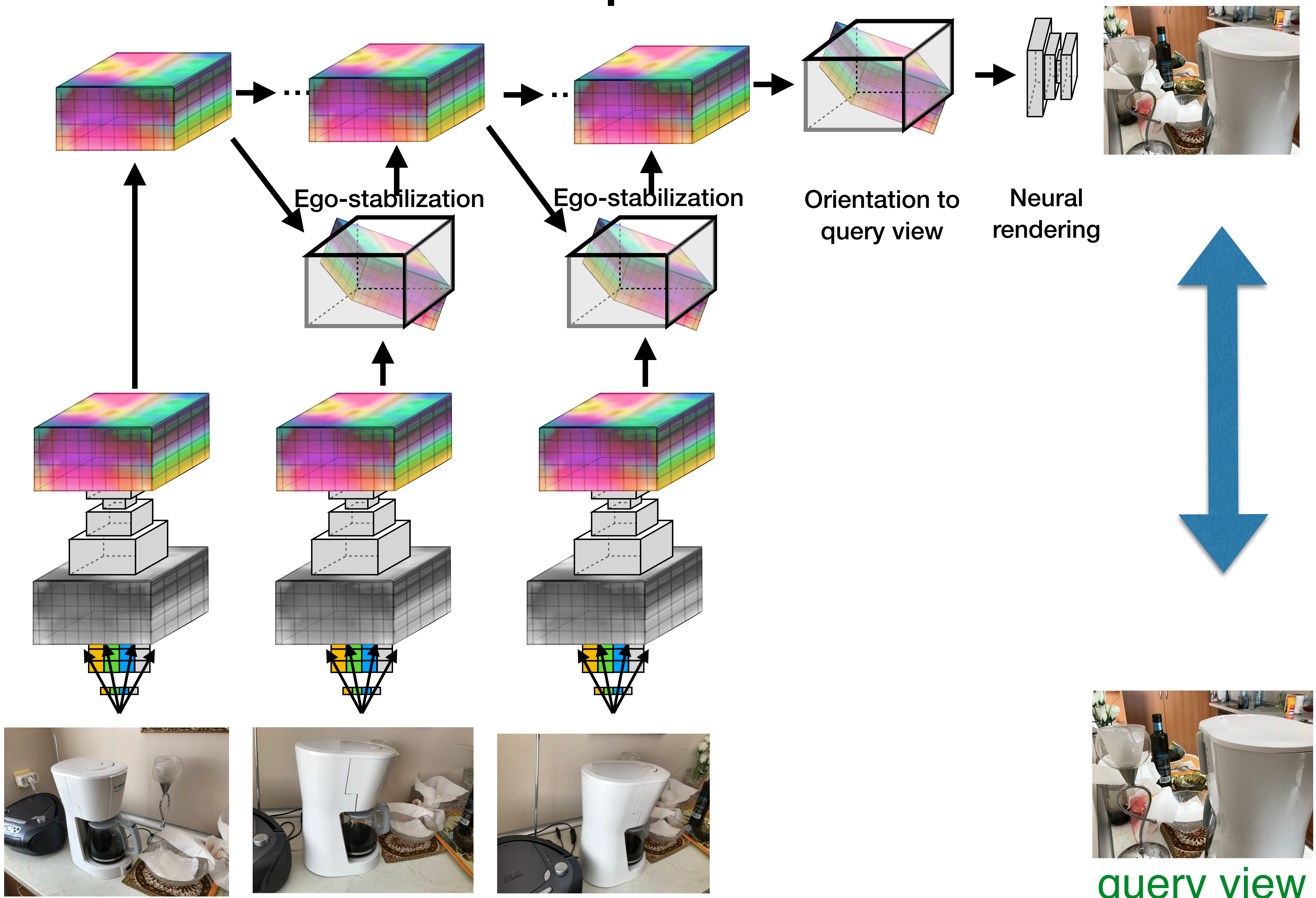
Geometry-Aware Recurrent Networks

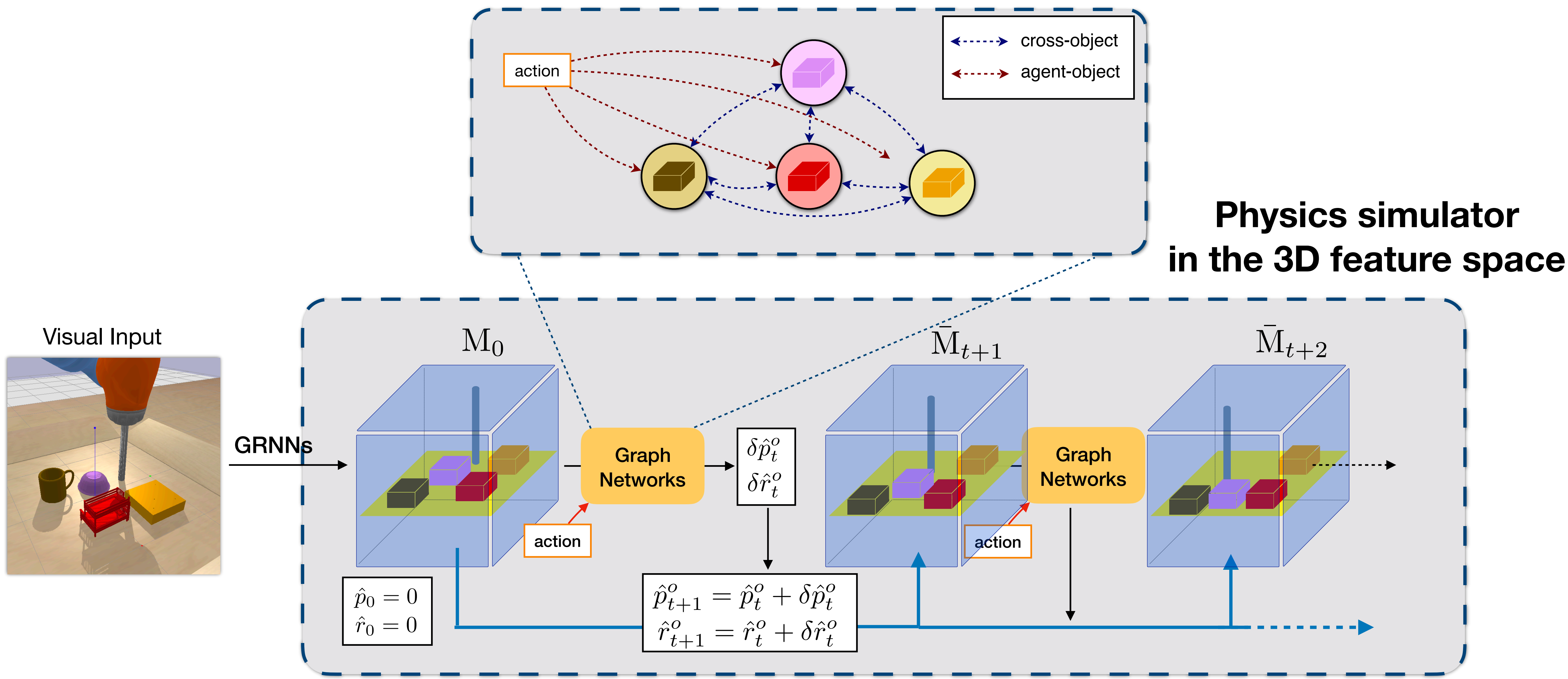


2D RNNs (conv-LSTMs/GRUs)



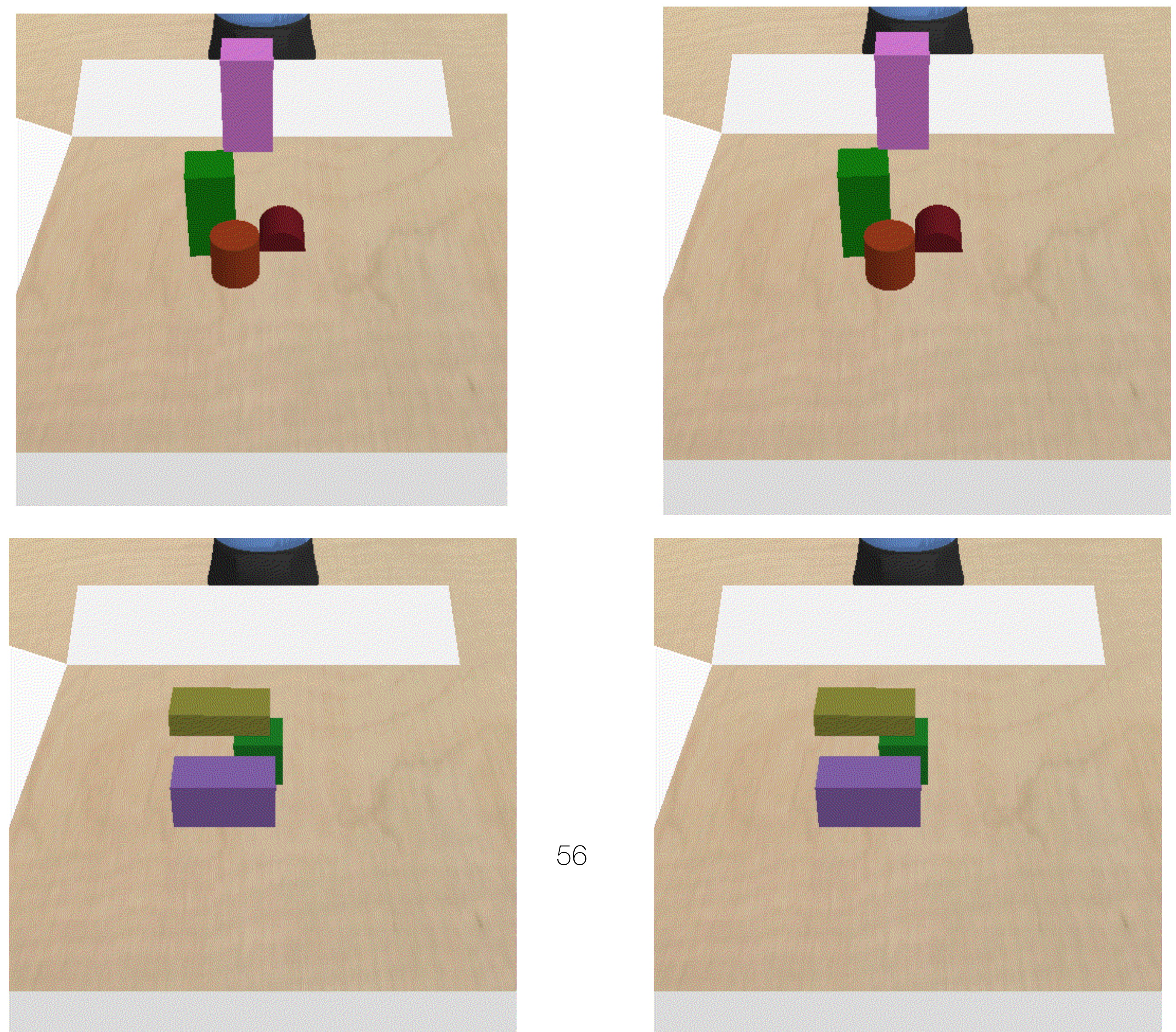
View prediction




Physics simulator
in the 3D feature space
action
cross-object
agent-object
Visual Input
GRNNs
M_0
$\bar{M}_{t+1}$
$\bar{M}_{t+2}$
Graph Networks
Graph Networks
action
action
$\delta\hat{p}_t^o$
$\delta\hat{r}_t^o$
$\hat{p}_0 = 0$
$\hat{r}_0 = 0$
$\hat{p}_{t+1}^o = \hat{p}_t^o + \delta\hat{p}_t^o$
$\hat{r}_{t+1}^o = \hat{r}_t^o + \delta\hat{r}_t^o$

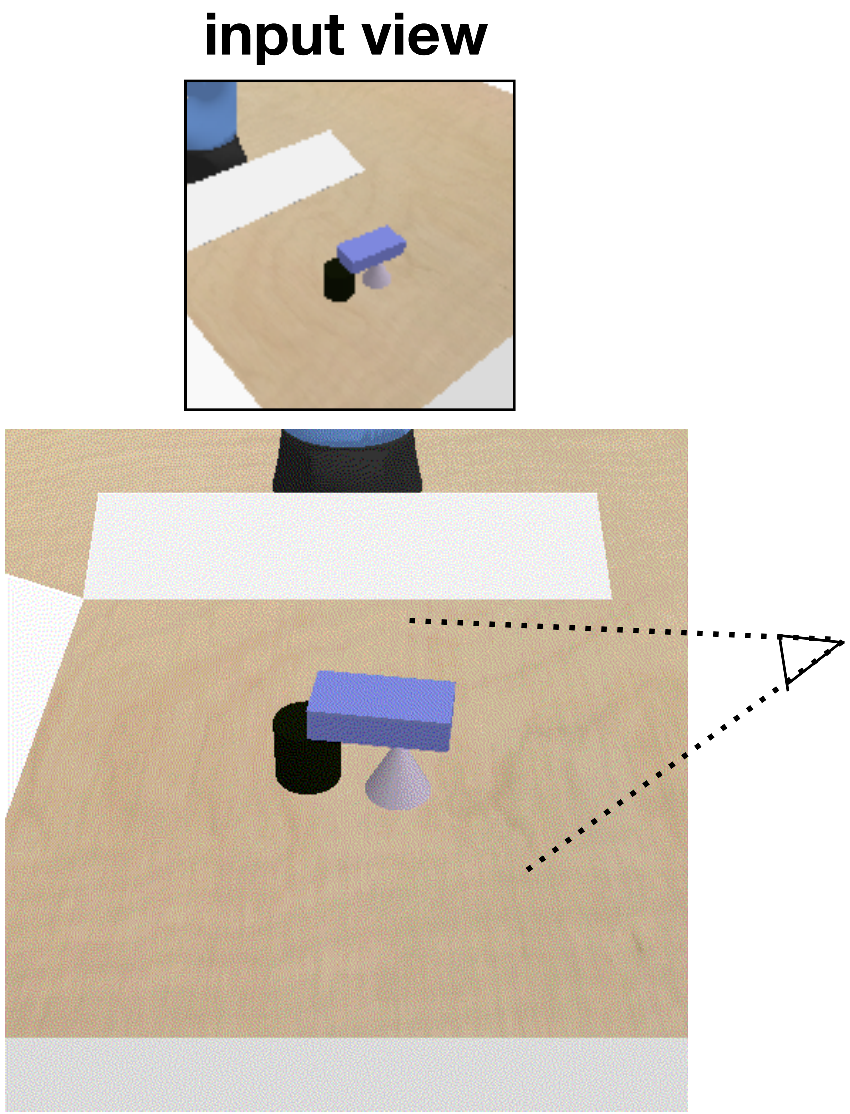
More diverse object dynamics

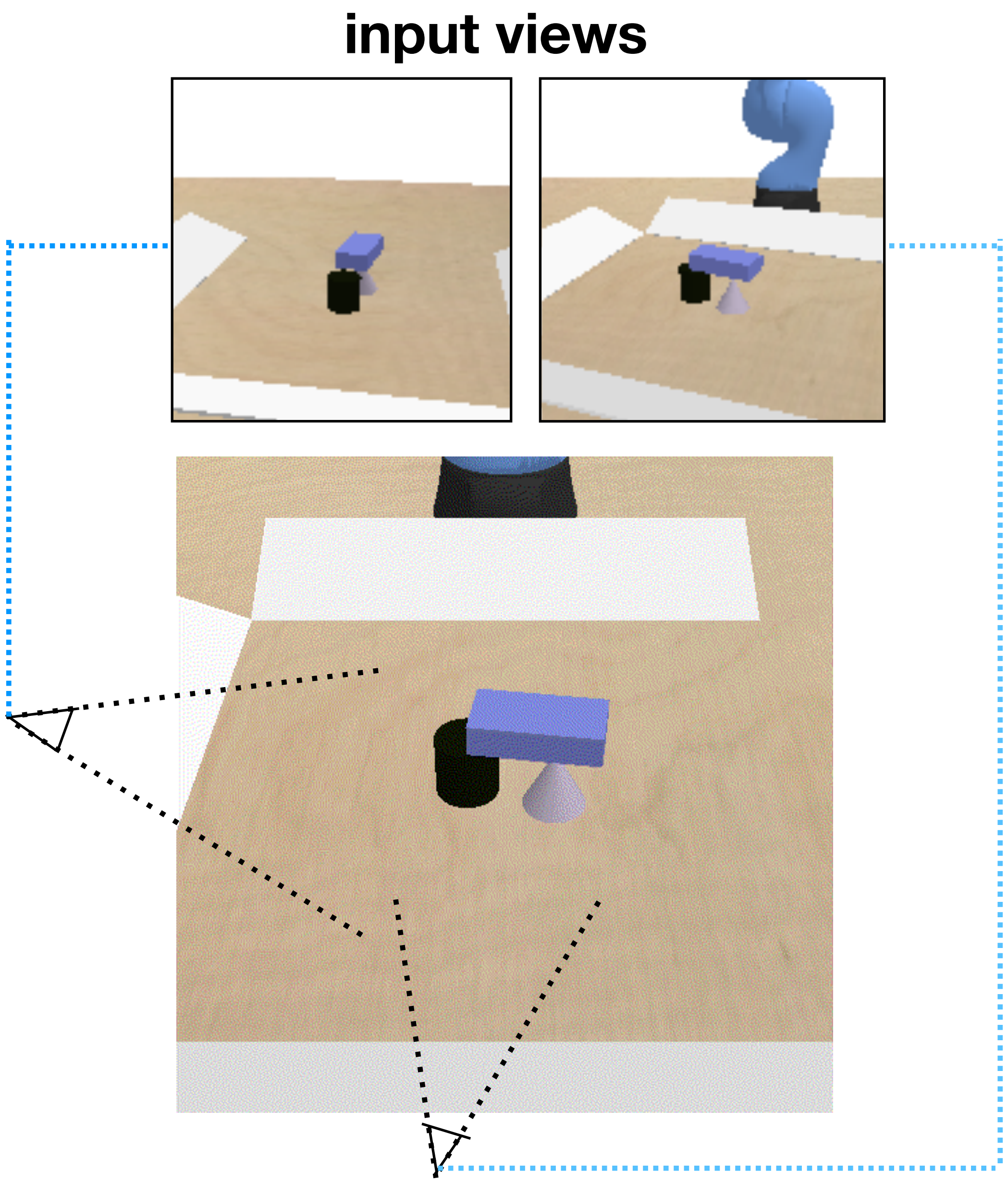
GT **Pred**



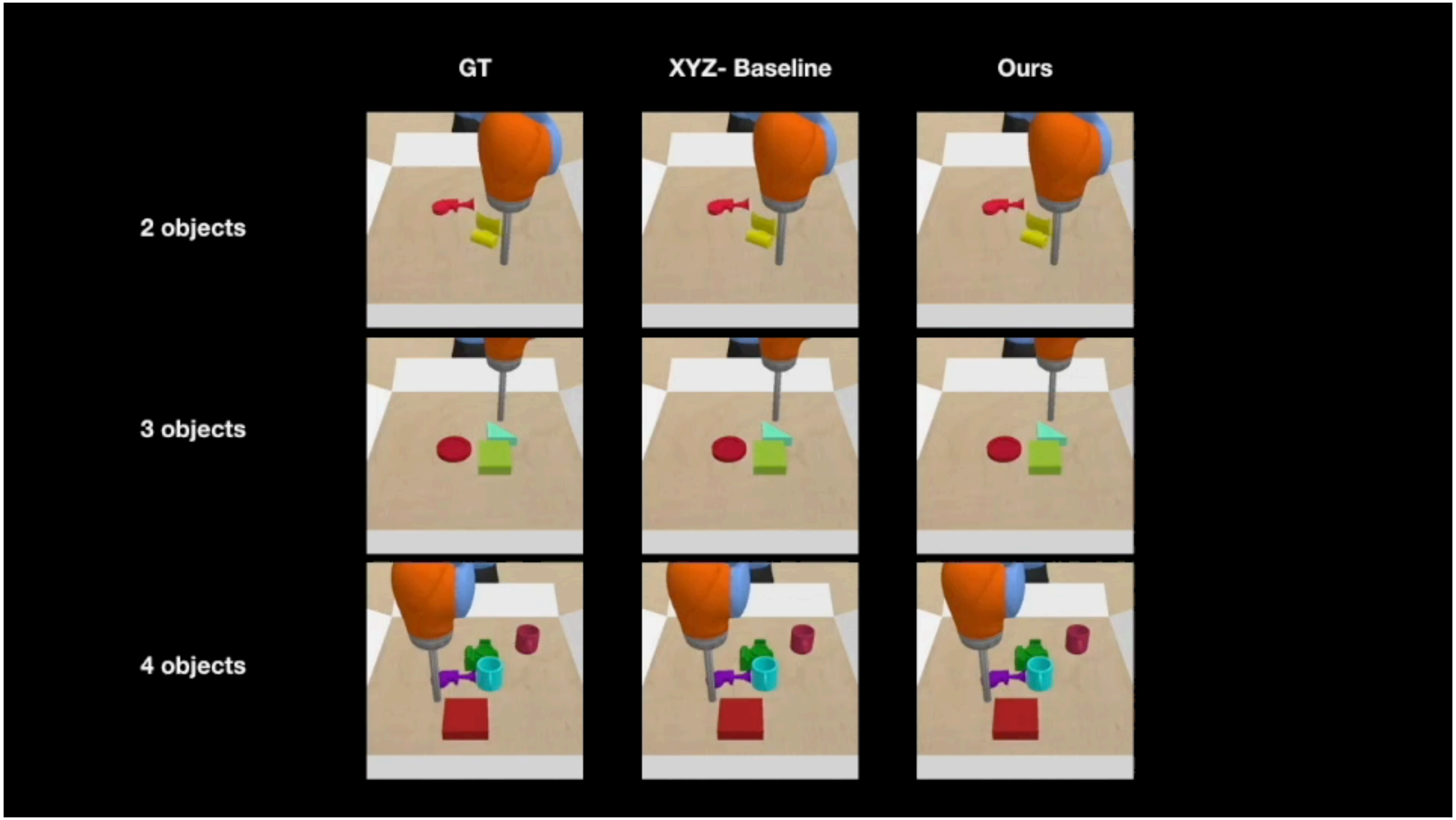
56

Intuitive physics under varying viewpoint



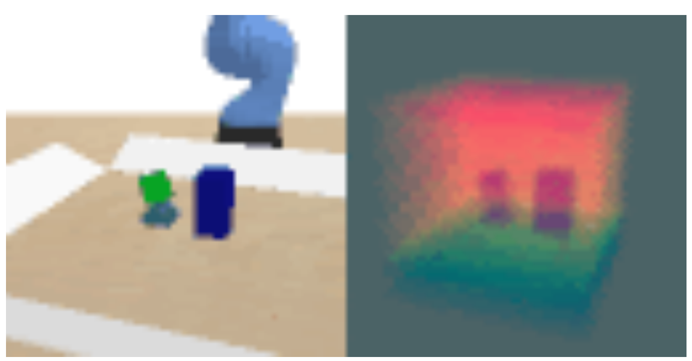
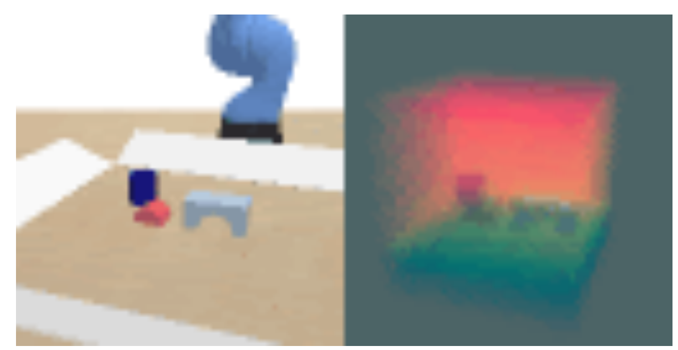


More diverse object dynamics

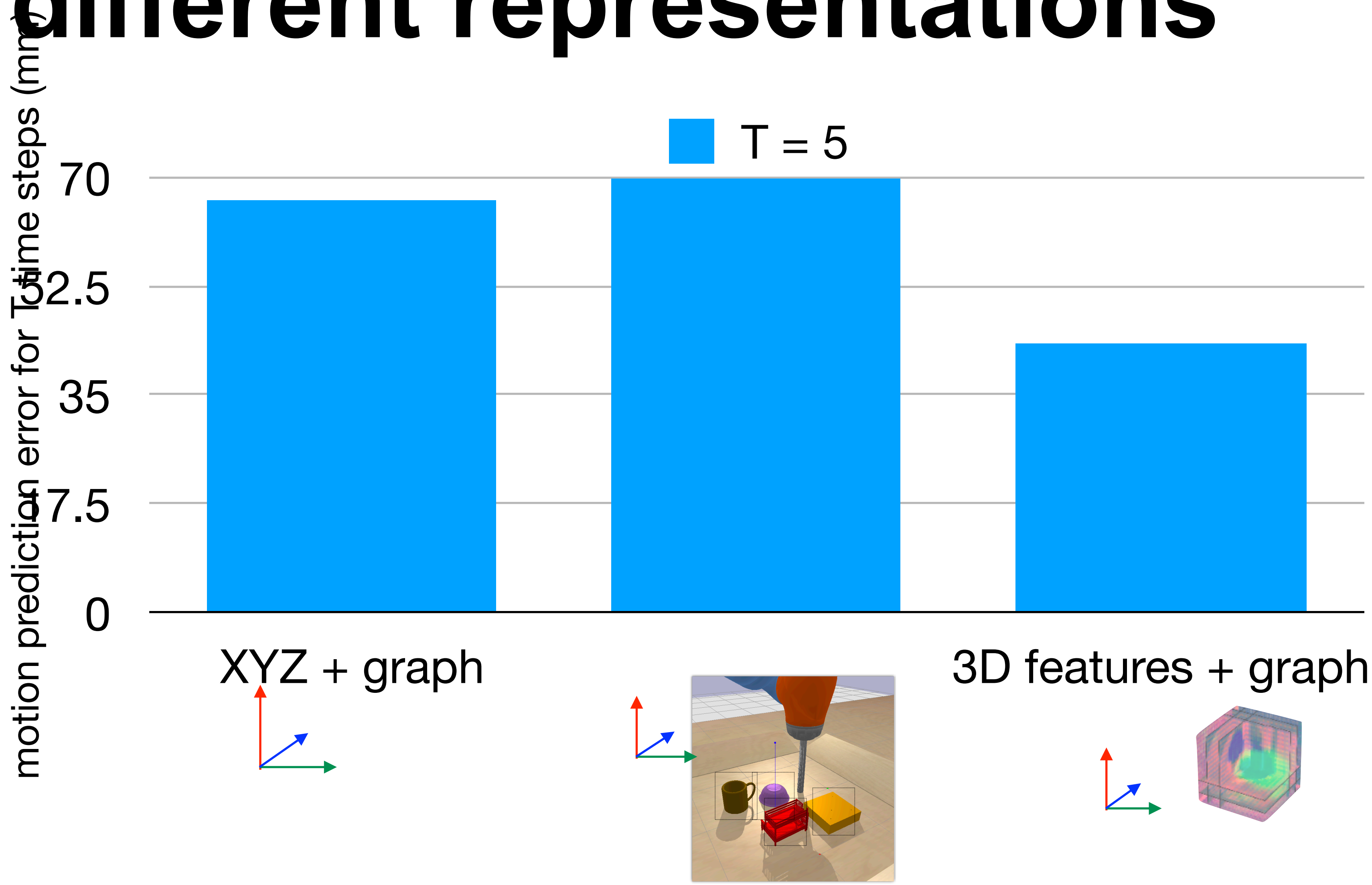


Learning object dynamics in a latent 3D feature space

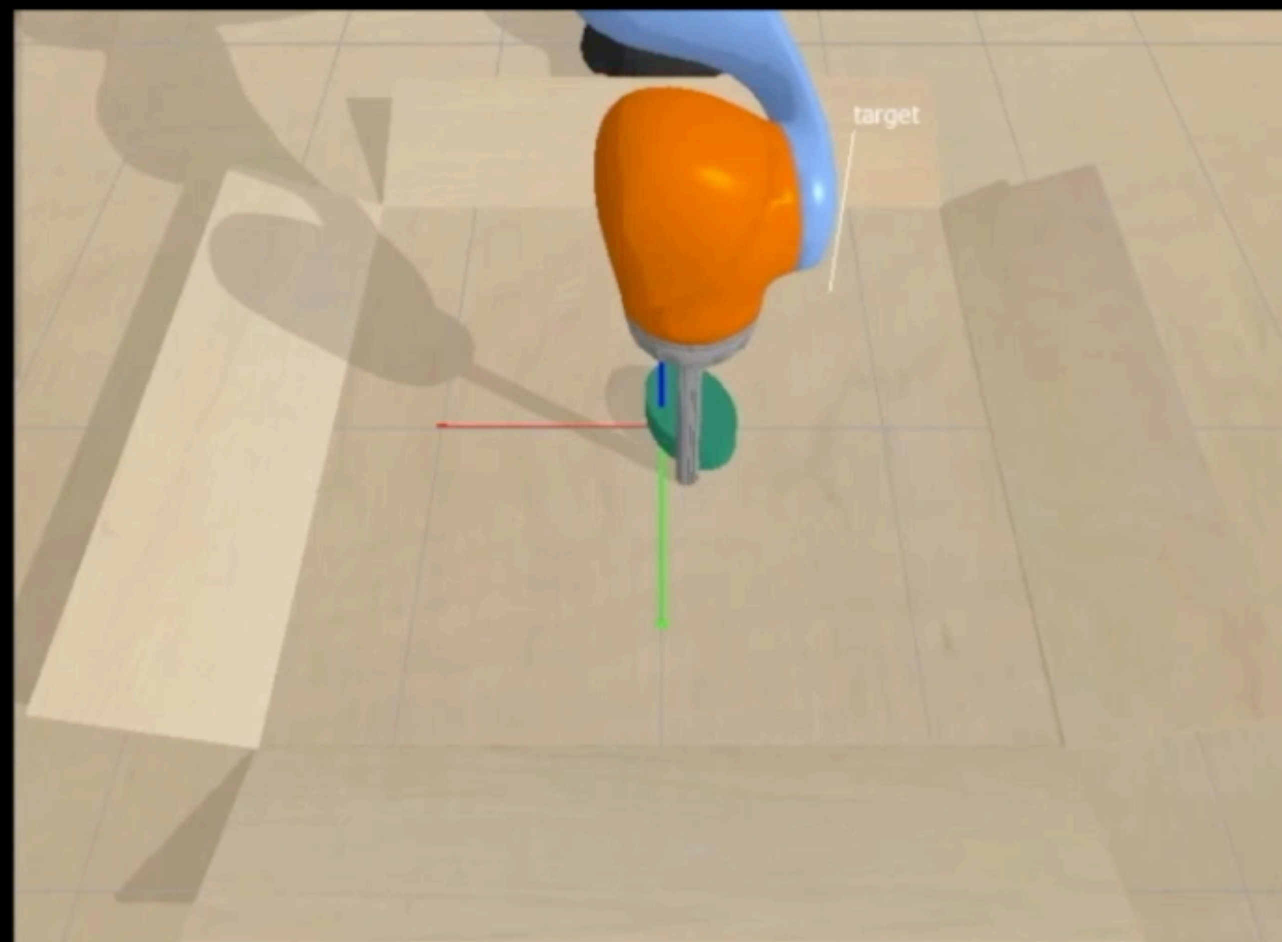


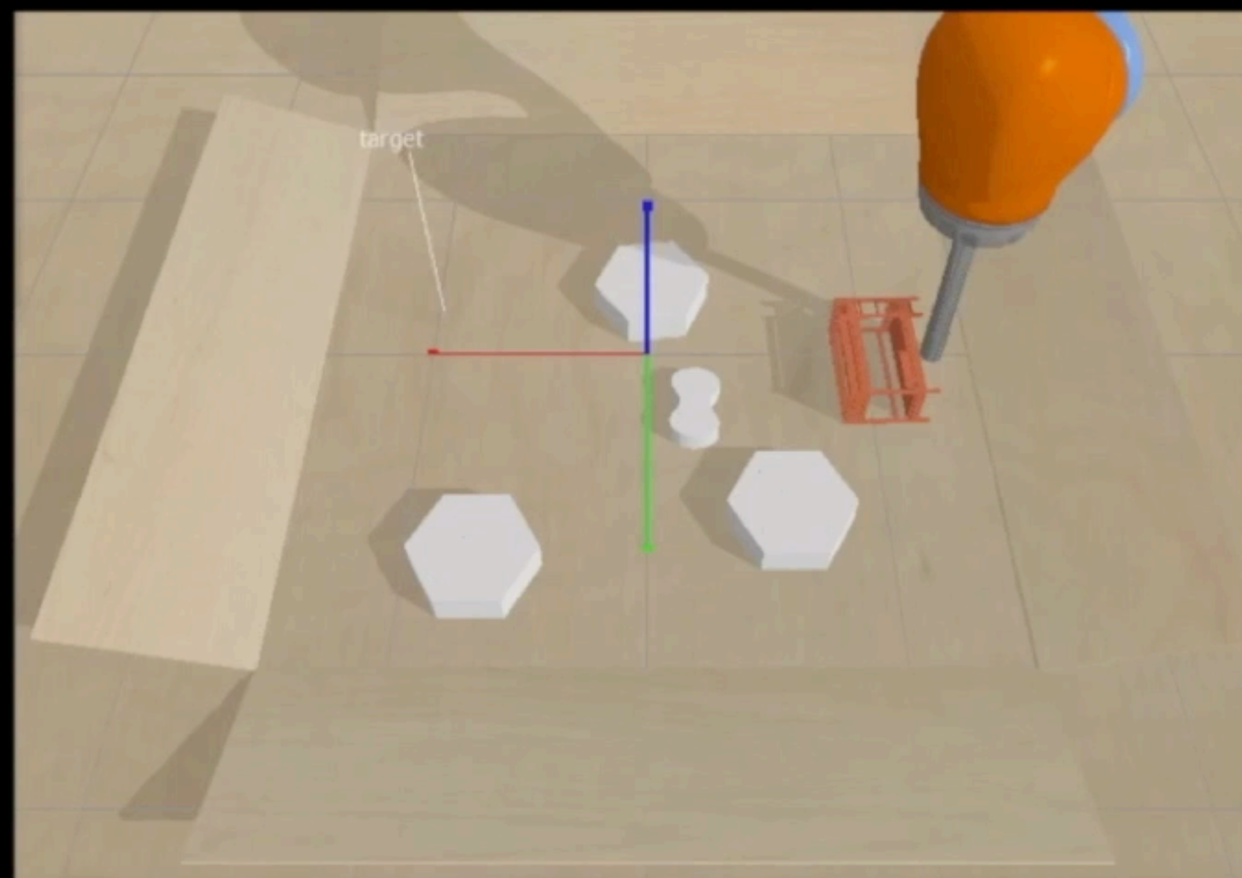
Comparison to models using different representations



Pushing - Simulation

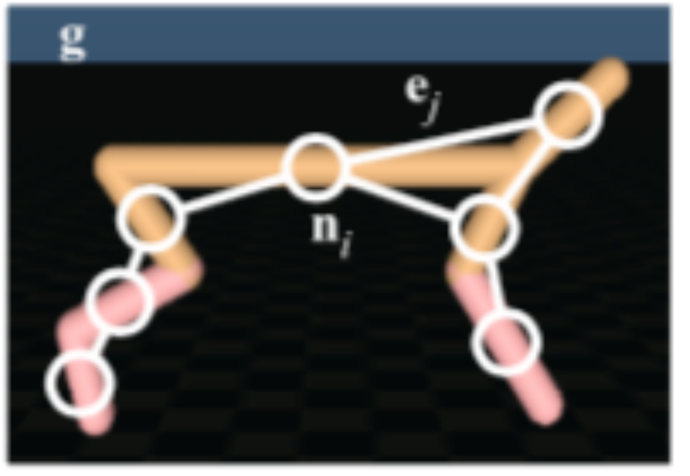


Obstacle Avoidance - Simulation Task 2



Robots as graphs

A physical system's bodies and joints can be represented by a graph's nodes and edges.



Node features

- Observable/dynamic: 3D position, 4D quaternion orientation, linear and angular velocities
- Unobservable/static: mass, inertia tensor
- Actions: forces applied on the joints

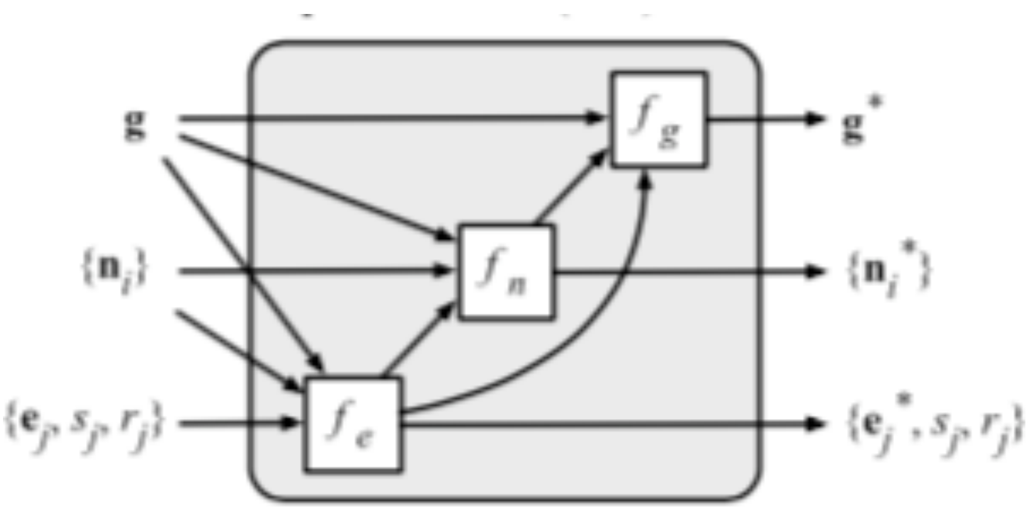
Predictions: I predict only the dynamic features, their temporal difference.
Train with regression.

Graph Networks as Learnable Physics Engines for Inference and Control, Gonzalez et al.

Robots as graphs

Node features

- Observable/dynamic: 3D position, 4D quaternion orientation, linear and angular velocities
- Unobservable/static: mass, inertia tensor
- Actions: forces applied on the joints
- No visual input here, much easier.



Algorithm 1 Graph network, GN

Input: Graph, $G = (\mathbf{g}, \{\mathbf{n}_i\}, \{\mathbf{e}_j, s_j, r_j\})$
for each edge $\{\mathbf{e}_j, s_j, r_j\}$ **do**
  Gather sender and receiver nodes $\mathbf{n}_{s_j}, \mathbf{n}_{r_j}$
  Compute output edges, $\mathbf{e}_j^* = f_e(\mathbf{g}, \mathbf{n}_{s_j}, \mathbf{n}_{r_j}, \mathbf{e}_j)$
end for
for each node $\{\mathbf{n}_i\}$ **do**
  Aggregate $\mathbf{e}_j^*$ per receiver, $\hat{\mathbf{e}}_i = \sum_{j/r_j=i} \mathbf{e}_j^*$
  Compute node-wise features, $\mathbf{n}_i^* = f_n(\mathbf{g}, \mathbf{n}_i, \hat{\mathbf{e}}_i)$
end for
Aggregate all edges and nodes $\hat{\mathbf{e}} = \sum_j \mathbf{e}_j^*$, $\hat{\mathbf{n}} = \sum_i \mathbf{n}_i^*$
Compute global features, $\mathbf{g}^* = f_g(\mathbf{g}, \hat{\mathbf{n}}, \hat{\mathbf{e}})$
Output: Graph, $G^* = (\mathbf{g}^*, \{\mathbf{n}_i^*\}, \{\mathbf{e}_j^*, s_j, r_j\})$

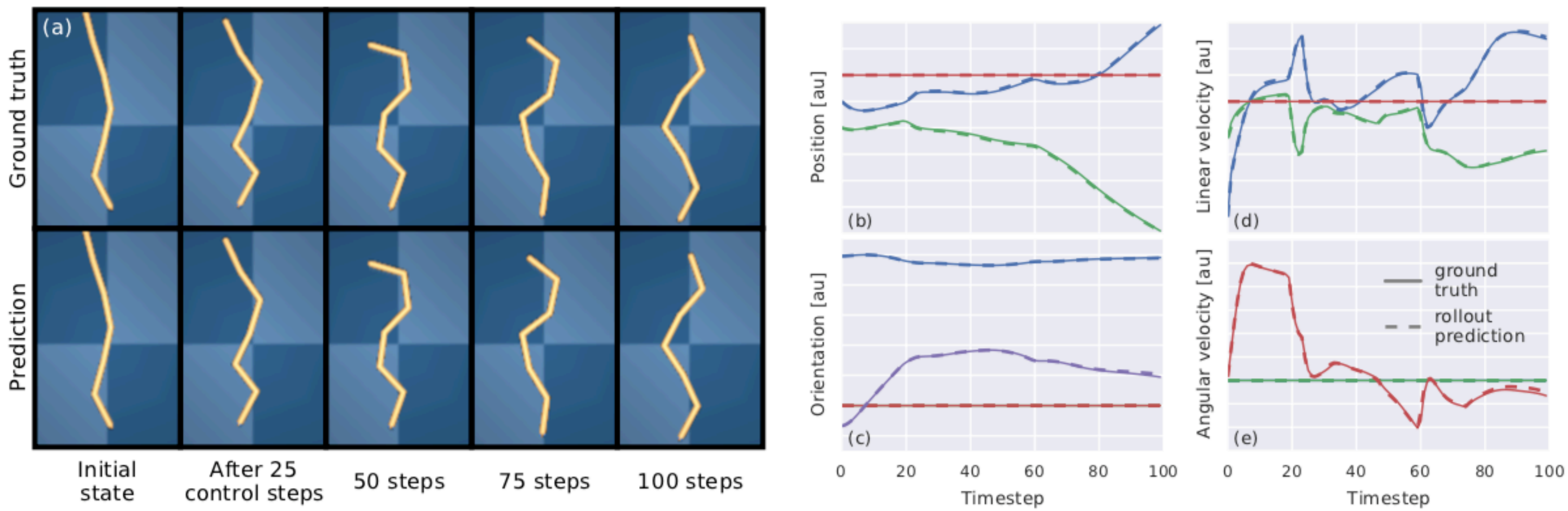
Predictions: I predict only the dynamic features, their temporal difference.
Train with regression.

Graph Networks as Learnable Physics Engines for Inference and Control, Gonzalez et al.

Robots as graphs

Node features

- Observable/dynamic: 3D position, 4D quaternion orientation, linear and angular velocities
- Unobservable/static: mass, inertia tensor
- Actions: forces applied on the joints

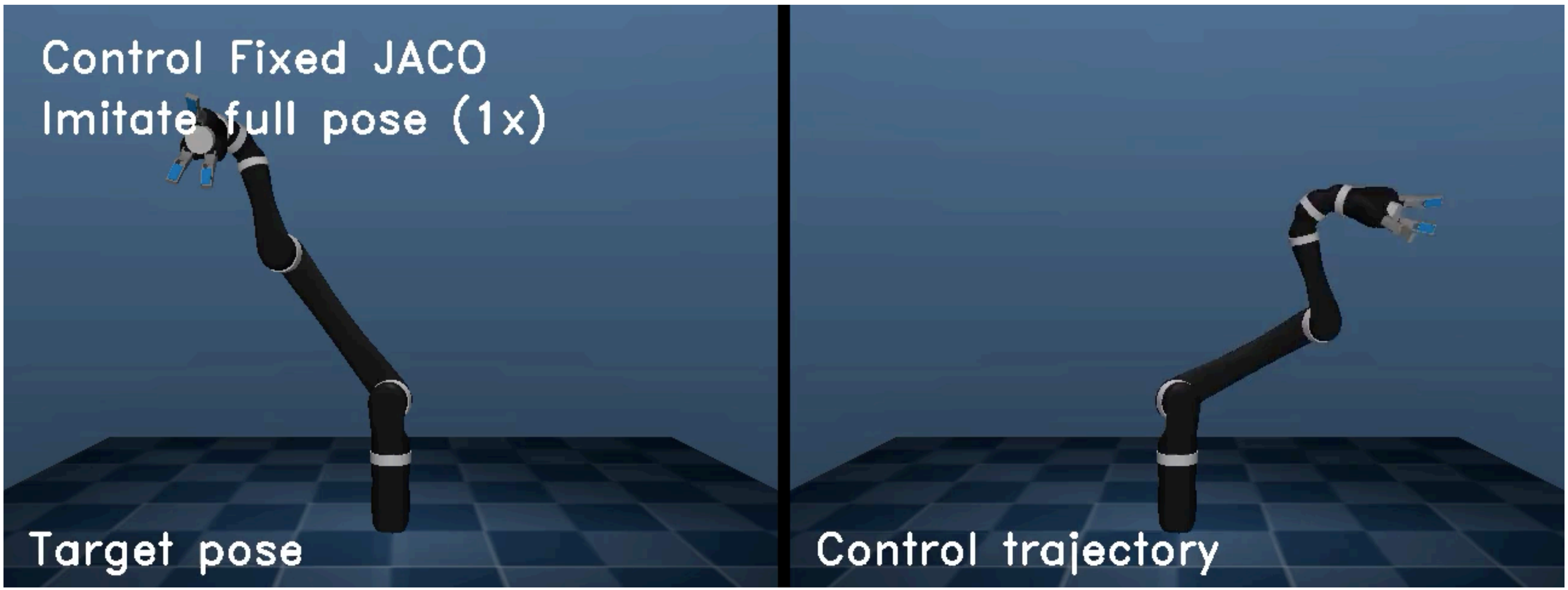


Predictions: I predict only the dynamic features, their temporal difference.

Graph Networks as Learnable Physics Engines for Inference and Control, Gonzalez et al.

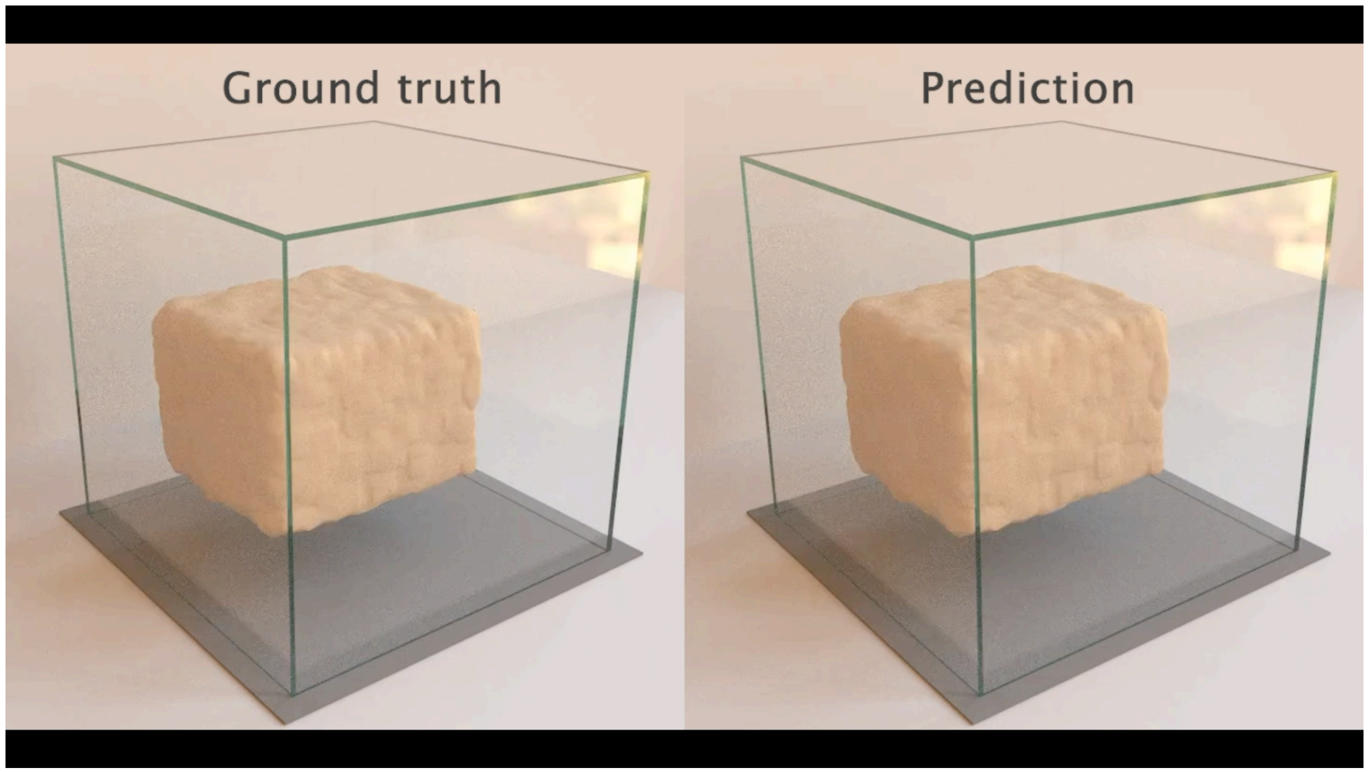
Model predictive control (MPC)

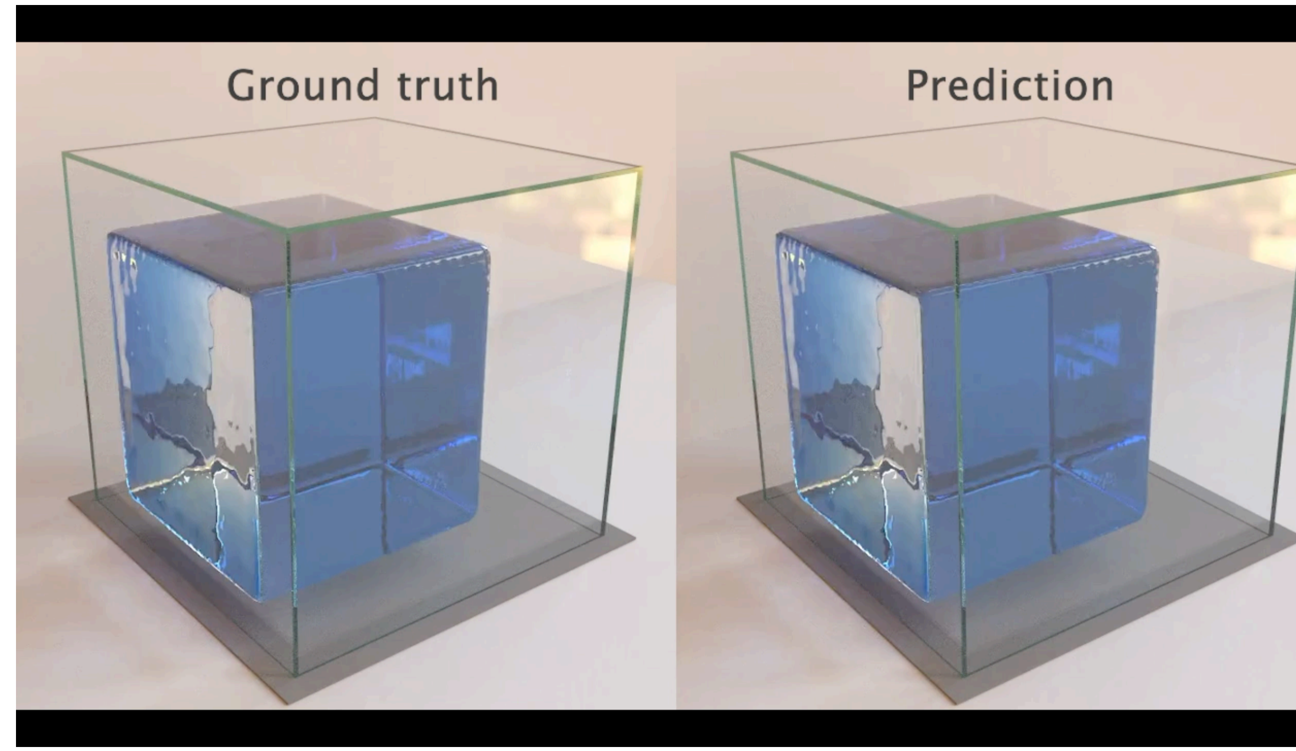
MPC to reach a target configuration

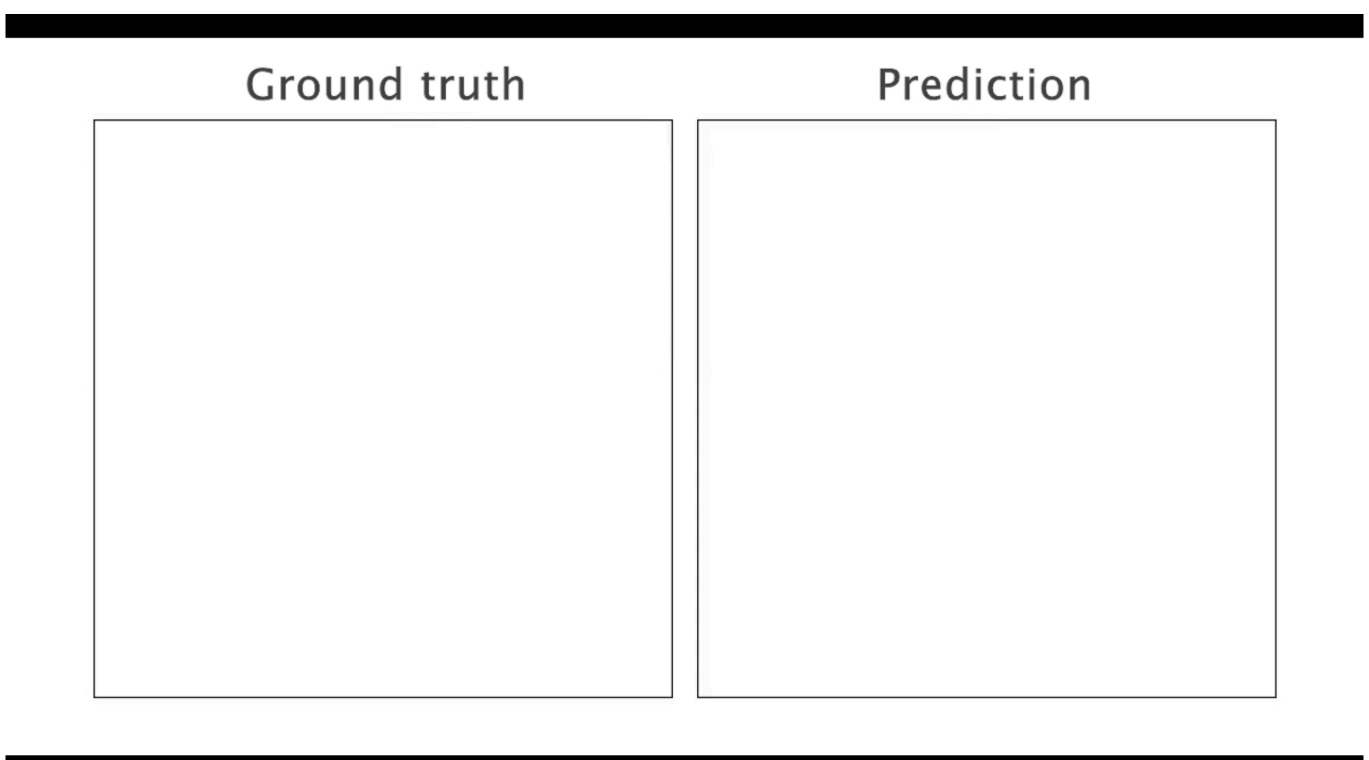


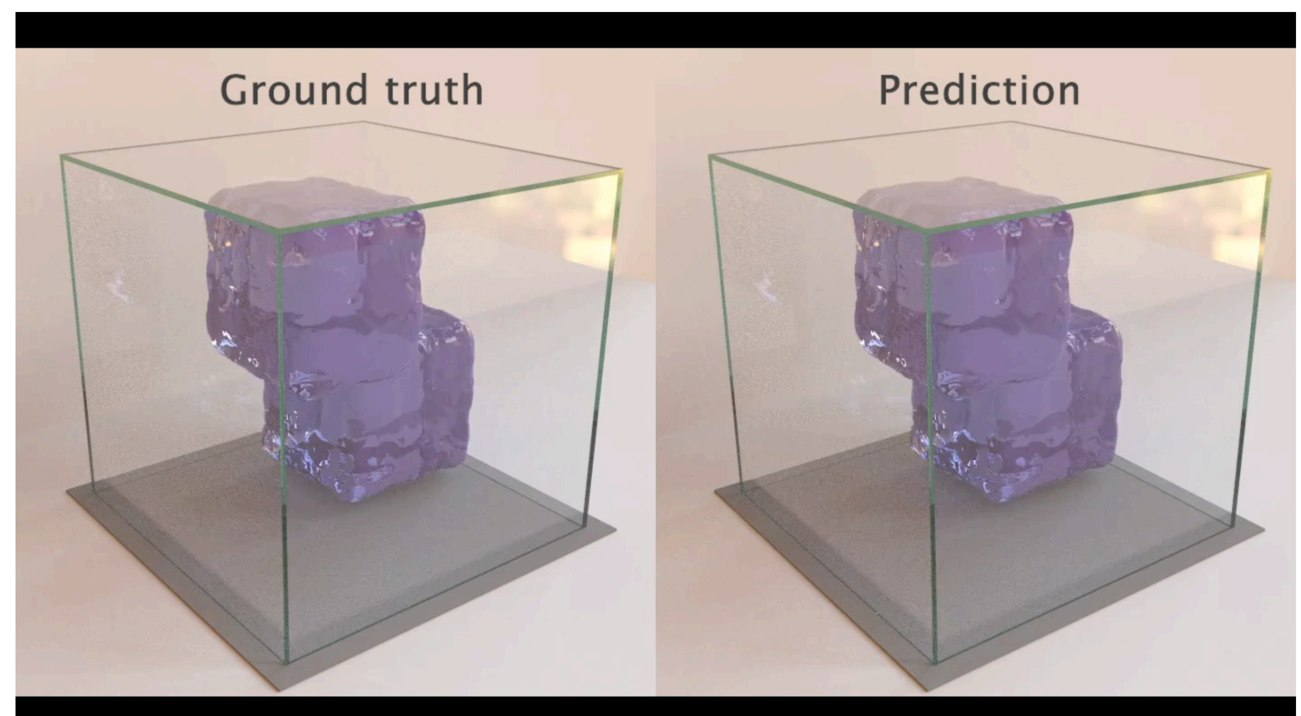
Graph Networks as Learnable Physics Engines for Inference and Control, Gonzalez et al.

GNNs over particles



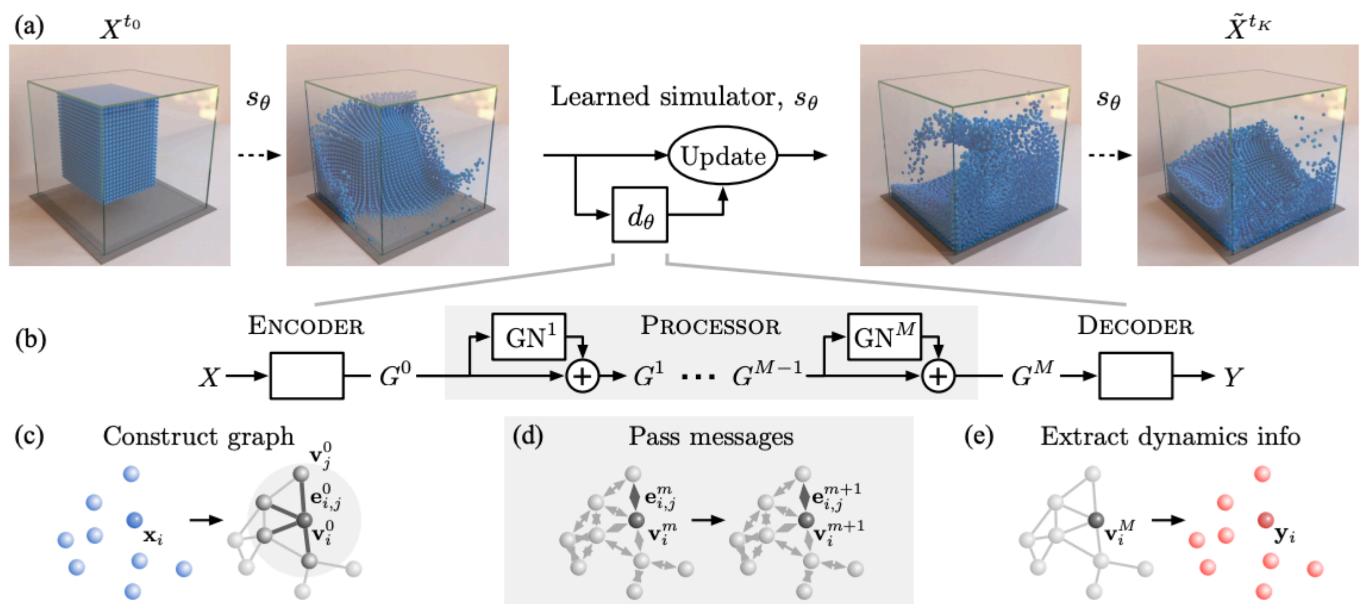






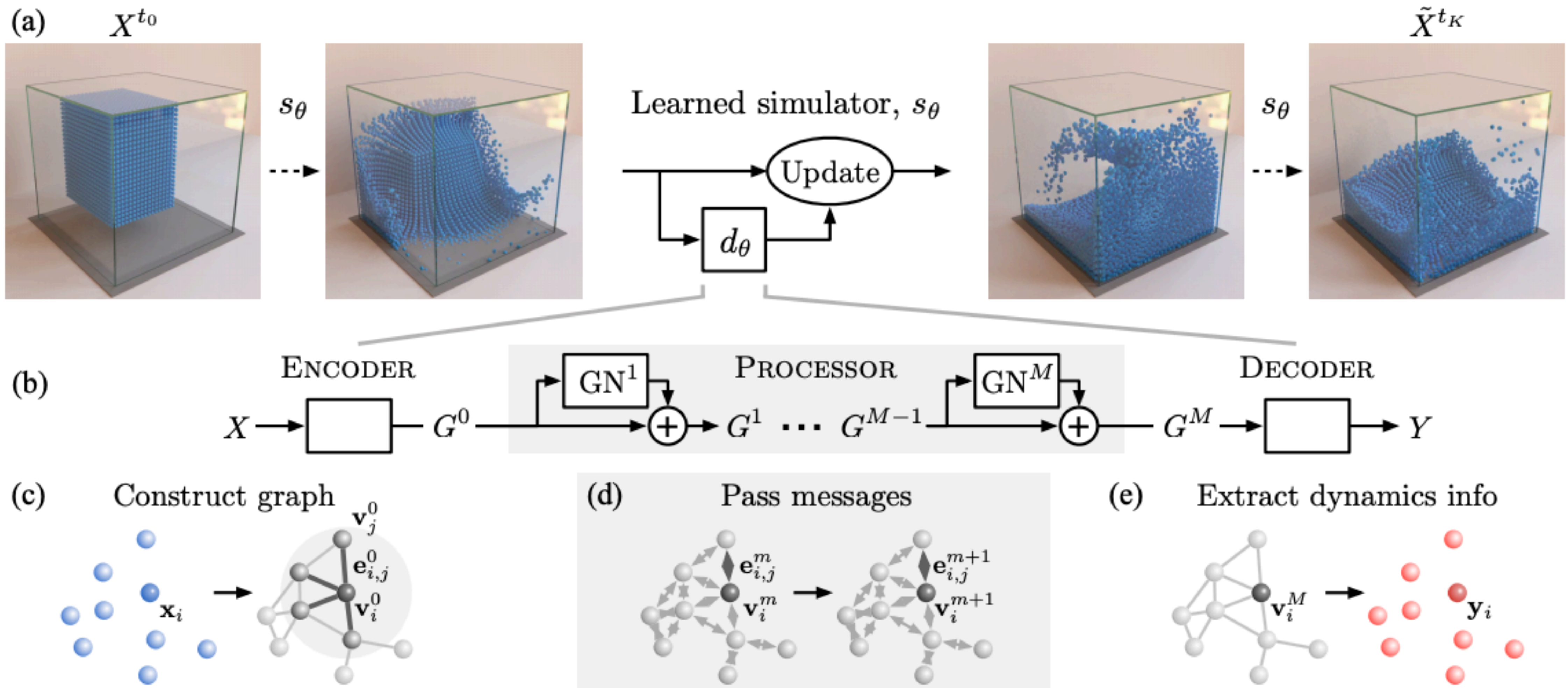
Learning to Simulate Complex Physics with Graph Networks

Alvaro Sanchez-Gonzalez [*1] **Jonathan Godwin** [*1] **Tobias Pfaff** [*1] **Rex Ying** [*2] **Jure Leskovec** [2]
Peter W. Battaglia [1]



Represent objects as graphs of particles and scenes as graph of all the particles from all objects.

- Q: Why? How are particle nodes different than object nodes?
- A: They do not need to capture appearance information only particles displacement! Appearances of particles stays constant over time, while appearance of objects changes: the appearance of the water changes, but the appearance of each of its particles did not. The shape of the object/material is captured simply by the particle graph (the location of its nodes).



- Input: particle velocities of the last 5 time steps, output: particle acceleration.
- Train for single step prediction.
- Handle error accumulation during unrolling by injecting noise in particle velocities during training.
- Q: how can we encode particle locations?
- A:Edges encode relative distances between two particles, no absolute position, else the neural net would not be translation invariant
- Multiple rounds of message passing: necessary to transmit the interaction across the graph. Each round has node and edge weights that are different.

Generalization