Carnegie Mellon
School of Computer Science

Deep Reinforcement Learning and Control

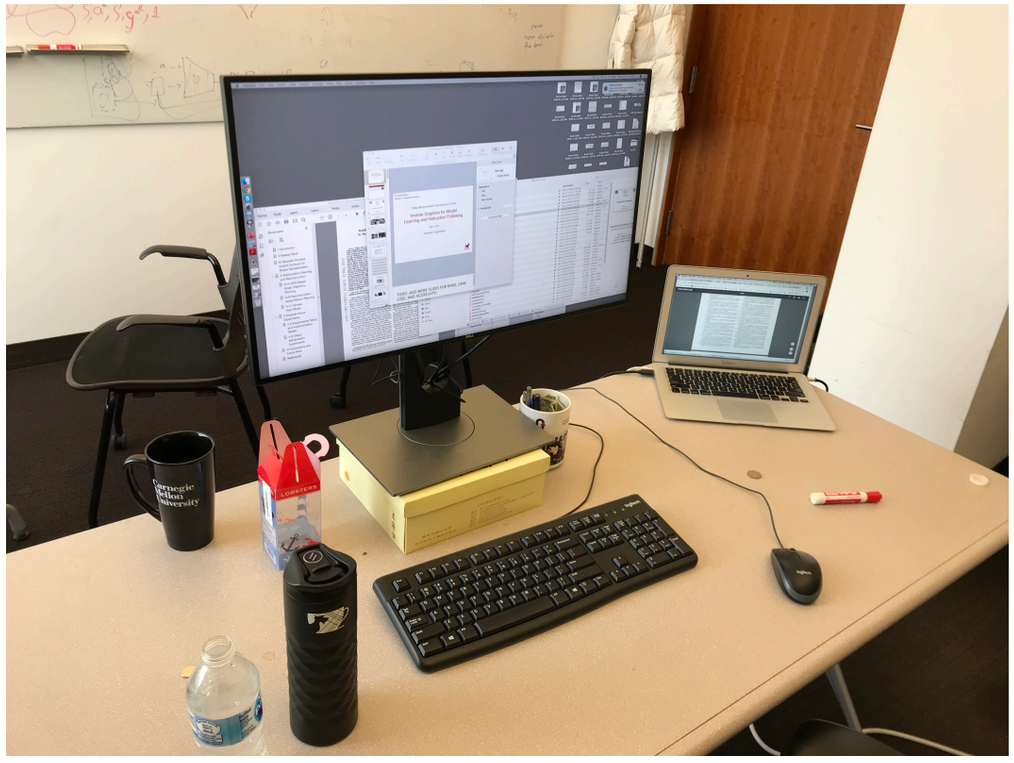
State representation learning for RL from pixels

Spring 2021, 10-403

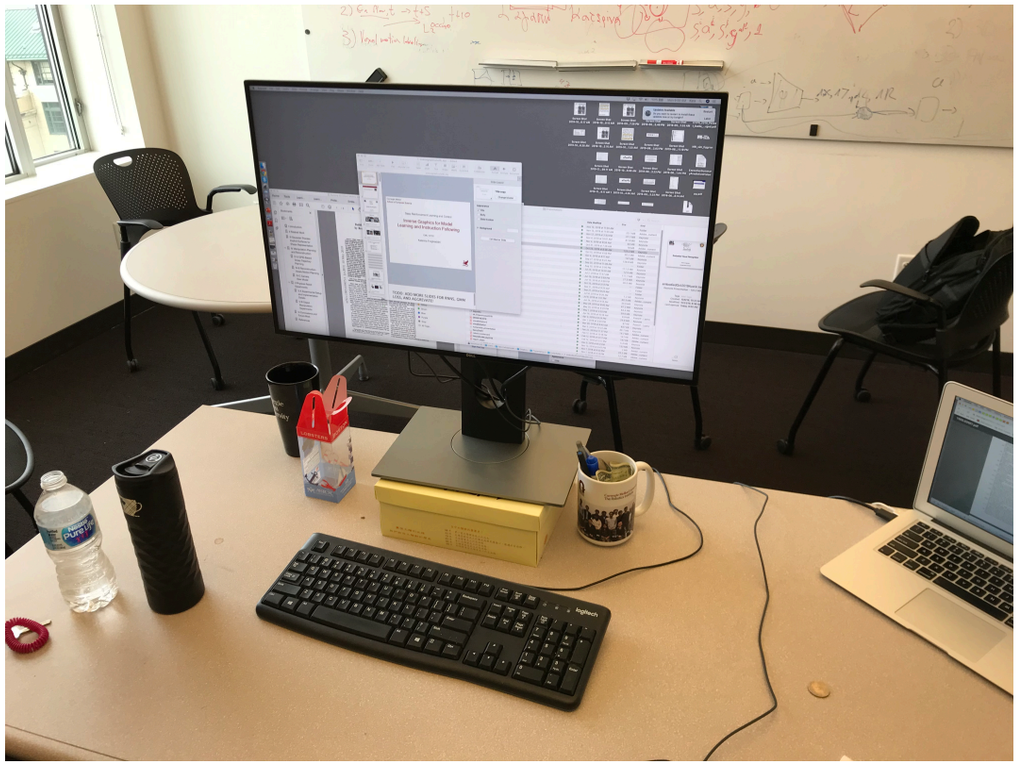
Katerina Fragkiadaki

RL from pixels

- Reinforcement learning from high dimensional observations (raw pixels) is sample-inefficient
- If the state information is present in the pixel data, then we should be able to learn representations that extract the relevant state information.
- It may be possible to learn from pixels as fast as from state given the right representation

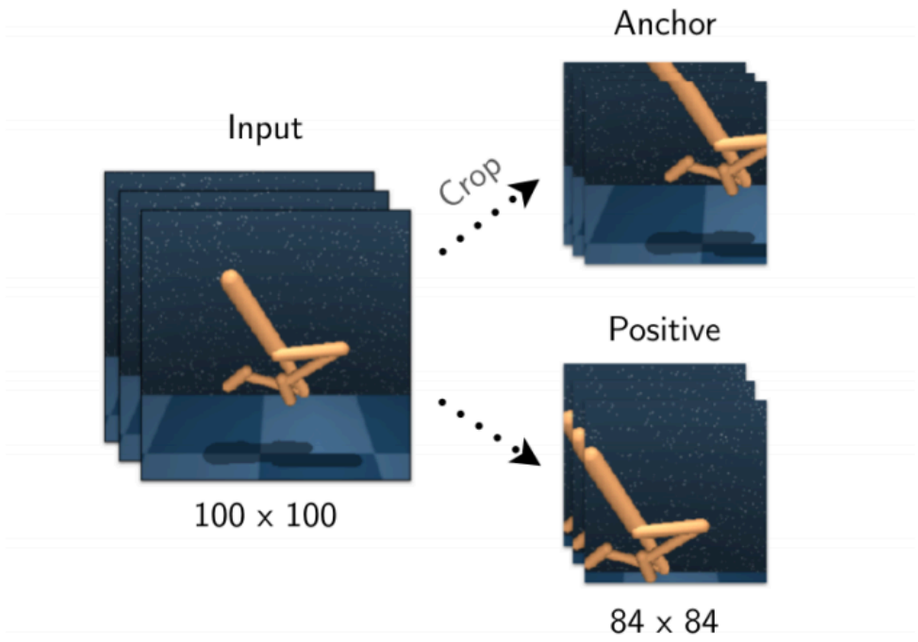


2 pictures depict identical scenes, and one picture depicts a different scene.



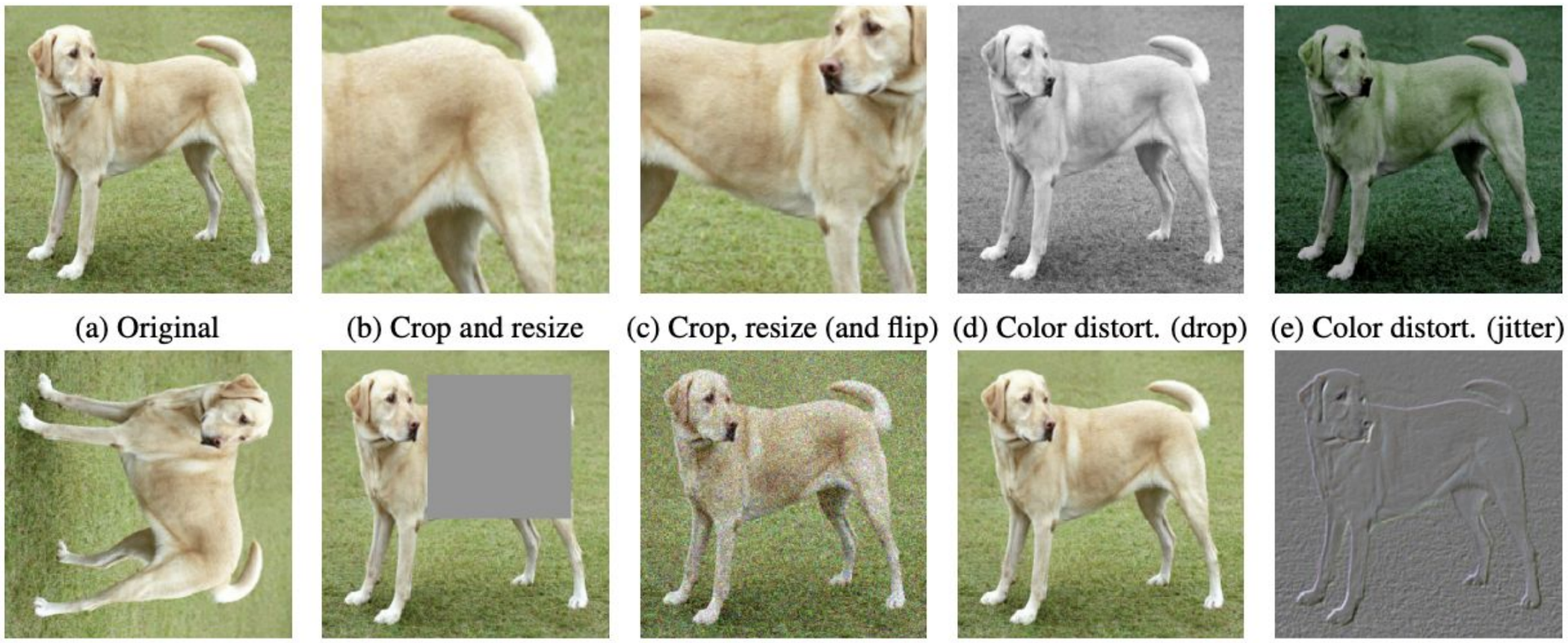
Instance discrimination

The current SOTA objective for self-supervised 2D representation learning for images!



The anchor and positive observations are two different **augmentations** of the same image while negatives come from other images.

Image Augmentations

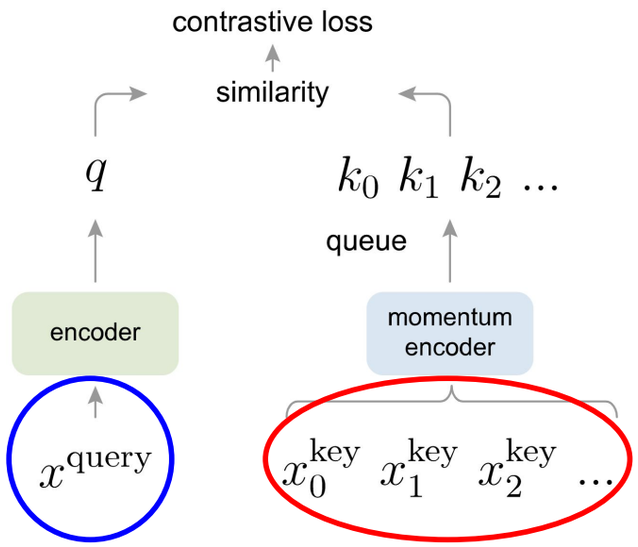


(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

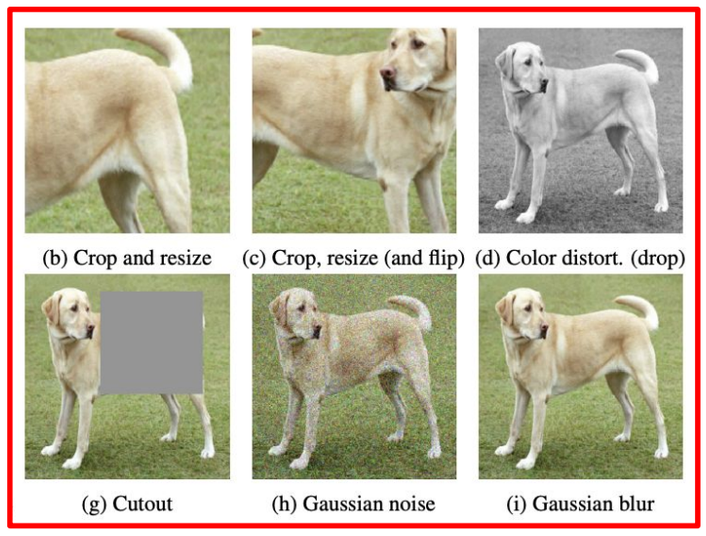
(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Self-supervision with instance discrimination

MoCo

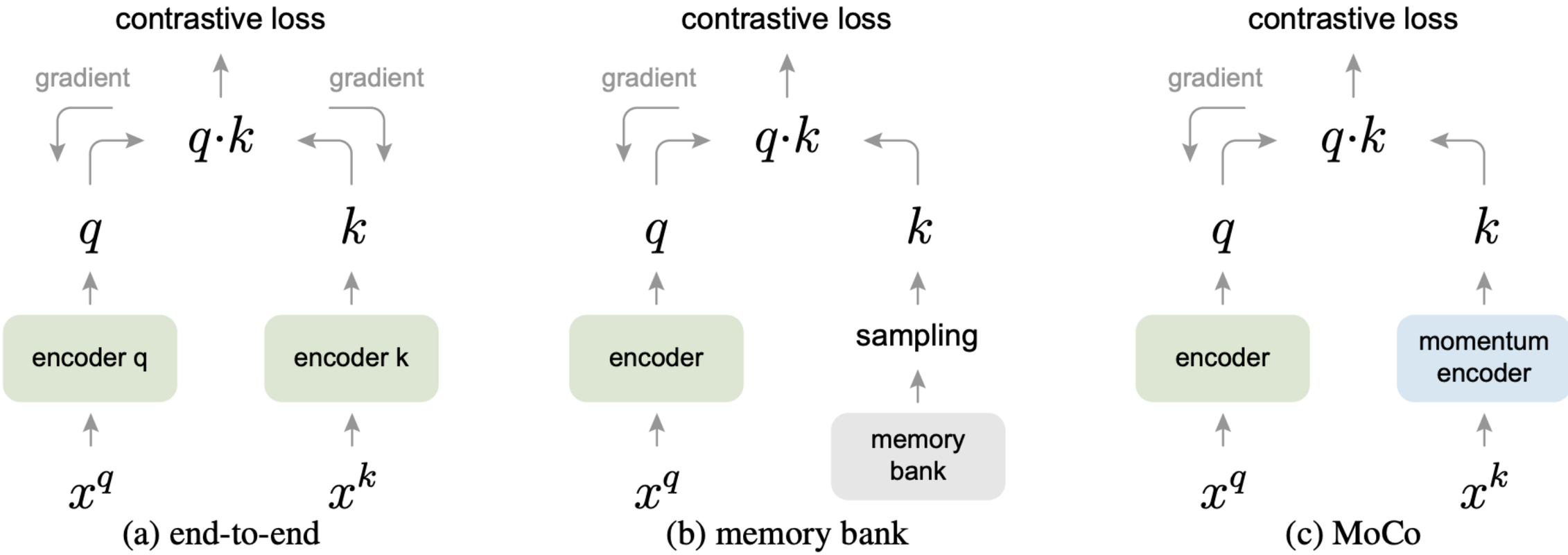


$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$



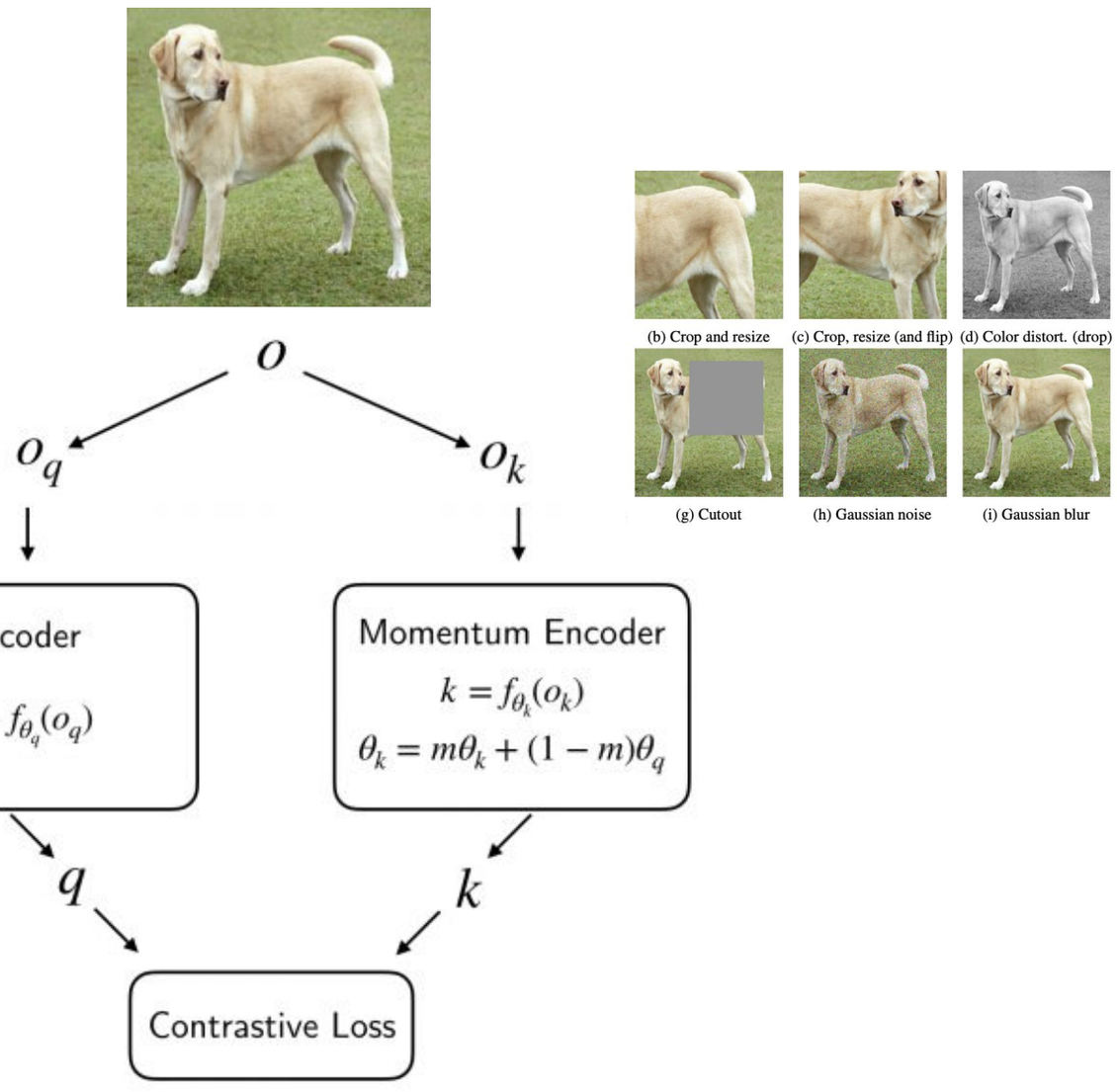
Momentum Contrast for Unsupervised Visual Representation Learning

Self-supervision with instance discrimination

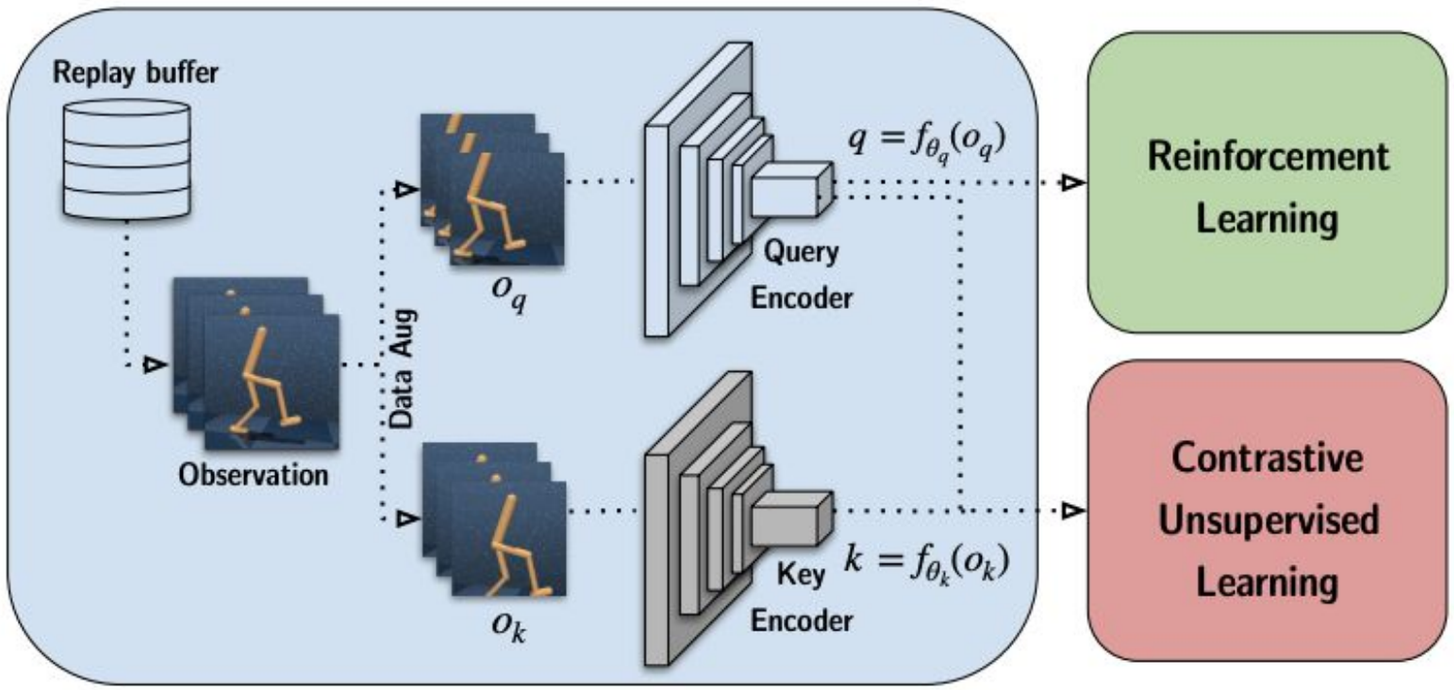


$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$

Momentum Contrast for Unsupervised Visual Representation Learning



CURL: Contrastive Unsupervised Representations for Reinforcement Learning



Positives = different random crops of the same observation

Negatives = Different observation

Loss = InfoNCE

$$\mathcal{L}_q = \log \frac{\exp(q^T W k_+)}{\exp(q^T W k_+) + \sum_{i=0}^{K-1} \exp(q^T W k_i)}$$

CURL: Contrastive Unsupervised Representations for Reinforcement Learning

Results

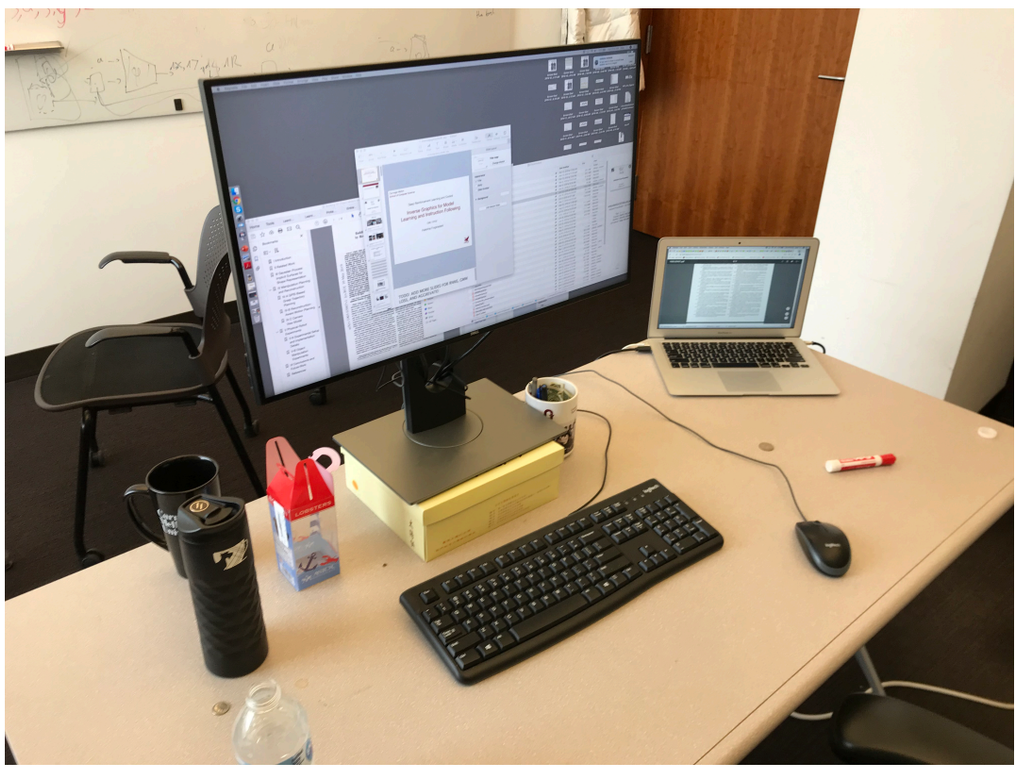
500K step scores	CURL	PlaNet	Dreamer	SAC+AE	SLACv1	Pixel SAC	State SAC
Finger, spin	**926 ± 45**	561 ± 284	796 ± 183	884 ± 128	673 ± 92	179 ± 166	923 ± 21
Cartpole, swingup	**841 ± 45**	475 ± 71	762 ± 27	735 ± 63	-	419 ± 40	848 ± 15
Reacher, easy	**929 ± 44**	210 ± 390	793 ± 164	627 ± 58	-	145 ± 30	923 ± 24
Cheetah, run	518 ± 28	305 ± 131	570 ± 253	550 ± 34	**640 ± 19**	197 ± 15	795 ± 30
Walker, walk	**902 ± 43**	351 ± 58	897 ± 49	847 ± 48	842 ± 51	42 ± 12	948 ± 54
Ball in cup, catch	**959 ± 27**	460 ± 380	879 ± 87	794± 58	852 ± 71	312± 63	974 ± 33
100K step scores							
Finger, spin	**767 ± 56**	136 ± 216	341 ± 70	740 ± 64	693 ± 141	179 ± 66	811±46
Cartpole, swingup	**582±146**	297±39	326±27	311±11	-	419±40	835±22
Reacher, easy	**538±233**	20±50	314±155	274±14	-	145±30	746±25
Cheetah, run	299 ±48	138±88	235± 137	267±24	**319±56**	197±15	616±18
Walker, walk	**403±24**	224±48	277±12	394±22	361±73	42±12	891±82
Ball in cup, catch	**769 ± 43**	0 ± 0	246 ± 174	391± 82	512 ± 110	312± 63	746±91

Game	Human	Random	Rainbow	SimPLe	OTRainbow	Eff. Rainbow	CURL
Alien	7127.7	227.8	318.7	616.9	**824.7**	739.9	558.2
Amidar	1719.5	5.8	32.5	88.0	82.8	**188.6**	142.1
Assault	742.0	222.4	231	527.2	351.9	431.2	**600.6**
Asterix	8503.3	210.0	243.6	**1128.3**	628.5	470.8	734.5
Bank Heist	753.1	14.2	15.55	34.2	**182.1**	51.0	131.6
Battle Zone	37187.5	2360.0	2360.0	5184.4	4060.6	10124.6	**14870.0**
Boxing	12.1	0.1	-24.8	**9.1**	2.5	0.2	1.2
Breakout	30.5	1.7	1.2	**16.4**	9.84	1.9	4.9
Chopper Command	7387.8	811.0	120.0	**1246.9**	1033.33	861.8	1058.5
crazy_climber	35829.4	10780.5	2254.5	**62583.6**	21327.8	16185.3	12146.5
demon_attack	1971.0	152.1	163.6	208.1	711.8	508.0	**817.6**
freeway	29.6	0.0	0.0	20.3	25.0	**27.9**	26.7
frostbite	4334.7	65.2	60.2	254.7	231.6	866.8	**1181.3**
gopher	2412.5	257.6	431.2	771.0	**778.0**	349.5	669.3
hero	30826.4	1027.0	487	2656.6	6458.8	**6857.0**	6279.3
jamesbond	302.8	29.0	47.4	125.3	112.3	301.6	**471.0**
kangaroo	3035.0	52.0	0.0	323.1	605.4	779.3	**872.5**
krull	2665.5	1598.0	1468	**4539.9**	3277.9	2851.5	4229.6
kung_fu_master	22736.3	258.5	0.	**17257.2**	5722.2	14346.1	14307.8
ms_pacman	6951.6	307.3	67	**1480.0**	941.9	1204.1	1465.5
Pong	14.6	-20.7	-20.6	**12.8**	1.3	-19.3	-16.5
Private eye	69571.3	24.9	0	58.3	100.0	97.8	**218.4**
Qbert	13455.0	163.9	123.46	**1288.8**	509.3	1152.9	1042.4
Road_Runner	7845.0	11.5	1588.46	5640.6	2696.7	**9600.0**	5661.0
seaquest	42054.7	68.4	131.69	**683.3**	286.92	354.1	384.5
Up_n_Down	11693.2	533.4	504.6	**3350.3**	2847.6	2877.4	2955.2

tl;dr It works better than everything else

``It is easy to infer the dynamics of a scene, to build an approximate mental simulator for it”

Chris Atkeson



``It is easy to infer the dynamics of a scene, to build an approximate mental simulator for it"

Chris Atkeson

This means you can do (without learning):

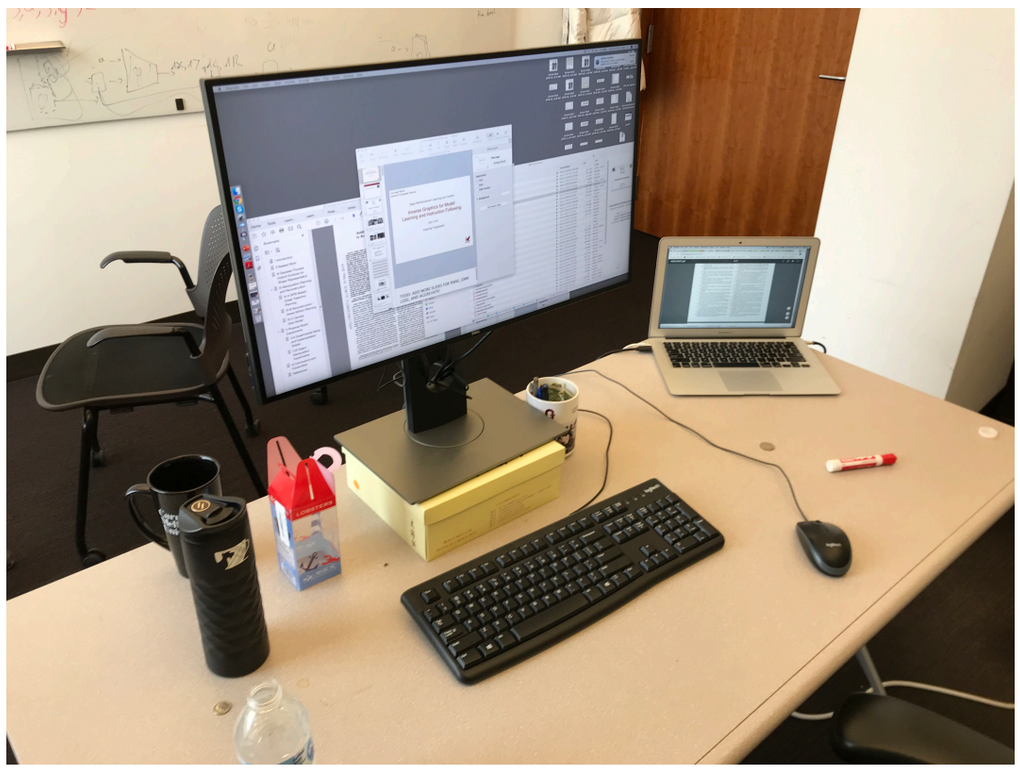
- *Object detection: infer where the objects are*
- *Free-space inference: find collision-free trajectories*
- *Affordance inference: imagine where objects can appear and where they cannot,*
- *Plan: find intermediate waypoints to take you half-way to your desired goals.*

What you cannot do (without learning):

- ***Plan/reason about contacts***

``It is easy to infer the dynamics of a scene, to build an approximate mental simulator for it”

Chris Atkeson



What is though trivial to infer from a 3 dimensional representation of the scene, it requires lots of labelled data to learn if you are in 2D…

Learning/reasoning about space from 2D images is hard because:

- There are projection artifacts: foreshortening (it’s not easy to know which object is close to which object and how much free space is there between them)
- There is no object permanence: objects disappear at occlusions
- Objects ``move” with camera motion
- Objects change size during camera zoom in / zoom out motion

SLAM

ORB-SLAM 2.0

- Scene and camera motion are disentangled
- Object permanence: objects do not disappear from the map as the camera moves around

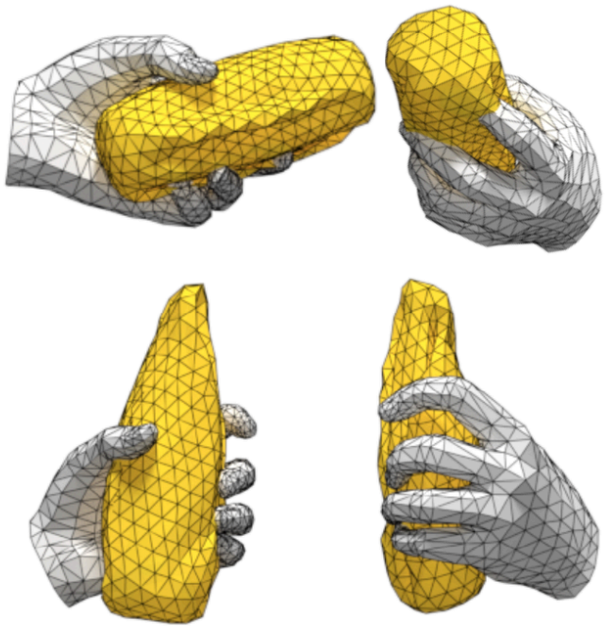
Self-driving uses 3D representations

Why don’t we try to map input images to 3D object models?

For self-driving we have done that, why not for object manipulation?

3D meshing the world

Why don’t we try to map input images to 3D meshes?



Do we know now how to place the bottle on the table?

*Beyond free-space, there is a lot of knowledge about contacts that 3D meshed versions of the world cannot **readily** provide, we need to learn those from 2D or 3D meshed data!*

(while we do not need to learn free space in general in 3D meshed data: it comes from free)

And self-driving cars care about free space and intended motion for objects

Learning Joint Reconstruction of Hands and Manipulated Objects, CVPR 2019

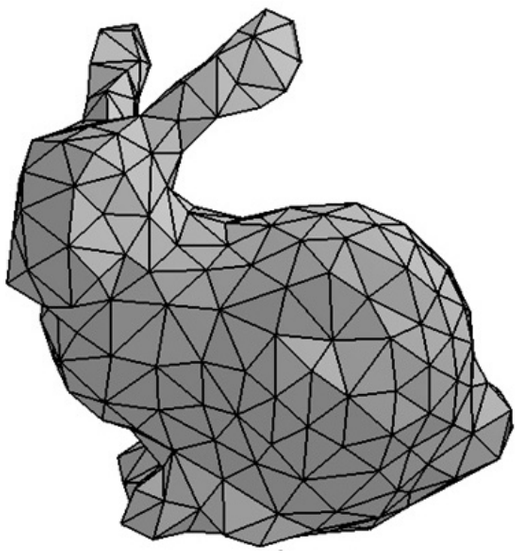
3D models are impossible and unnecessary

``Internal world models which are complete representations of the external environment, besides being impossible to obtain, are not at all necessary for agents to act in a competent manner.”

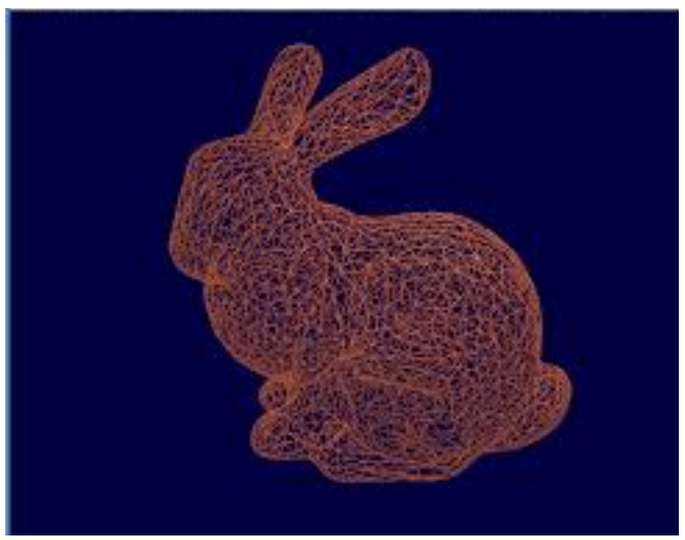
Intelligence without reason, IJCAI, Rodney Brooks (1991)

3D models are impossible and unecessary

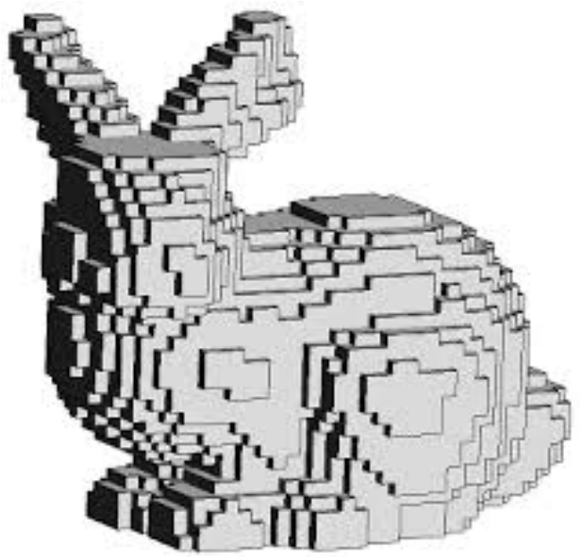
3D mesh



3D pointcloud



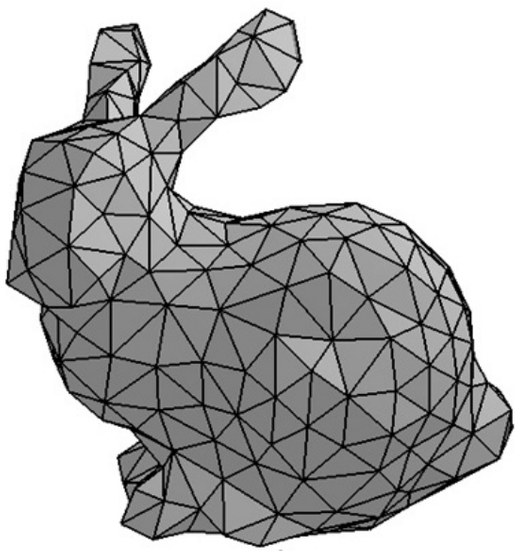
3D voxel occupancy



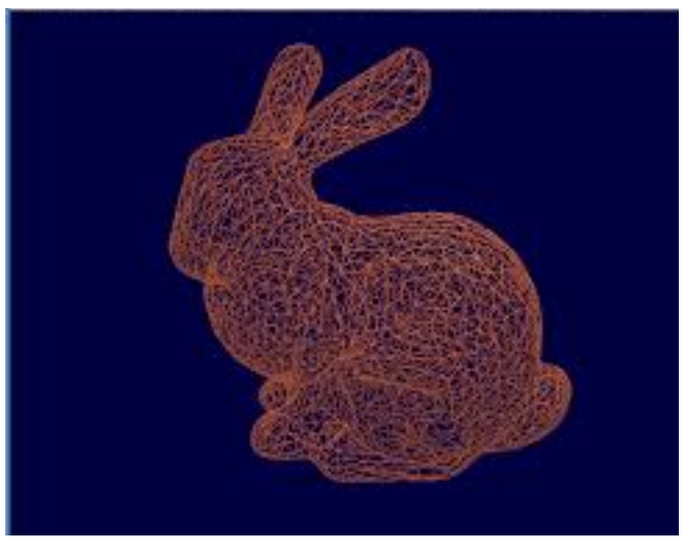
- They do not optimize the right end task: catching the rabbit.
- They optimize for 3D reconstruction quality
- 2D image to 3D mesh reconstruction requires a lot of human labelled data

What we need

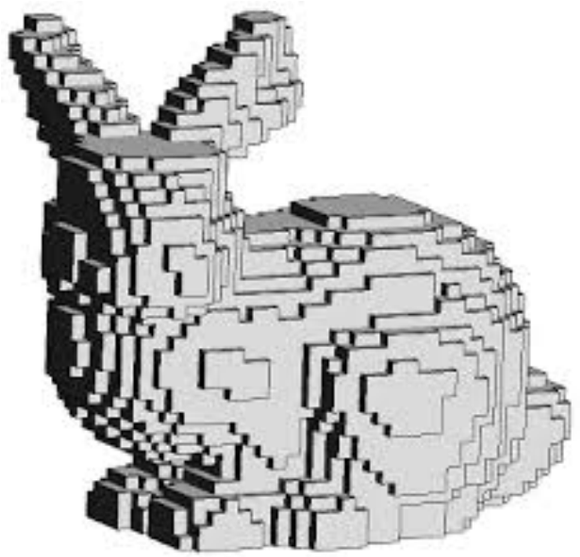
3D mesh



3D pointcloud



3D voxel occupancy



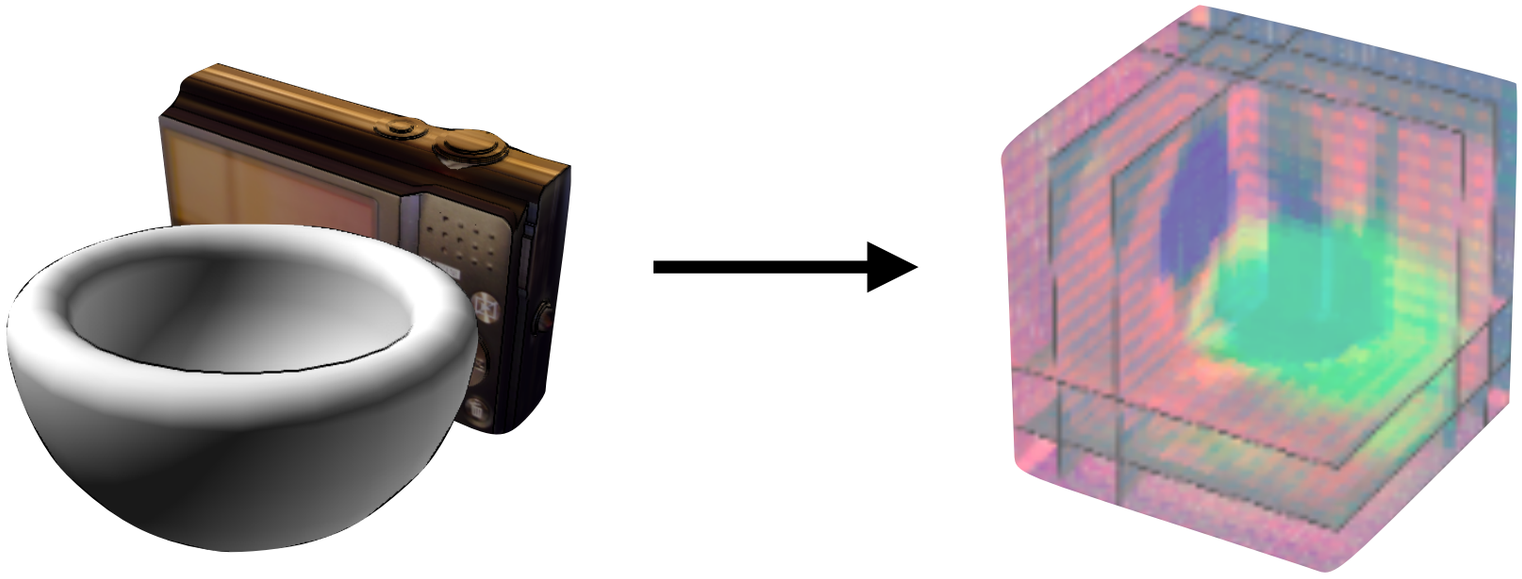
- We need to link 3D representations with the end-task of behaviour learning.
- We should be able to learn without any human supervision.

SLAM

ORB-SLAM 2.0

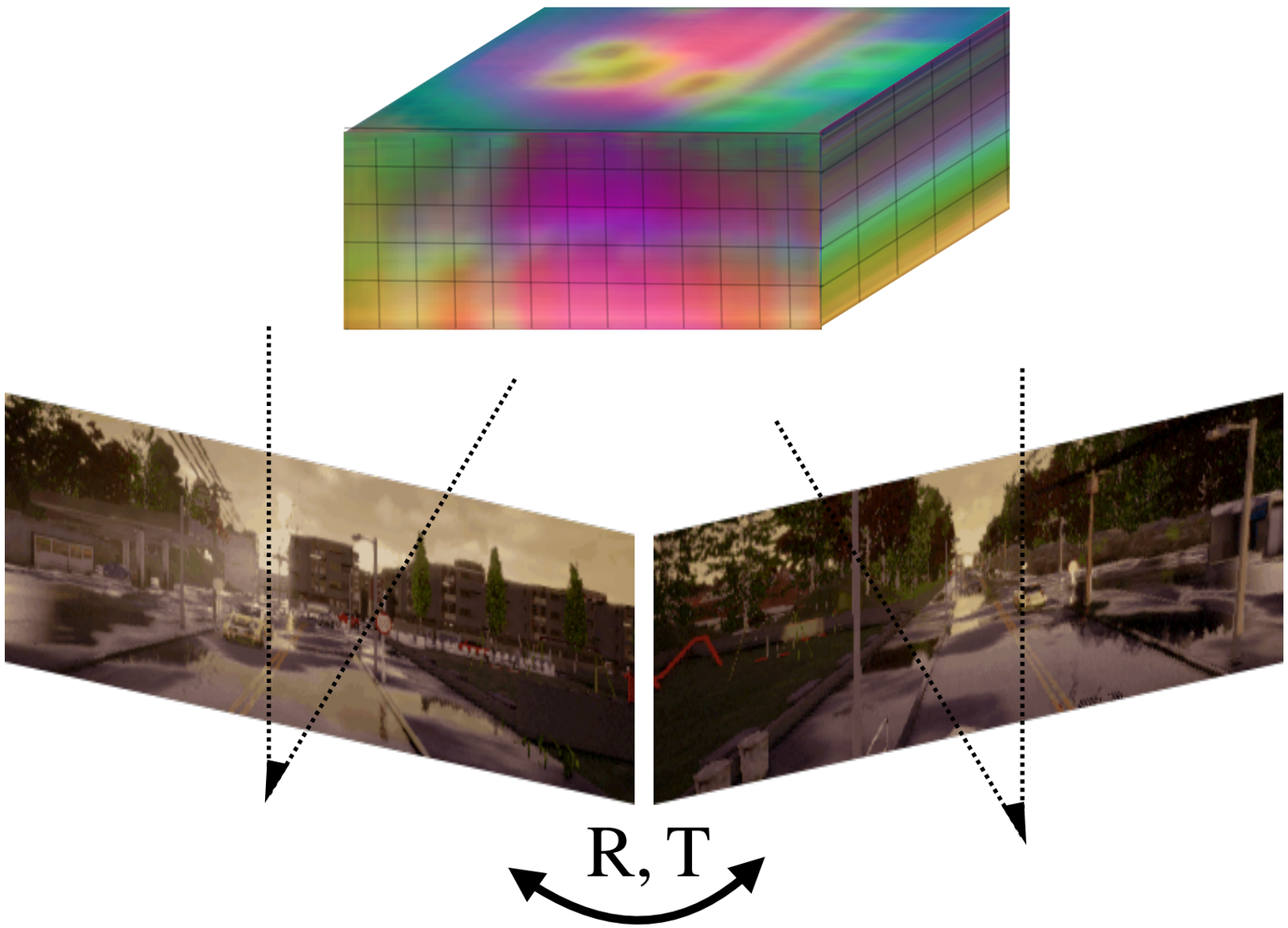
- SLAM disentangles a video into scene appearance (point cloud map) and camera motion
- Objects persist in the pointcloud map

3D feature representations



$$\mathbf{M} \in \mathbb{R}^{H \times W \times D \times C}$$

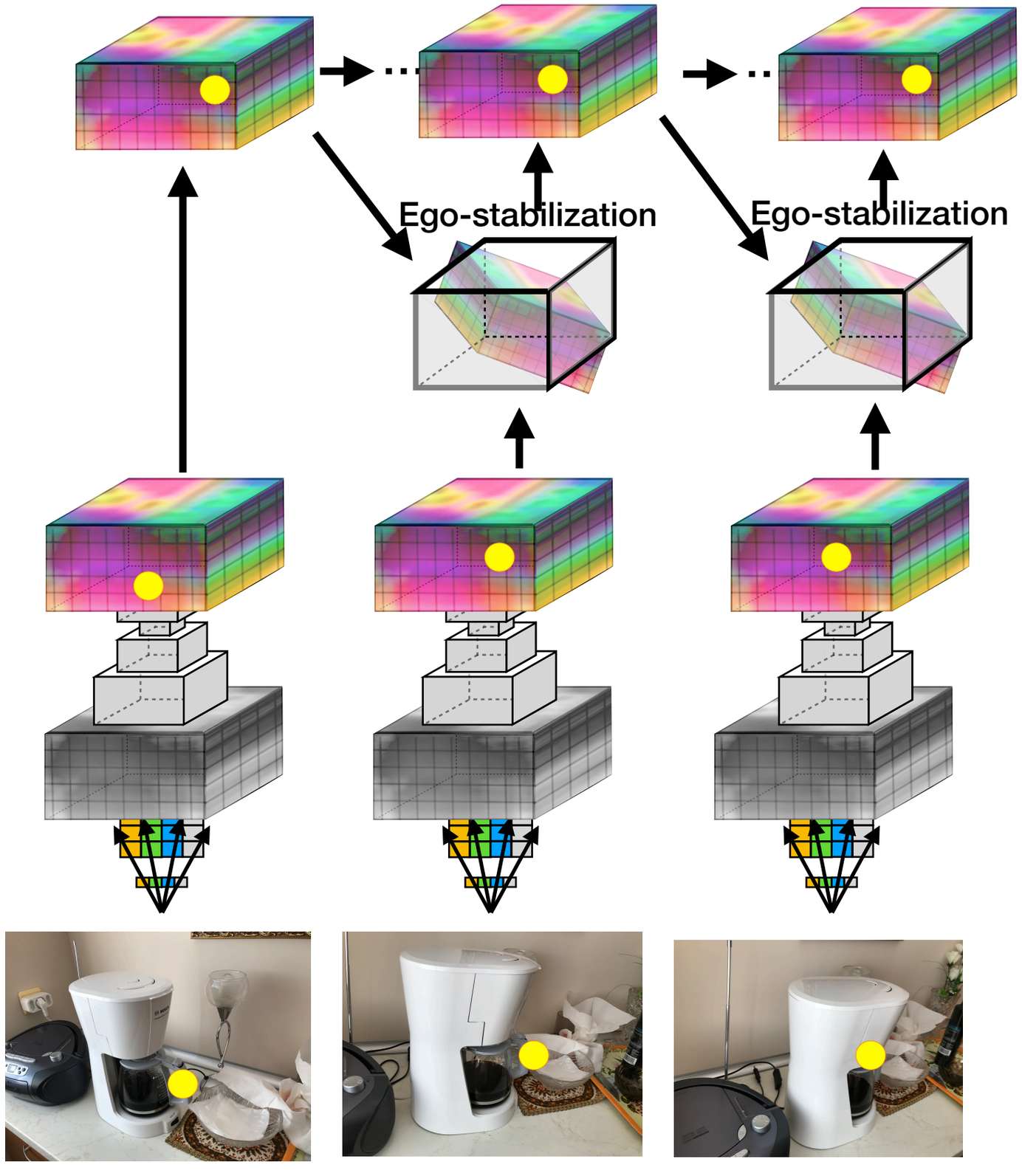
Geometry-Aware Recurrent Networks



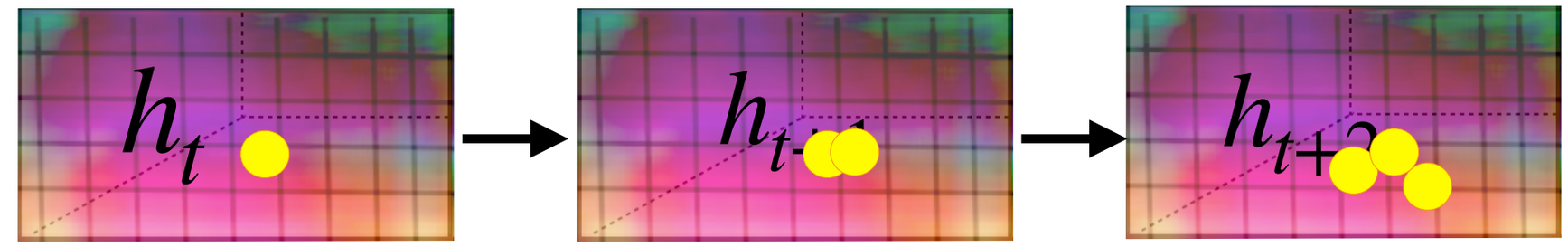
- 3-dimensional latent state
- Egomotion-stable latent state updates

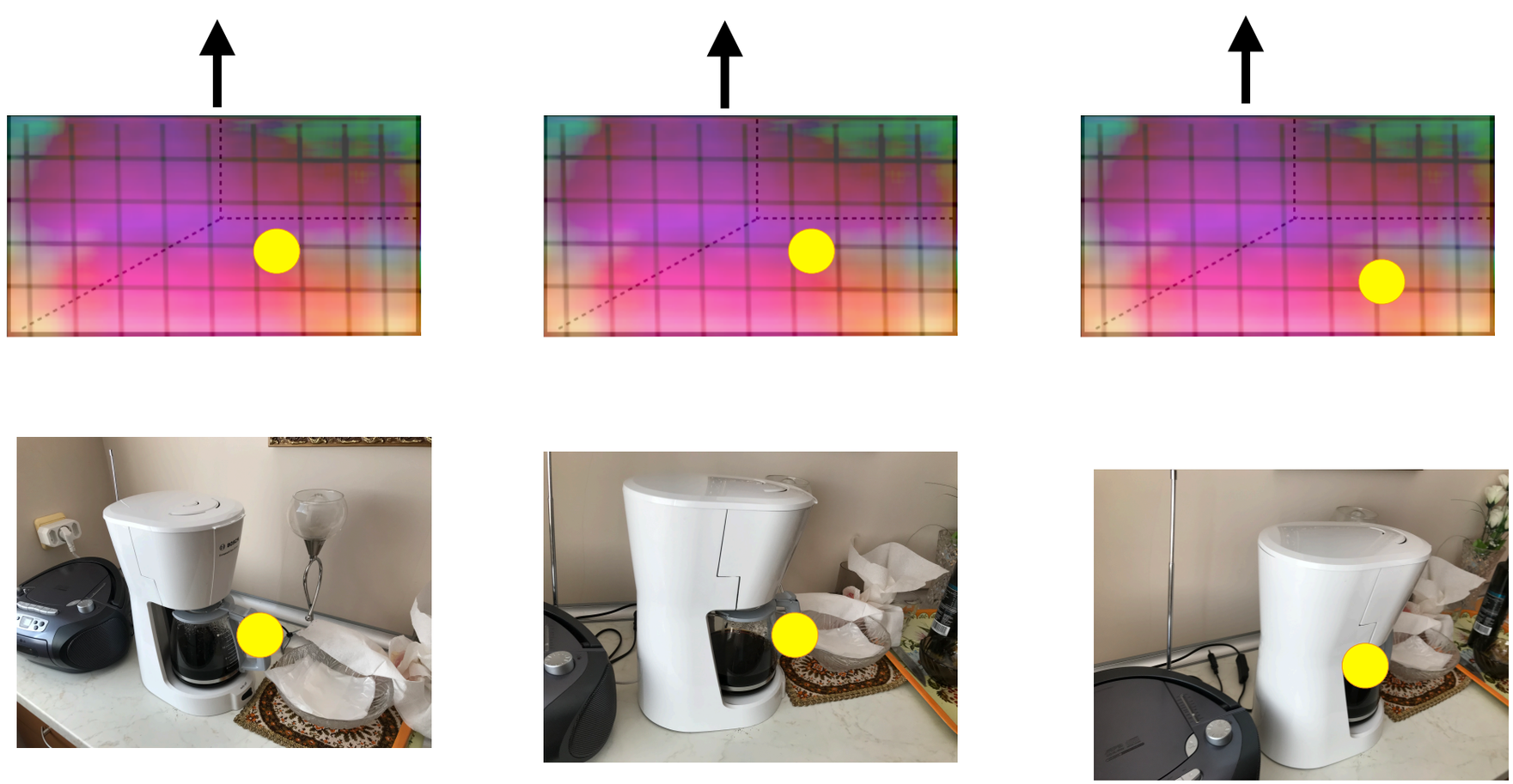
Learning spatial common sense with geometry-aware recurrent networks, Tung et al. CVPR 2019

Geometry-Aware Recurrent Networks

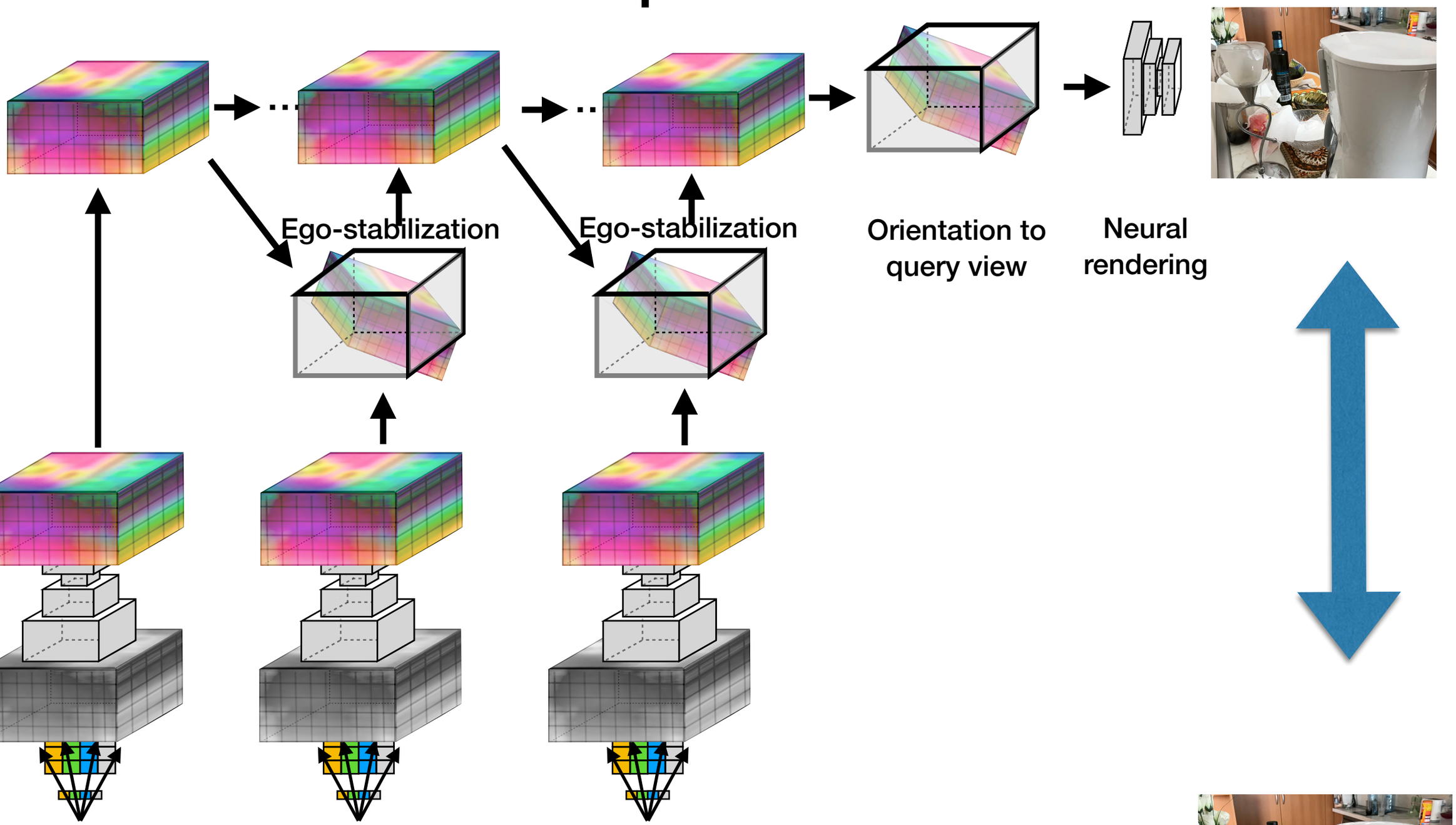


2D RNNs (conv-LSTMs/GRUs)





View prediction



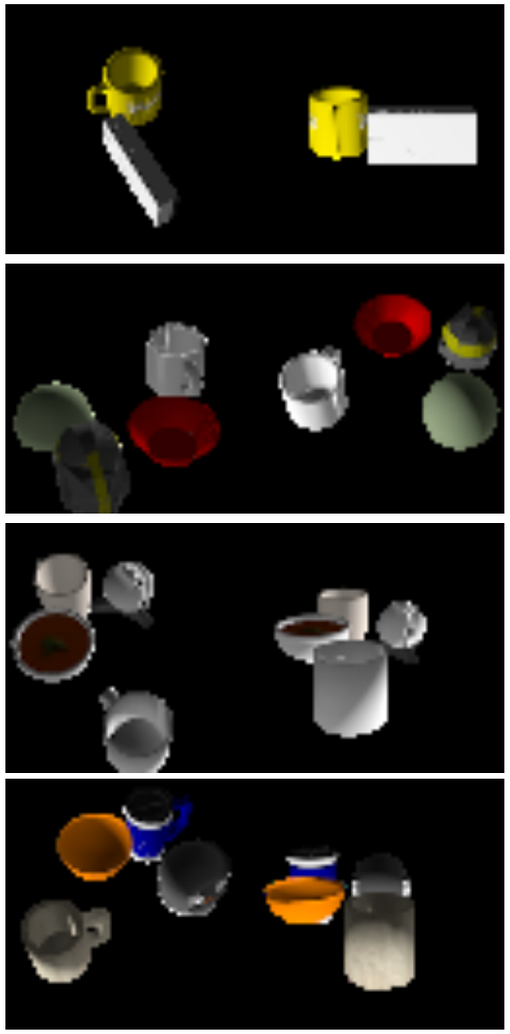
Strong generalization to scenes with more objects

Input views

[1] Neural scene representation and rendering DeepMind, Science, 2018

Strong generalization to scenes with more objects

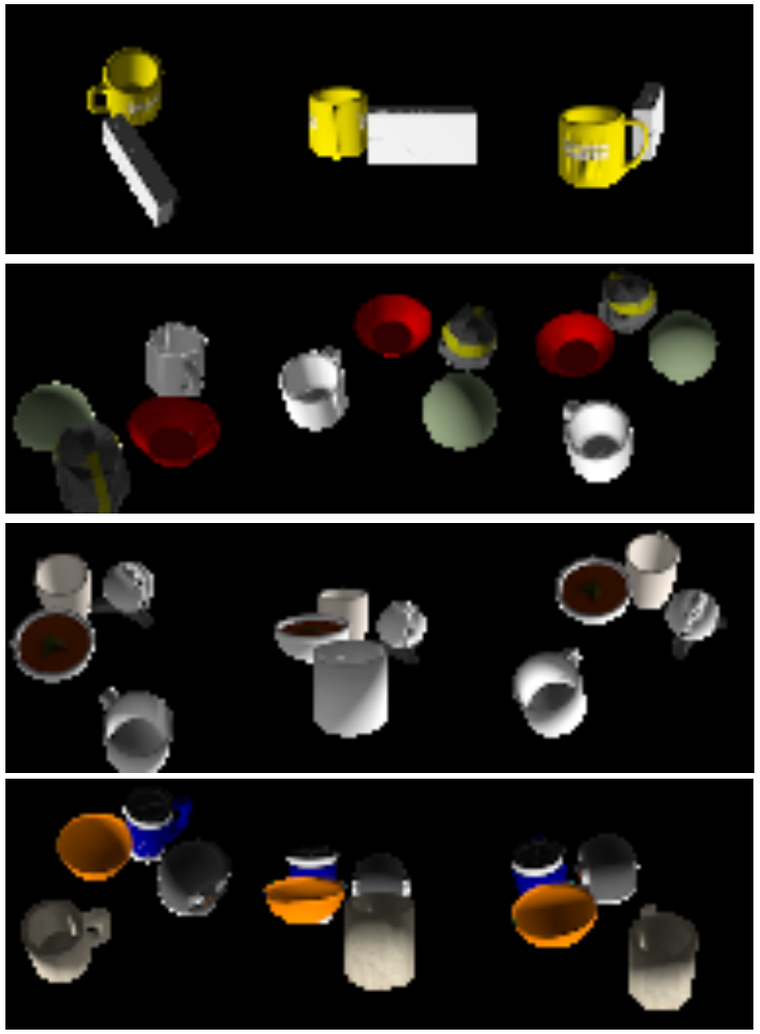
Input views



[1] Neural scene representation and rendering DeepMind, Science, 2018

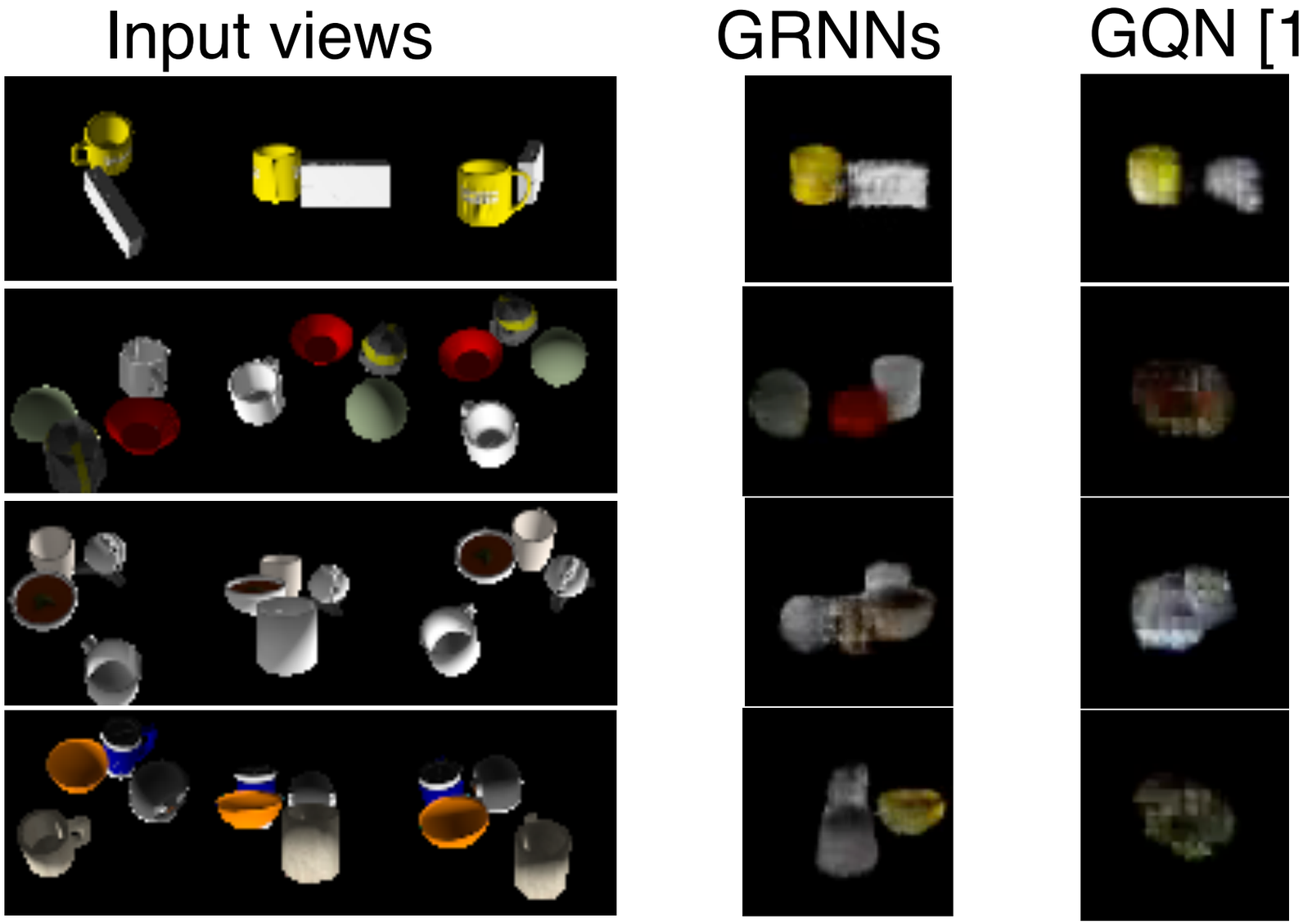
Strong generalization to scenes with more objects

Input views



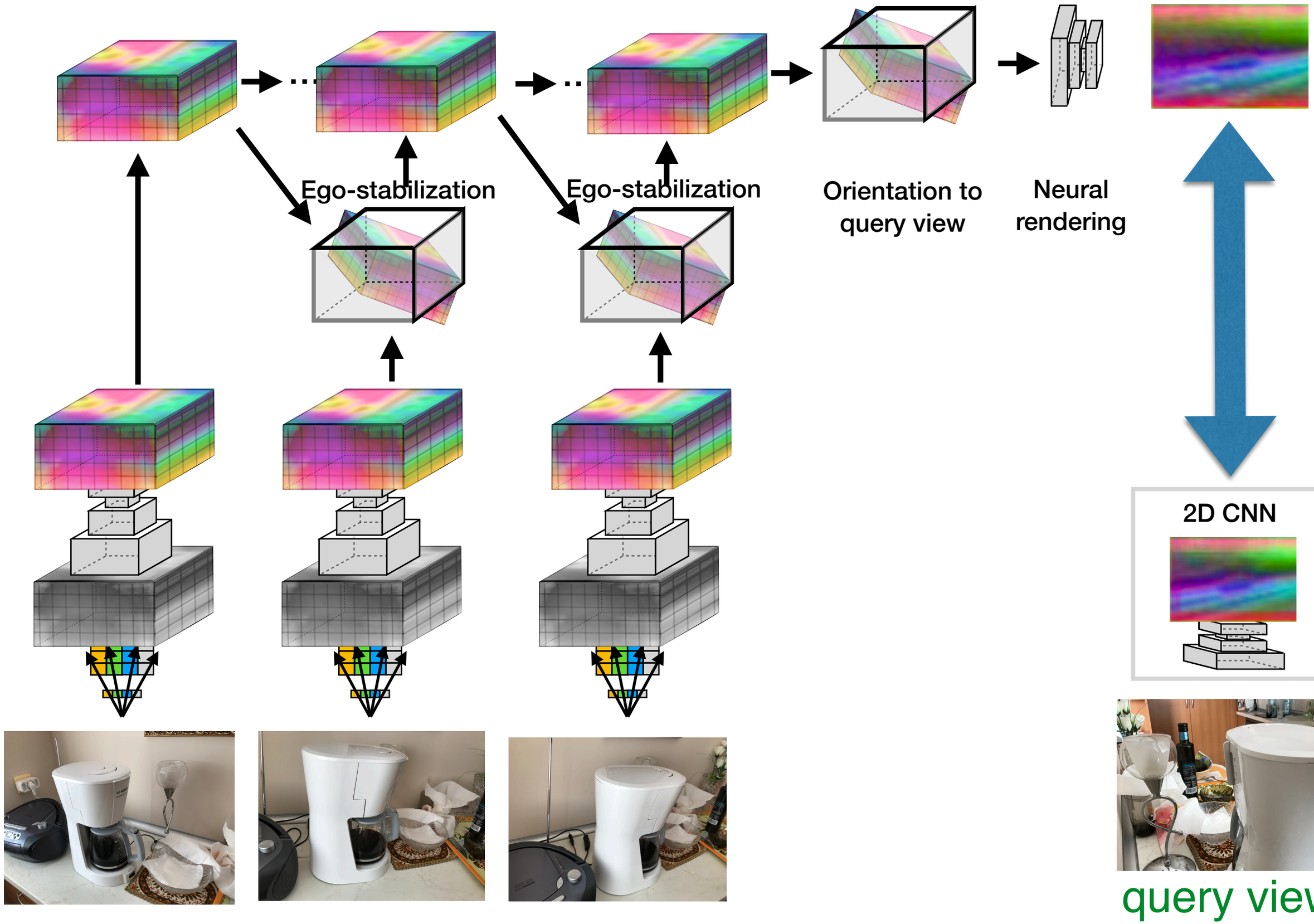
[1] Neural scene representation and rendering DeepMind, Science, 2018

Strong generalization to scenes with more objects

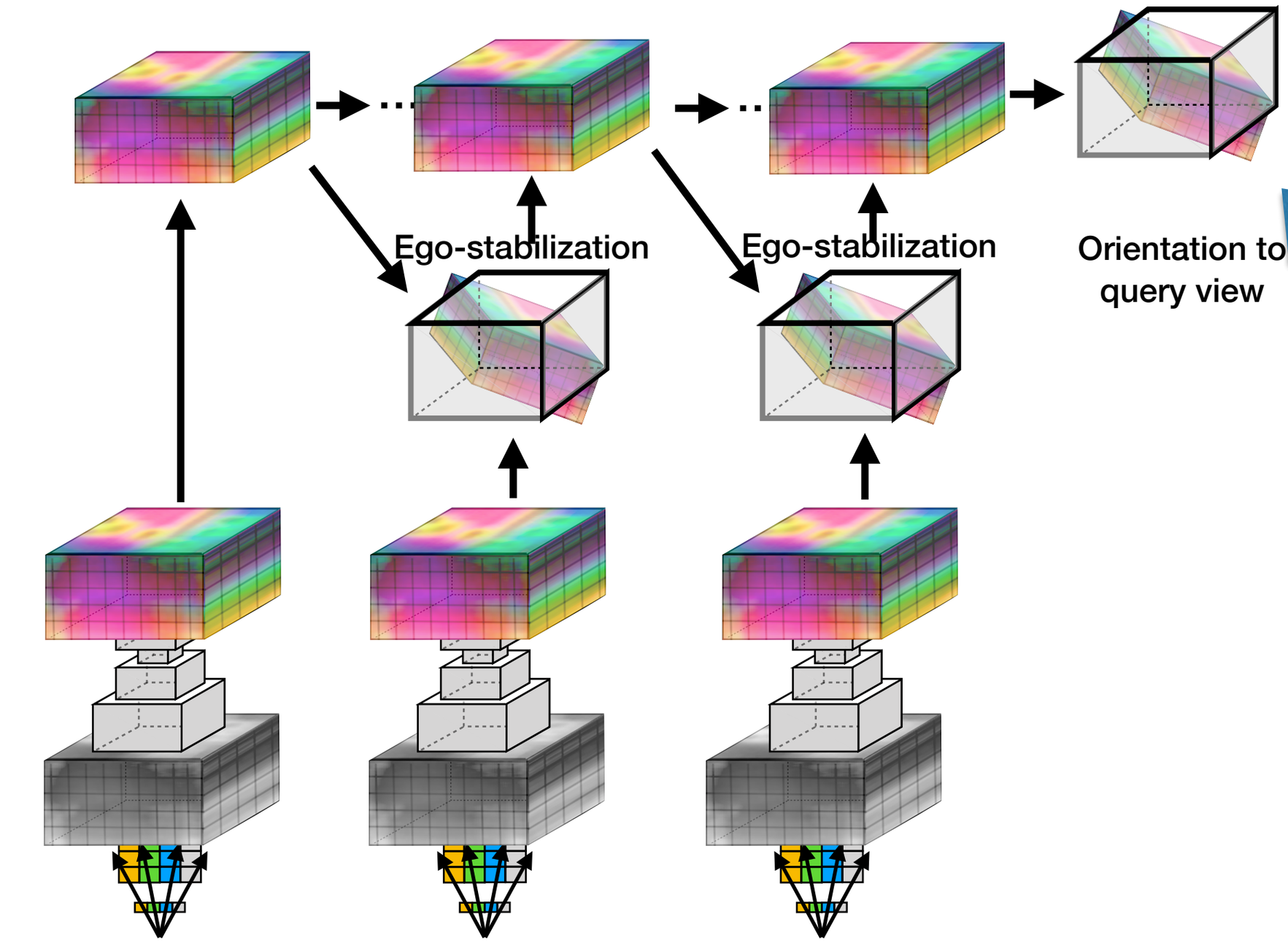


[1] Neural scene representation and rendering DeepMind, Science, 2018

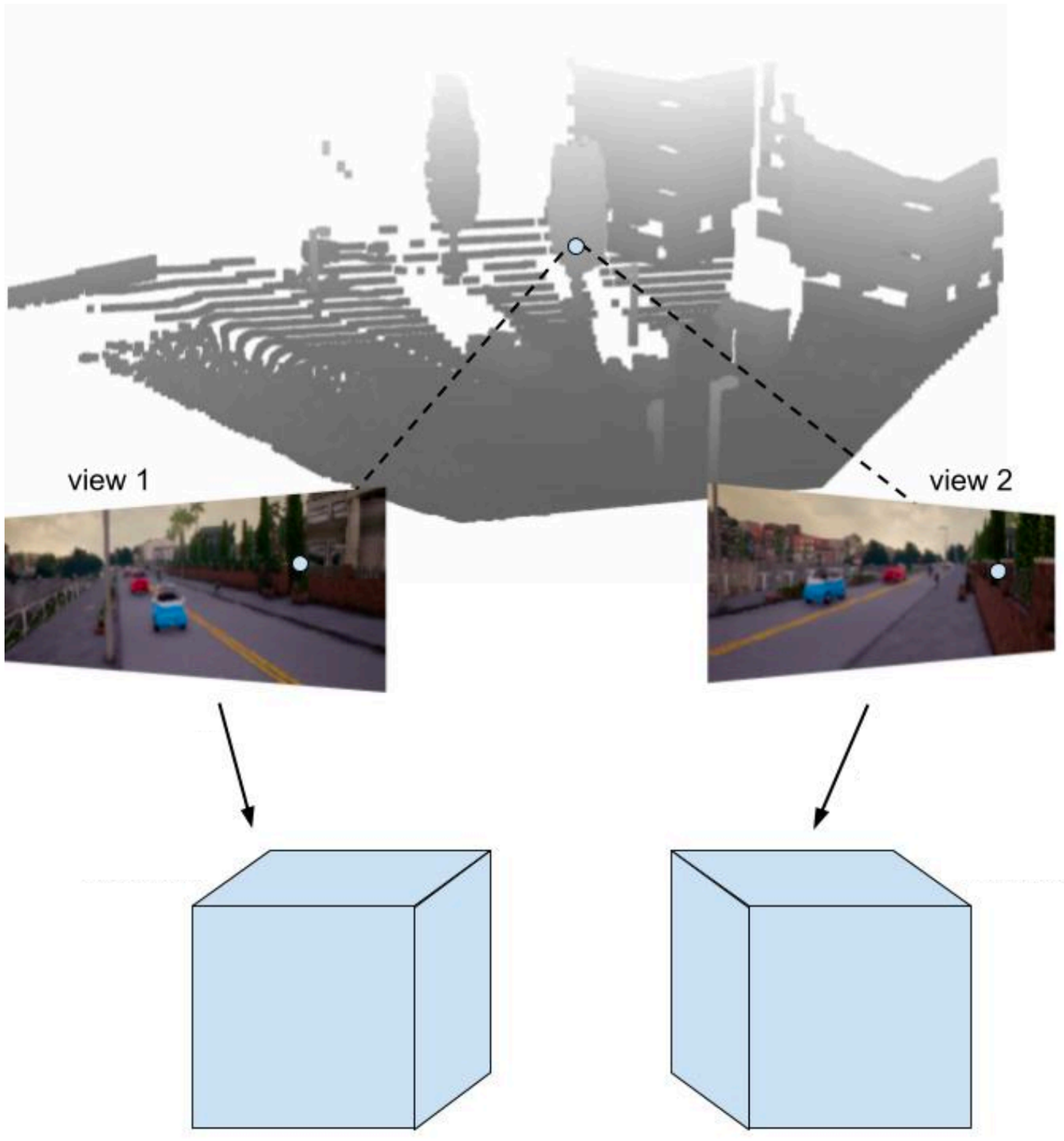
View contrastive prediction

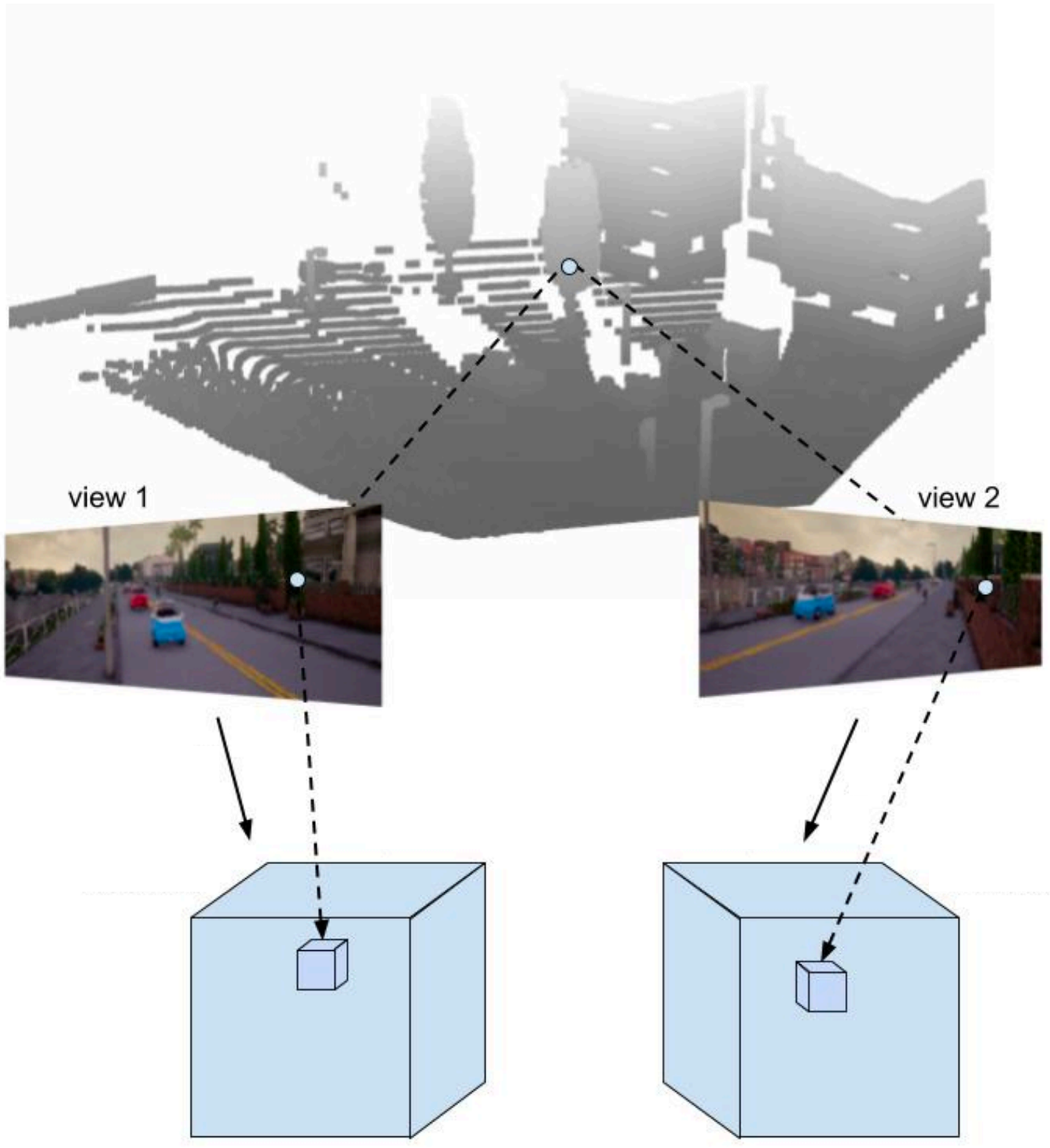


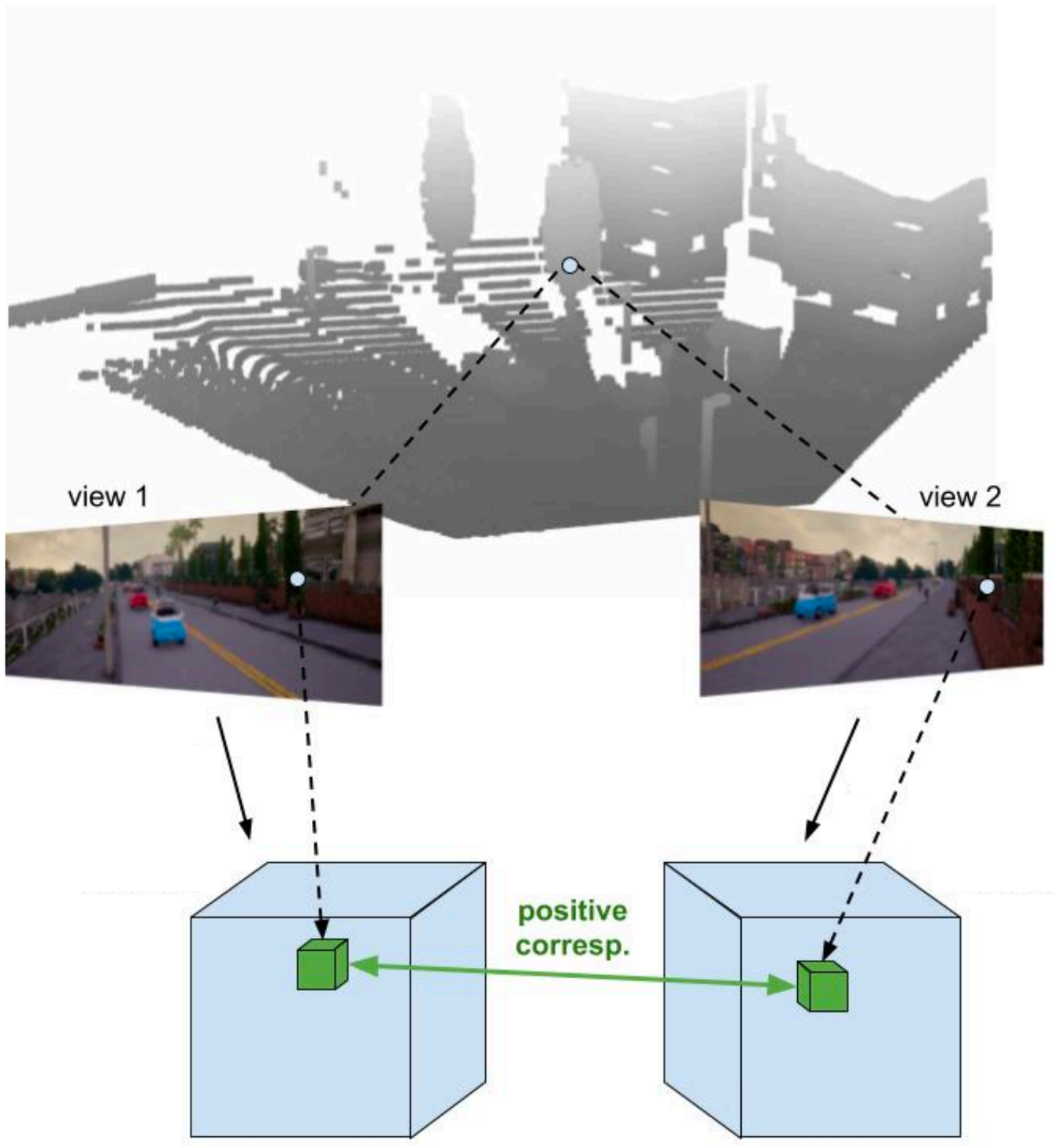
View contrastive prediction

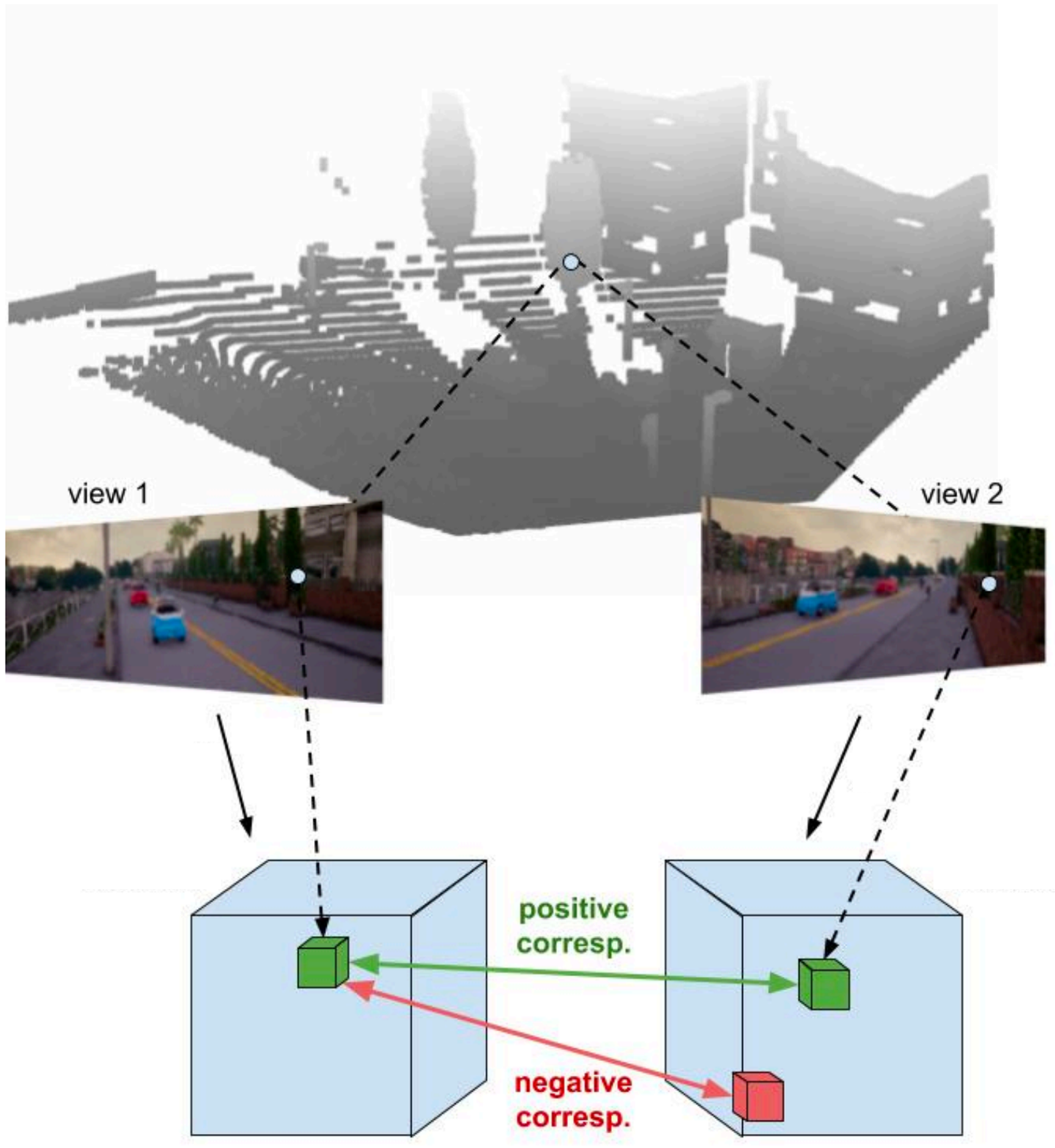


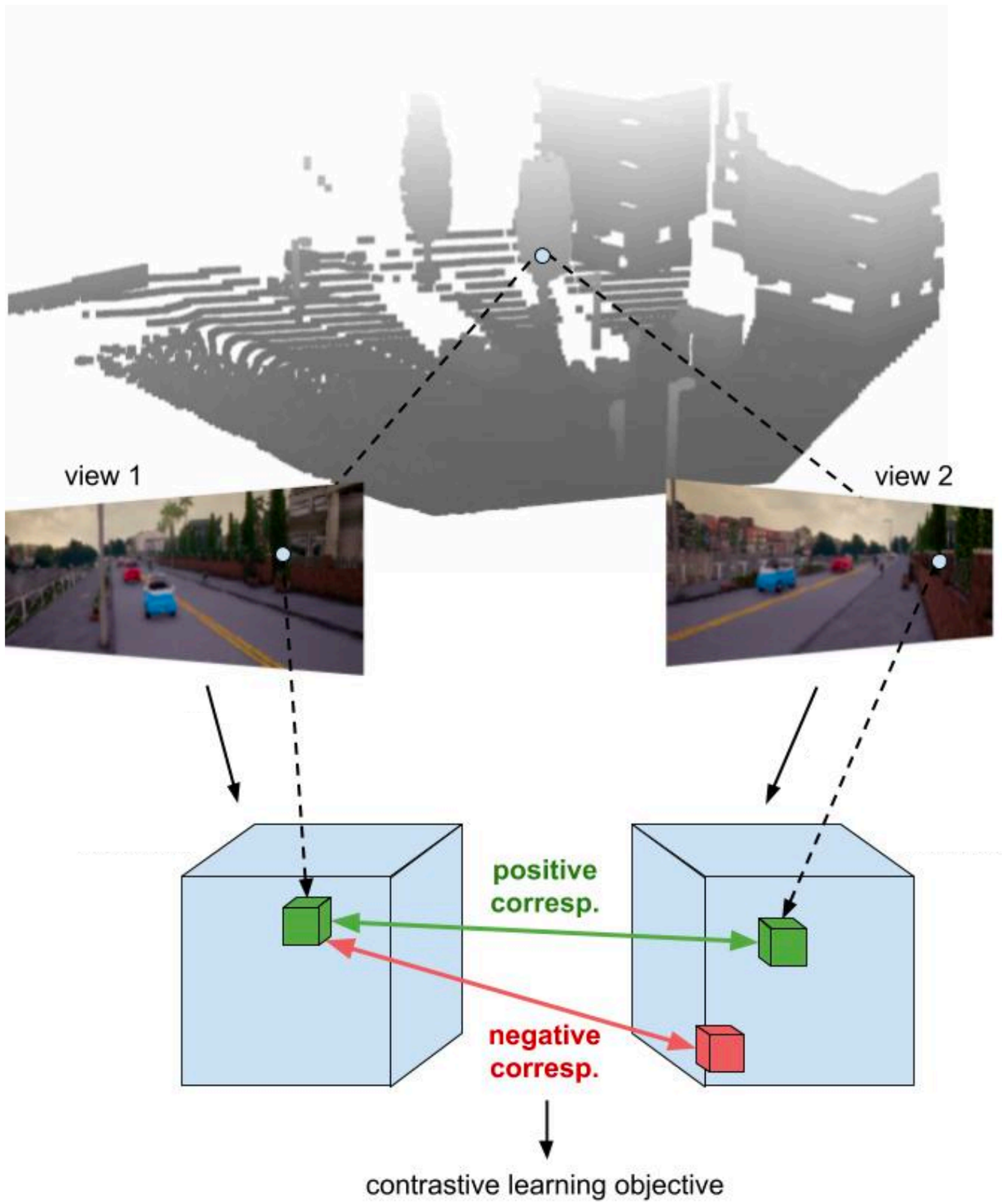
query view


view 1
view 2

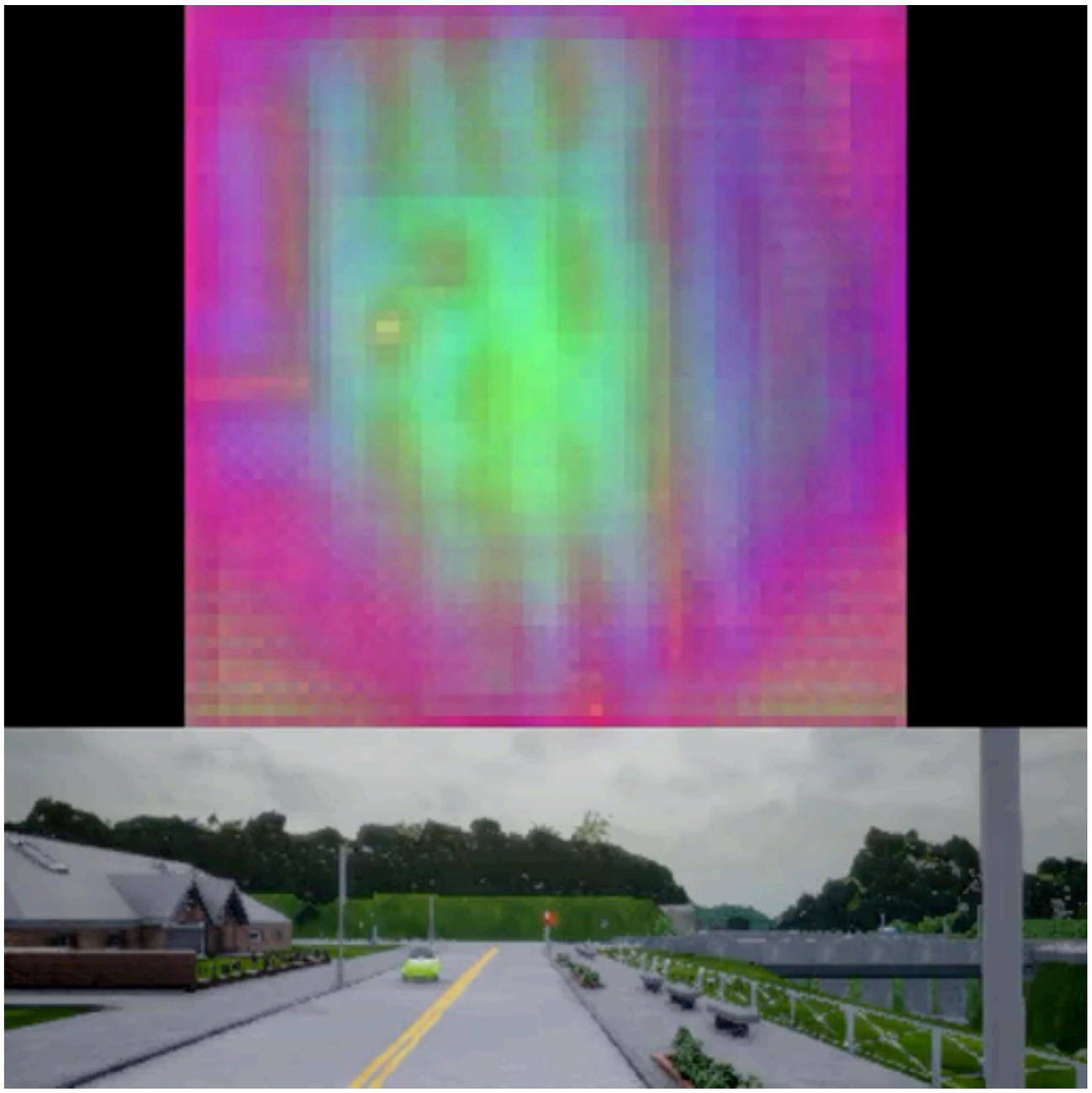

view 1
view 2


view 1
view 2
positive
corresp.


view 1
view 2
positive
corresp.
negative
corresp.


view 1
view 2
positive
corresp.
negative
corresp.
contrastive learning objective

View-contrastive feature training

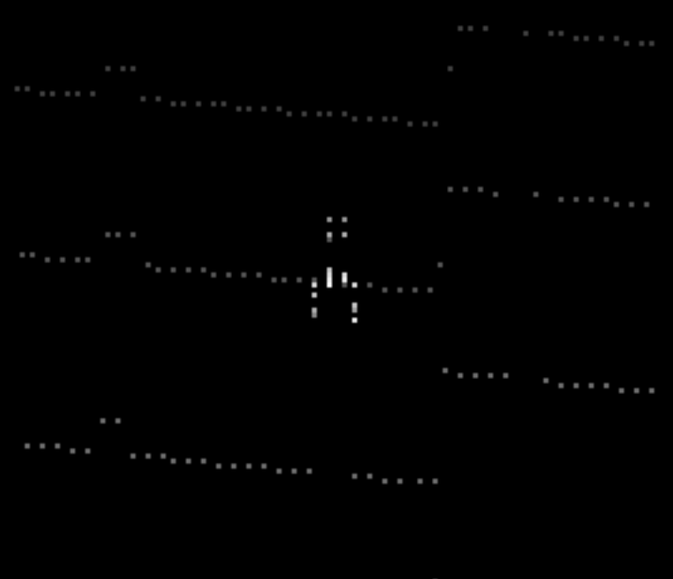


Learning from Unlabelled Videos Using Contrastive Predictive Neural 3D Mapping, Harley et al., ICLR 2020

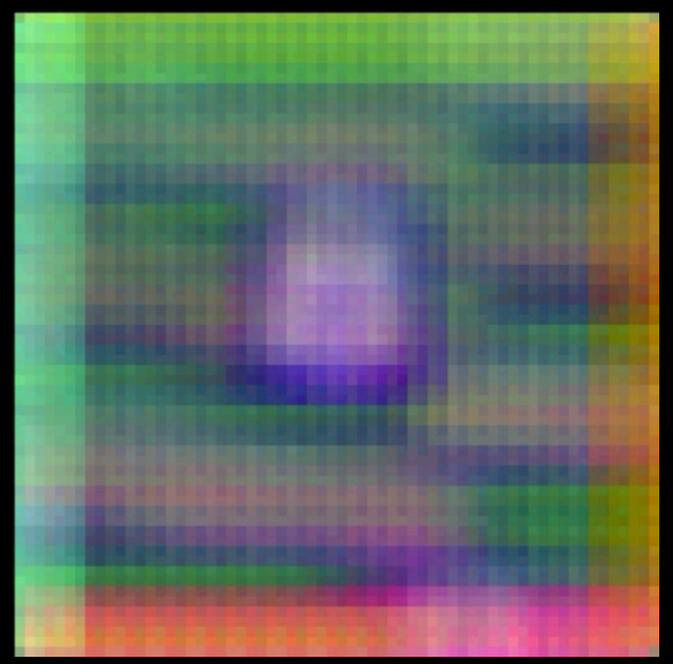
View contrastive pretraining helps 3D object tracking

Perspective view

Search region



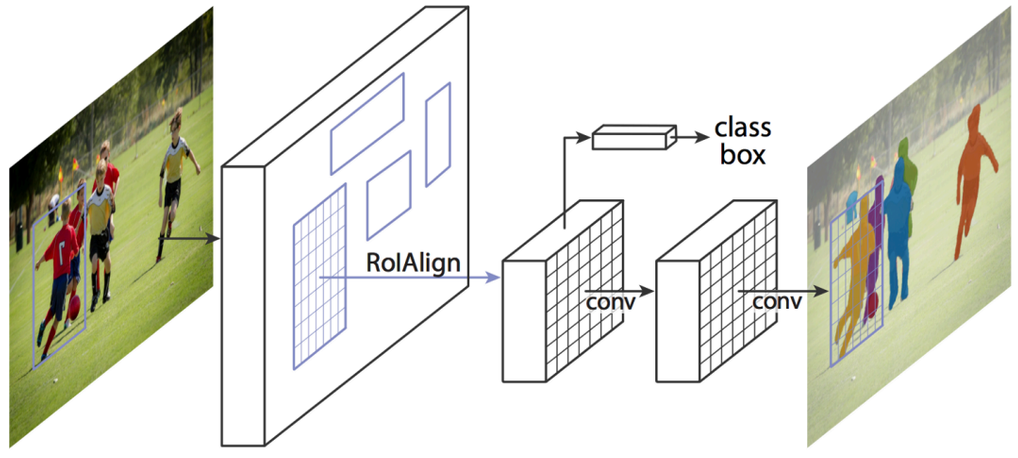
Bird’s eye view



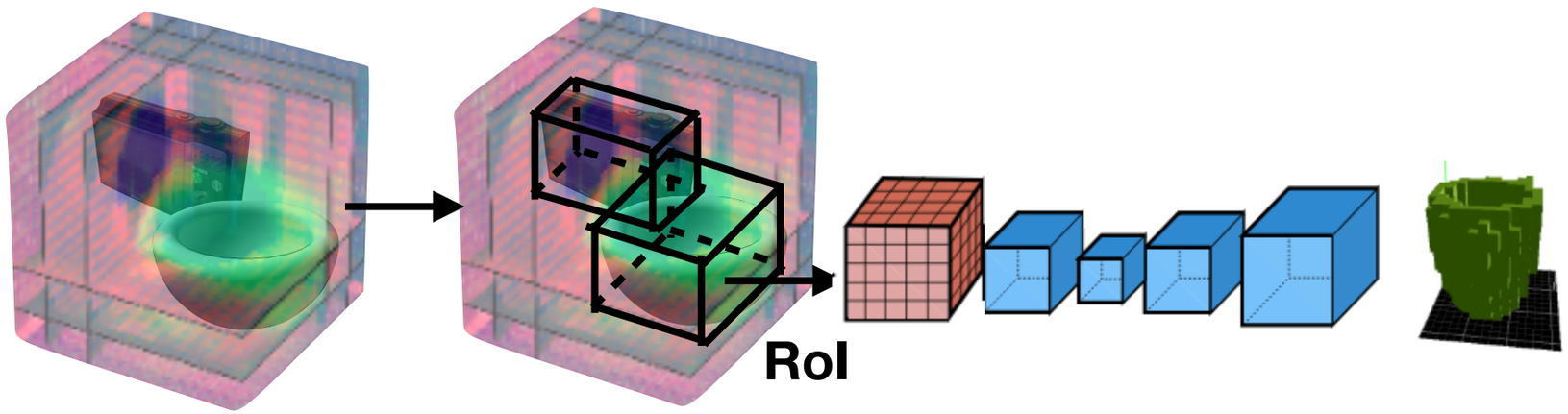
Search region features

Tracking Emerges by Looking around Static Scenes with 3D Neural Mapping, Harley et al. ECCV 2020

3D object detection with GRNNs

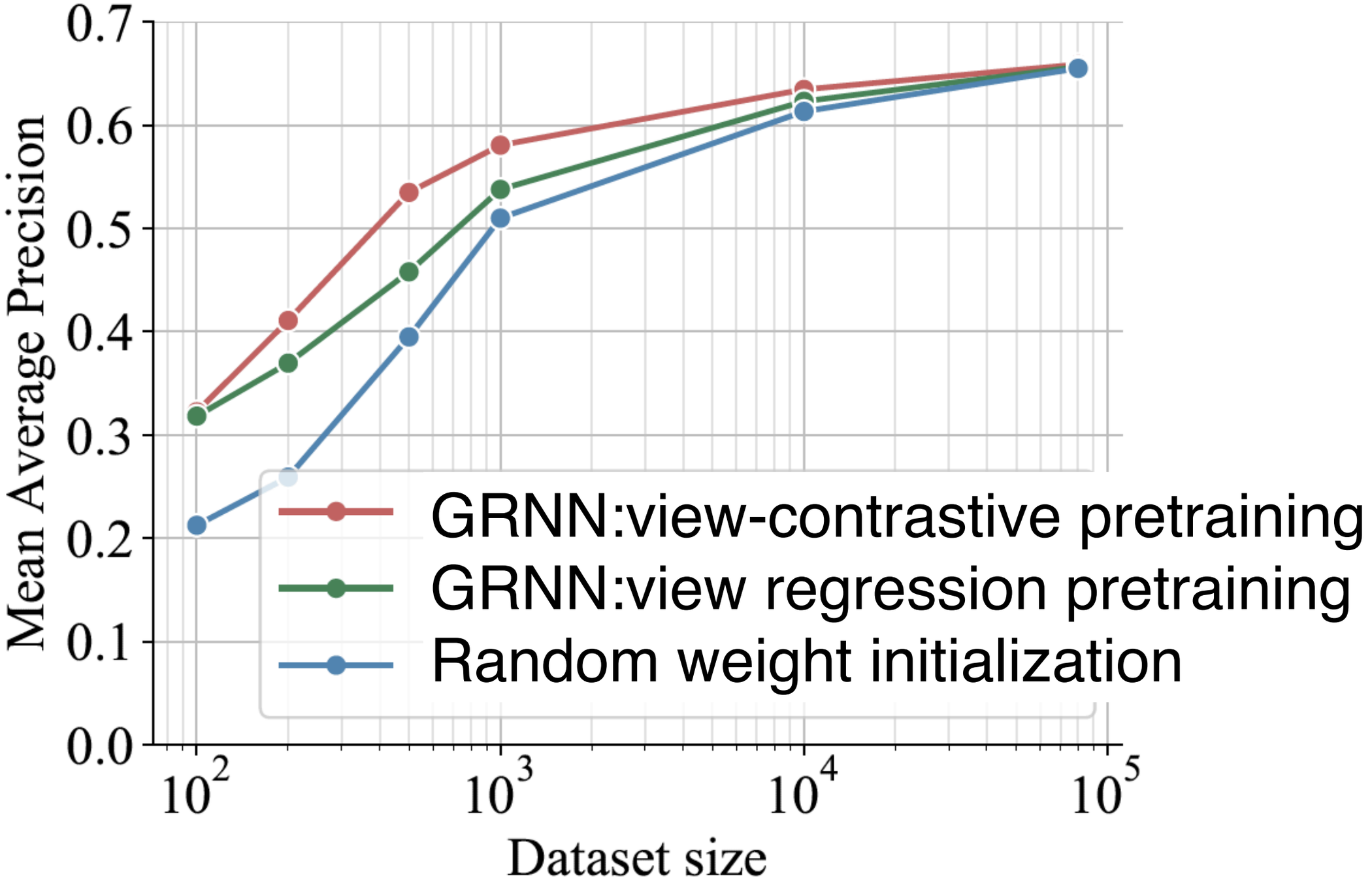


2D Mask RCNN [K. He et al]



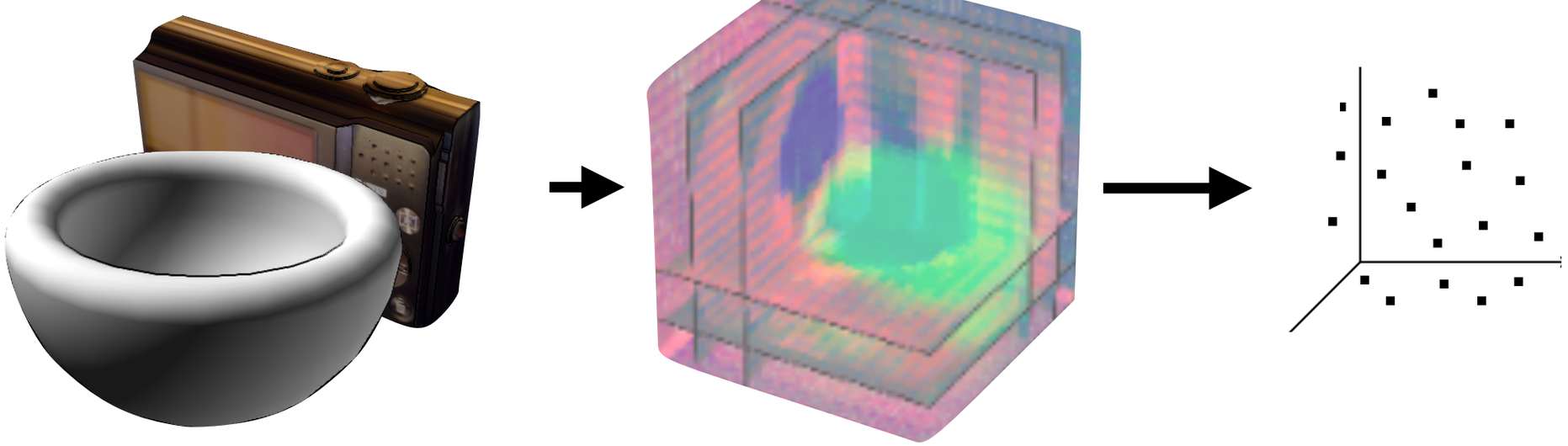
3D Mask RCNN

View contrastive pretraining helps 3D object detection



Learning from Unlabelled Videos Using Contrastive Predictive Neural 3D Mapping, Harley et al., ICLR 2020

Continuous 3D feature maps with implicit functions

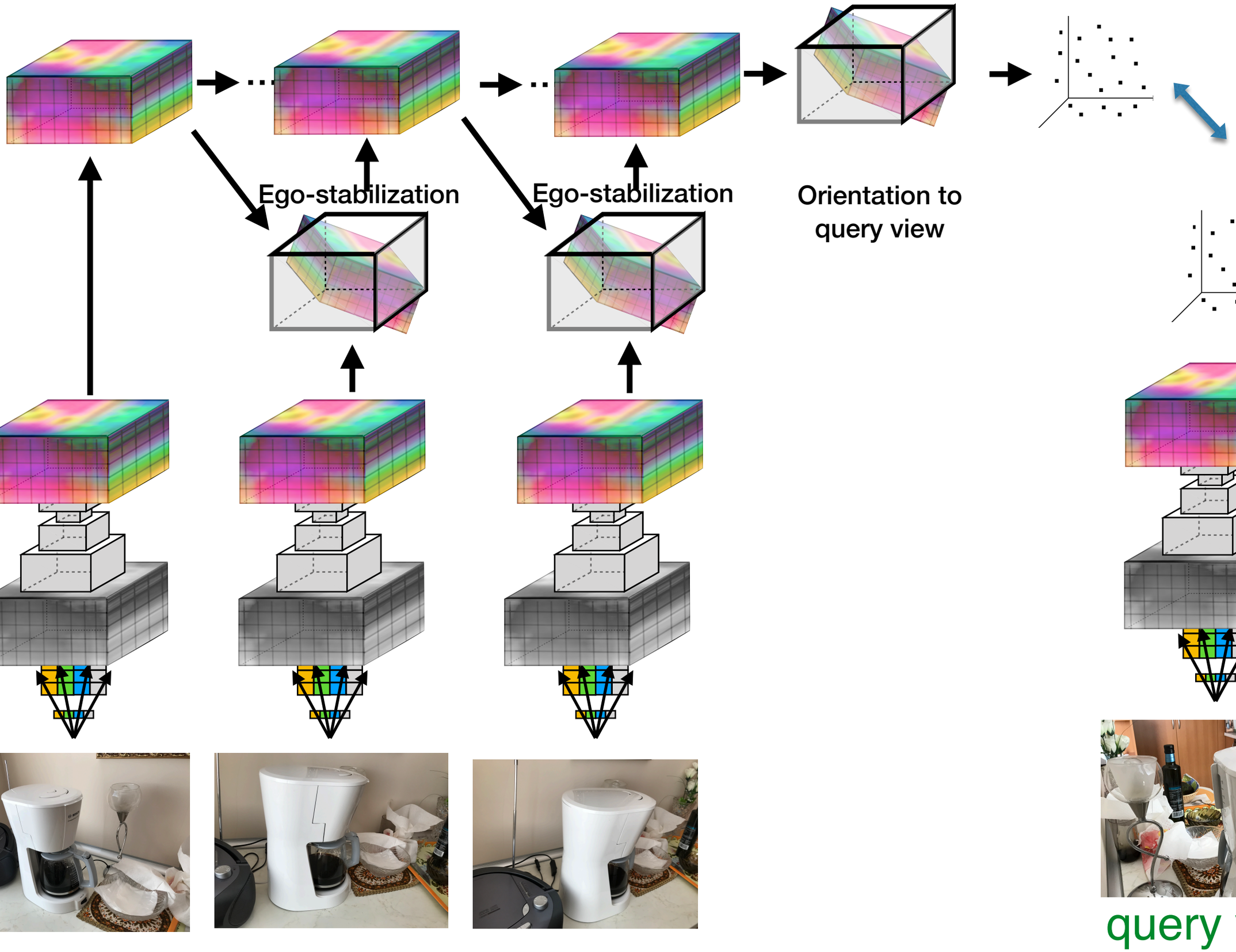


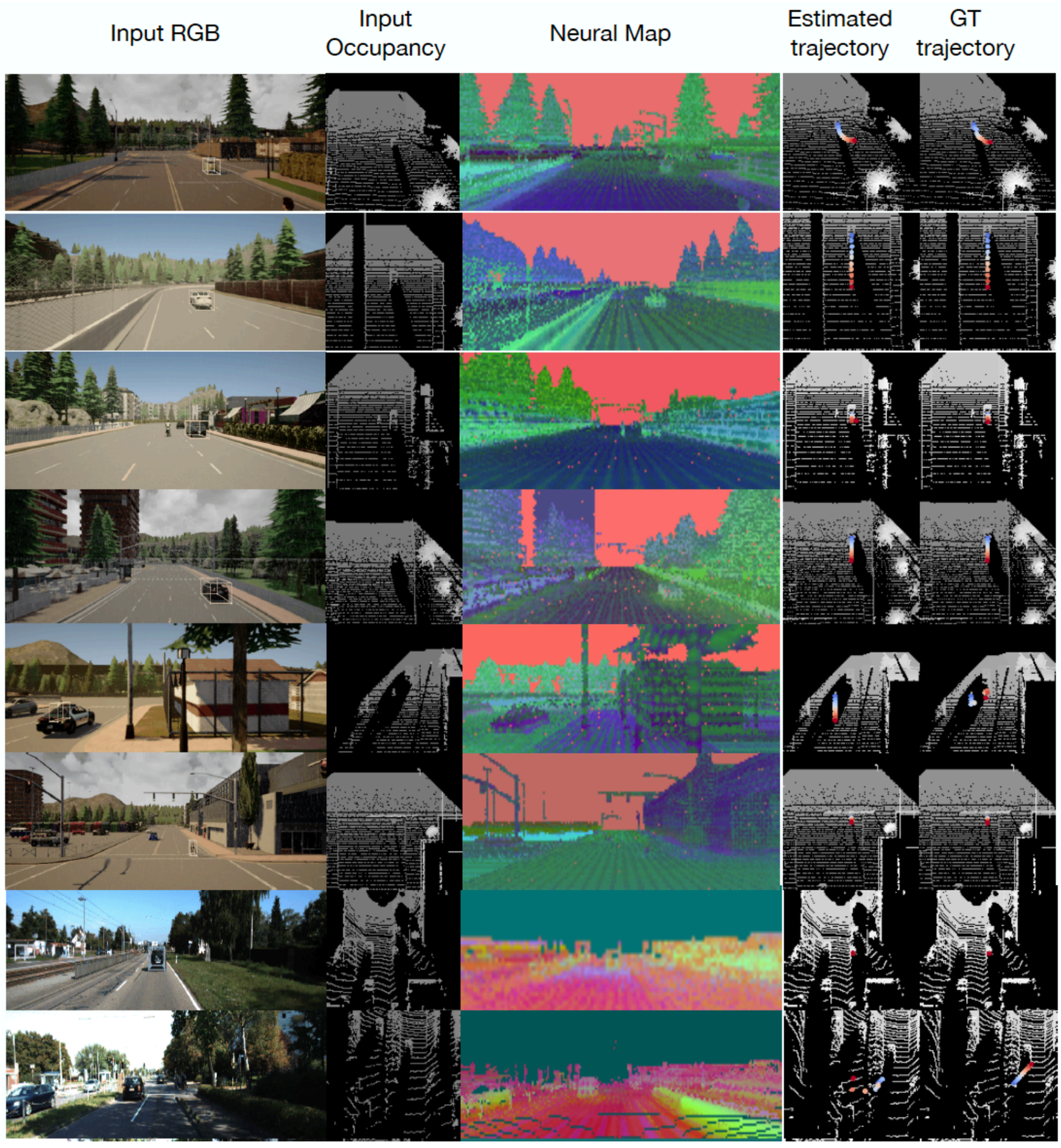
$$\mathbf{M} \in \mathbb{R}^{H \times W \times D \times C}$$

$$f^F\left(\mathbf{M}_{i,j,k}, (dX, dY, dZ); \theta_F\right) \rightarrow \mathbf{M}_{i+dX, j+dY, k+dZ} \in \mathbb{R}^C$$

$$f^O\left(\mathbf{M}_{i,j,k}, (dX, dY, dZ); \theta_O\right) \rightarrow \{0,1\}$$

View contrastive prediction




Input RGB
Input Occupancy
Neural Map
Estimated trajectory
GT trajectory

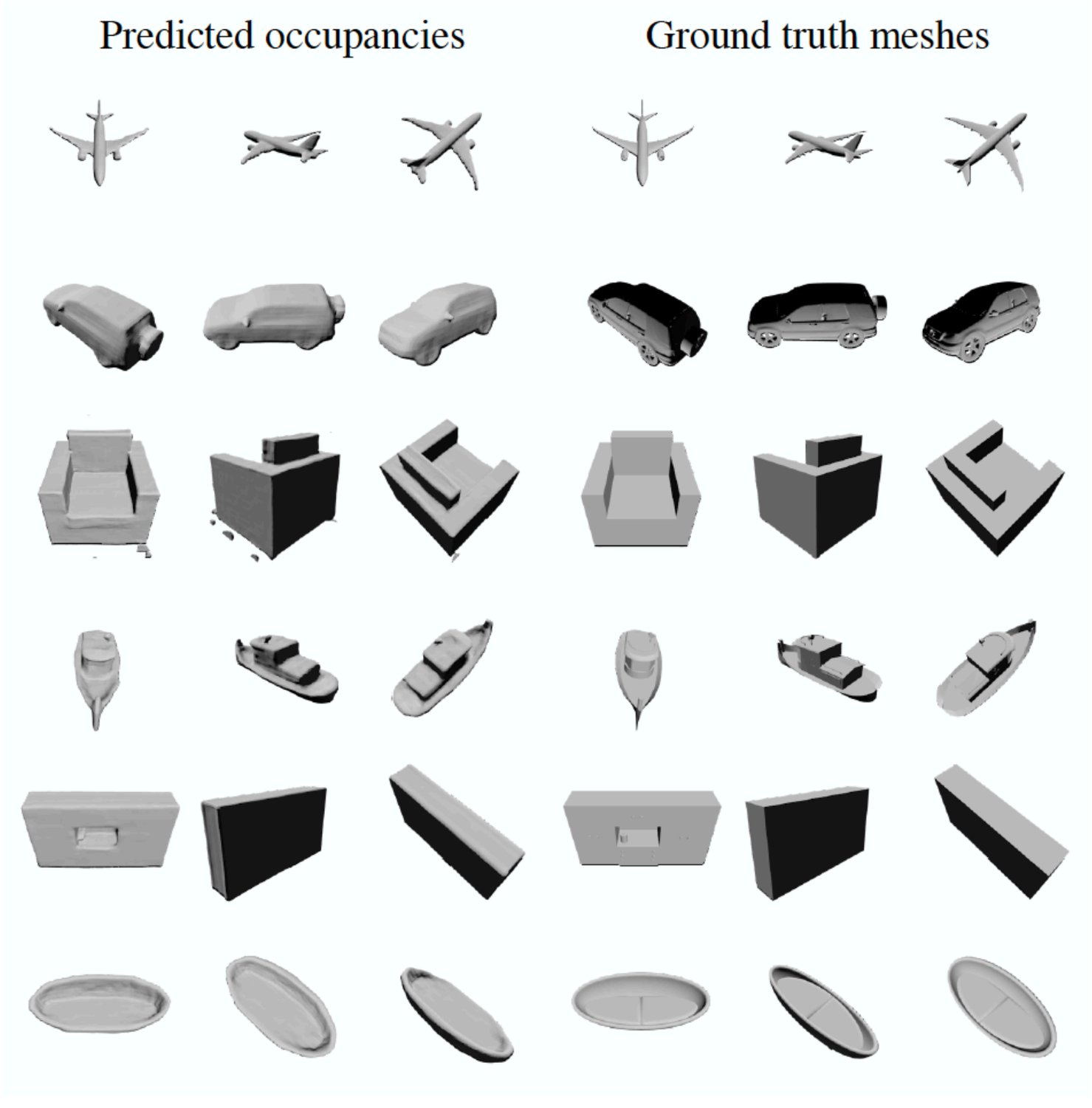
Evaluating the learnt features

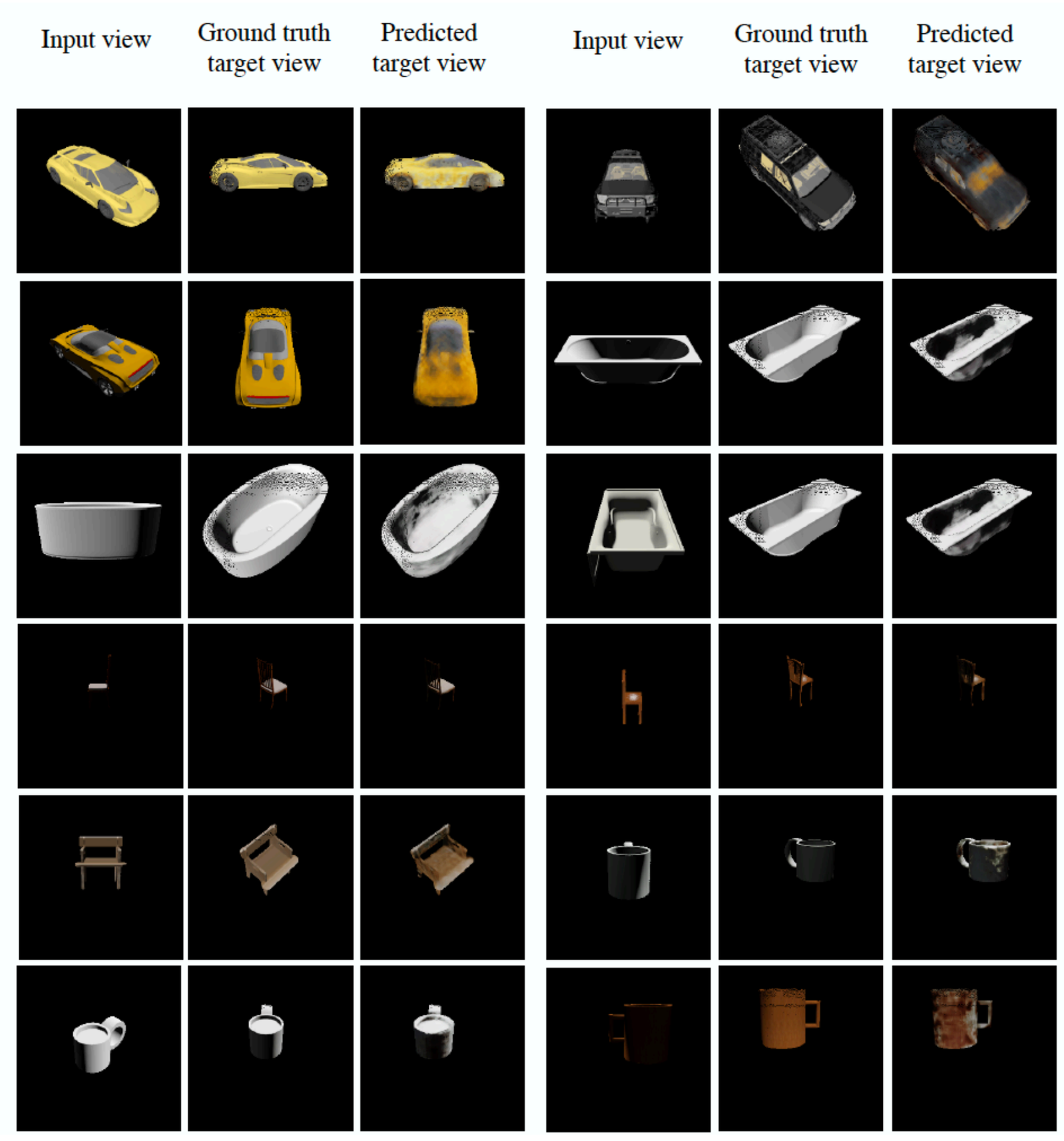
Object re-indentification/tracking

Method	CARLA	KITTI
3CNets (ours)	**0.61**	**0.54**
3DPointContrast [55]	0.55	0.48
3DVoxContrast [13]	0.37	0.23
2.5DDenseObjectNets [7]	0.24	0.19

Pretraining for a point cloud 3D object detector

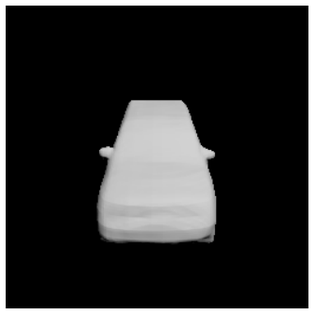
Method	0.25	0.30	0.40	0.50
VoteNet [36]	0.32	0.26	0.24	0.20
3CNets -VoteNet (Ours)	**0.51**	**0.47**	**0.41**	**0.32**

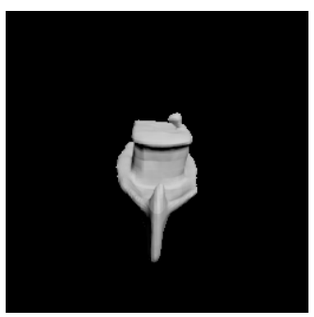

Predicted occupancies
Ground truth meshes


Input view
Ground truth target view
Predicted target view
Input view
Ground truth target view
Predicted target view

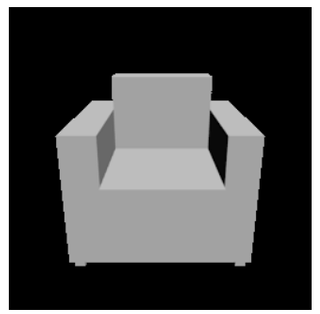
Occupancy Prediction Results

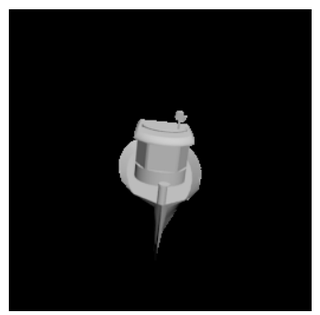
Predicted Occupancy





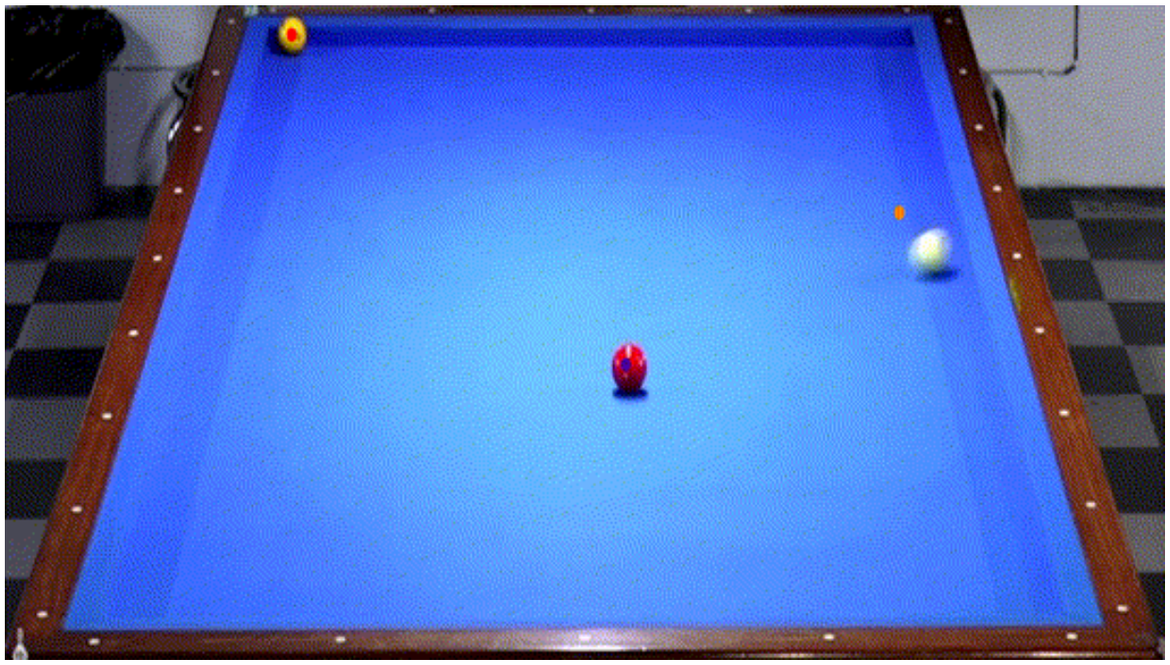
Ground Truth Mesh



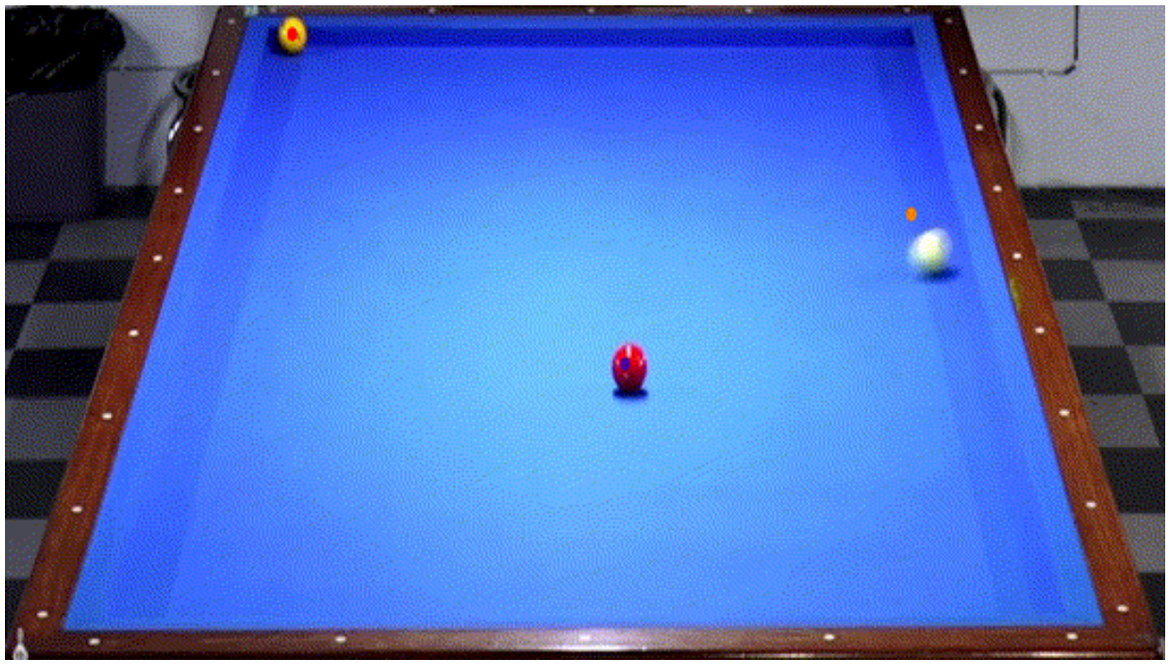


Learning object interactions

GT



Prediction



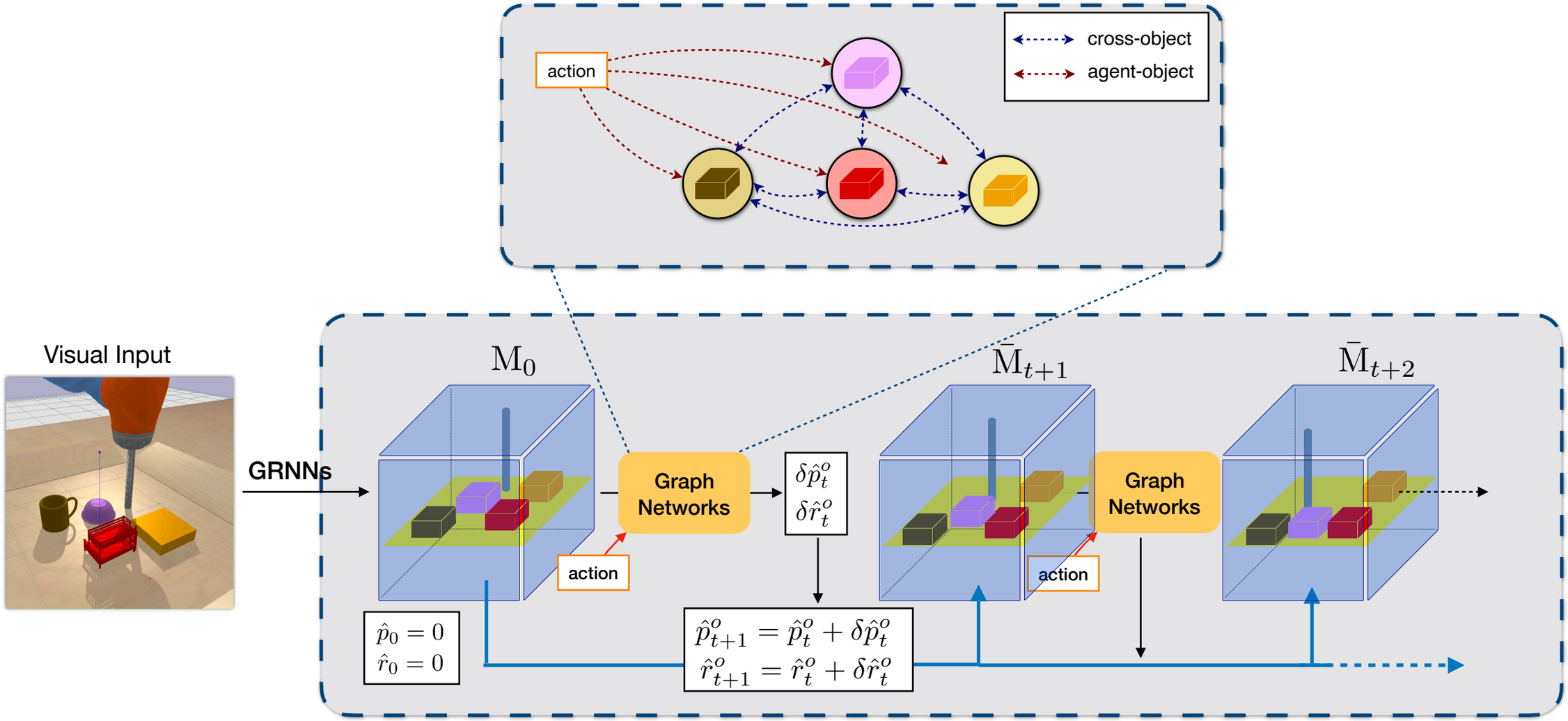
- General method: detect objects, input their appearance and motion history into a Graph Neural Network and predict their future motion trajectories
- **Problem**: predicted object motion trajectories are in 2D image space, predictions fail to generalize across viewpoints

Learning Long-term Visual Dynamics with Region Proposal Interaction Networks

Learning 3D object dynamics

- To learn how objects move, it is important to capture object permanence
- 3D geometry networks do this by construction

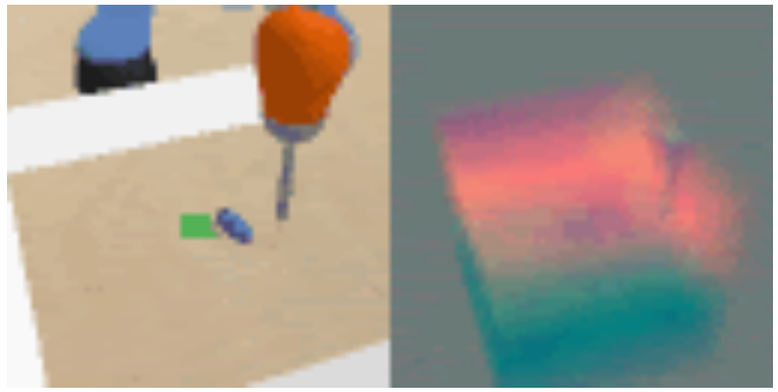
Graph Neural Networks over object 3D feature maps

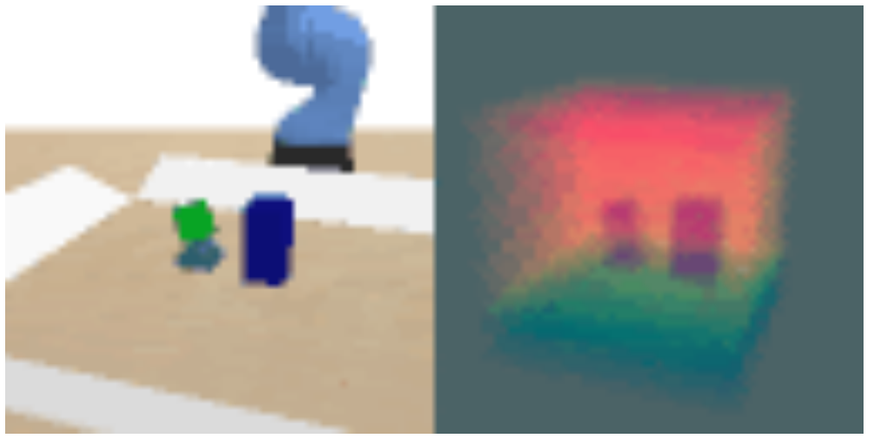


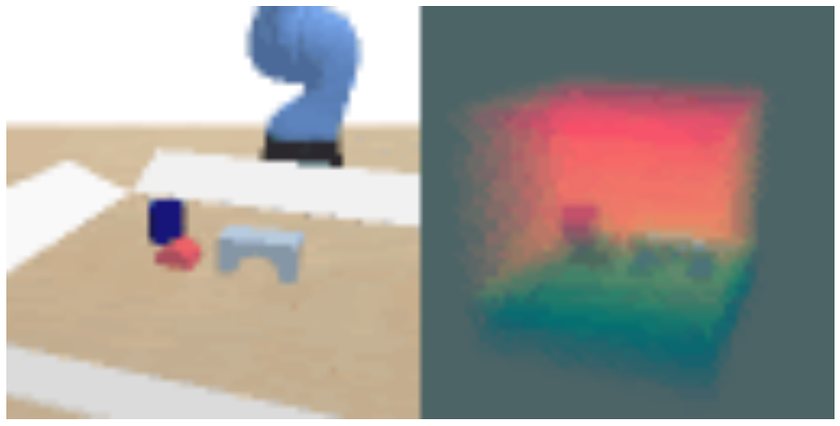
The output is the 3D translation and 3D rotation of each object

3D-OES: Viewpoint-Invariant Object-Factorized Environment Simulators, CoRL 2020

Graph Neural Networks over object 3D feature maps

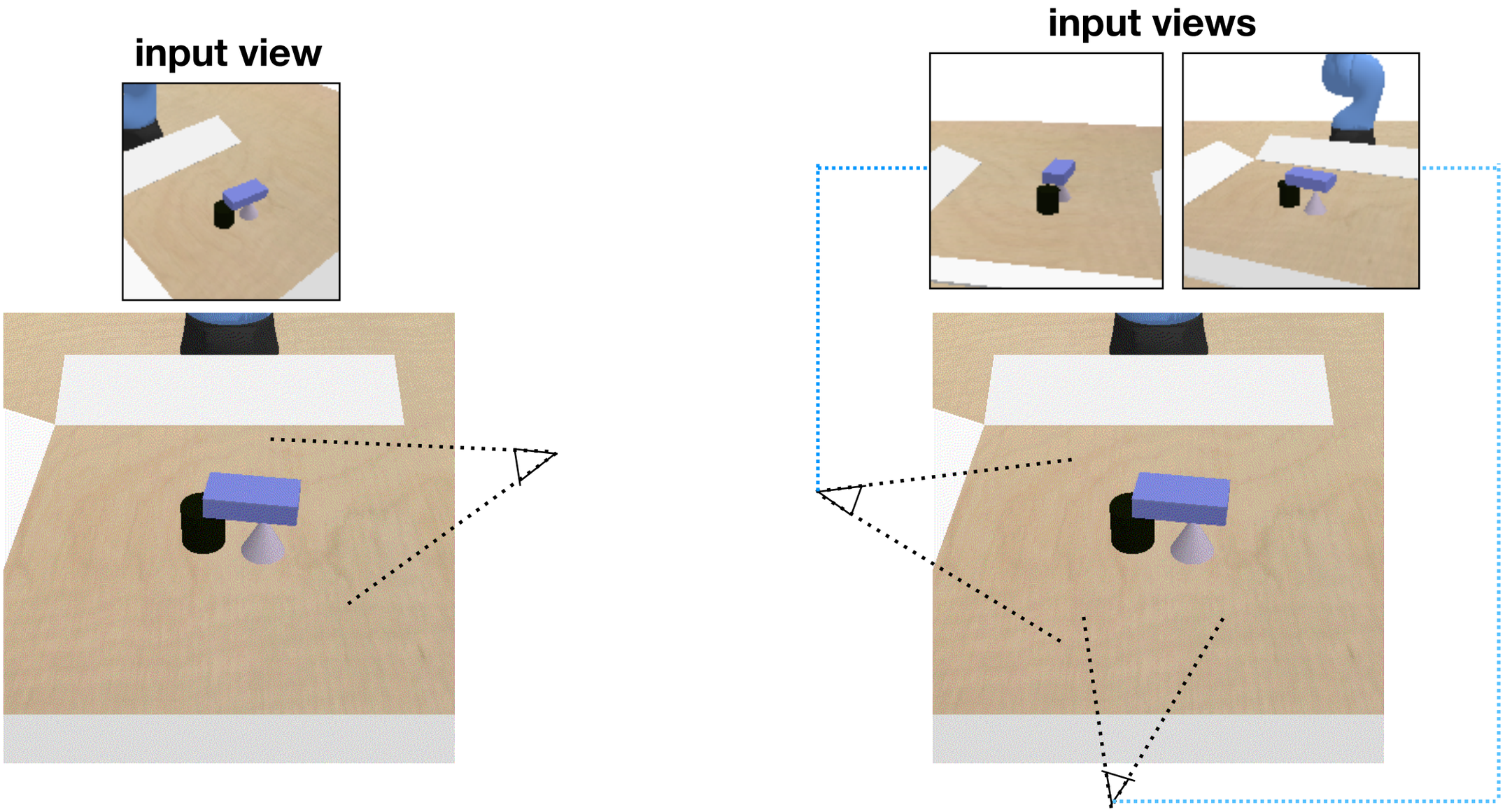






3D-OES: Viewpoint-Invariant Object-Factorized Environment Simulators, CoRL 2020

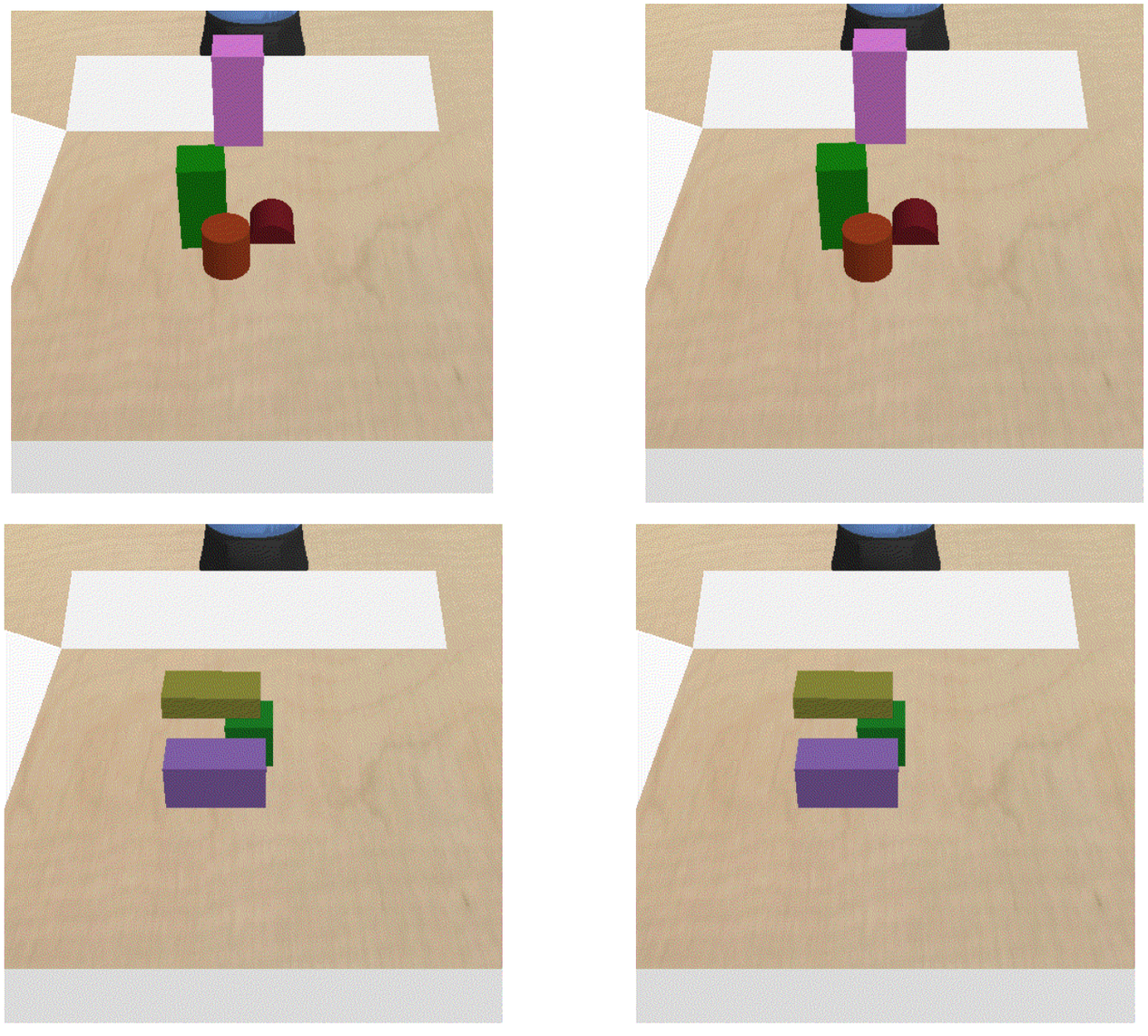
Varying viewpoint



3D-OES: Viewpoint-Invariant Object-Factorized Environment Simulators, CoRL 2020

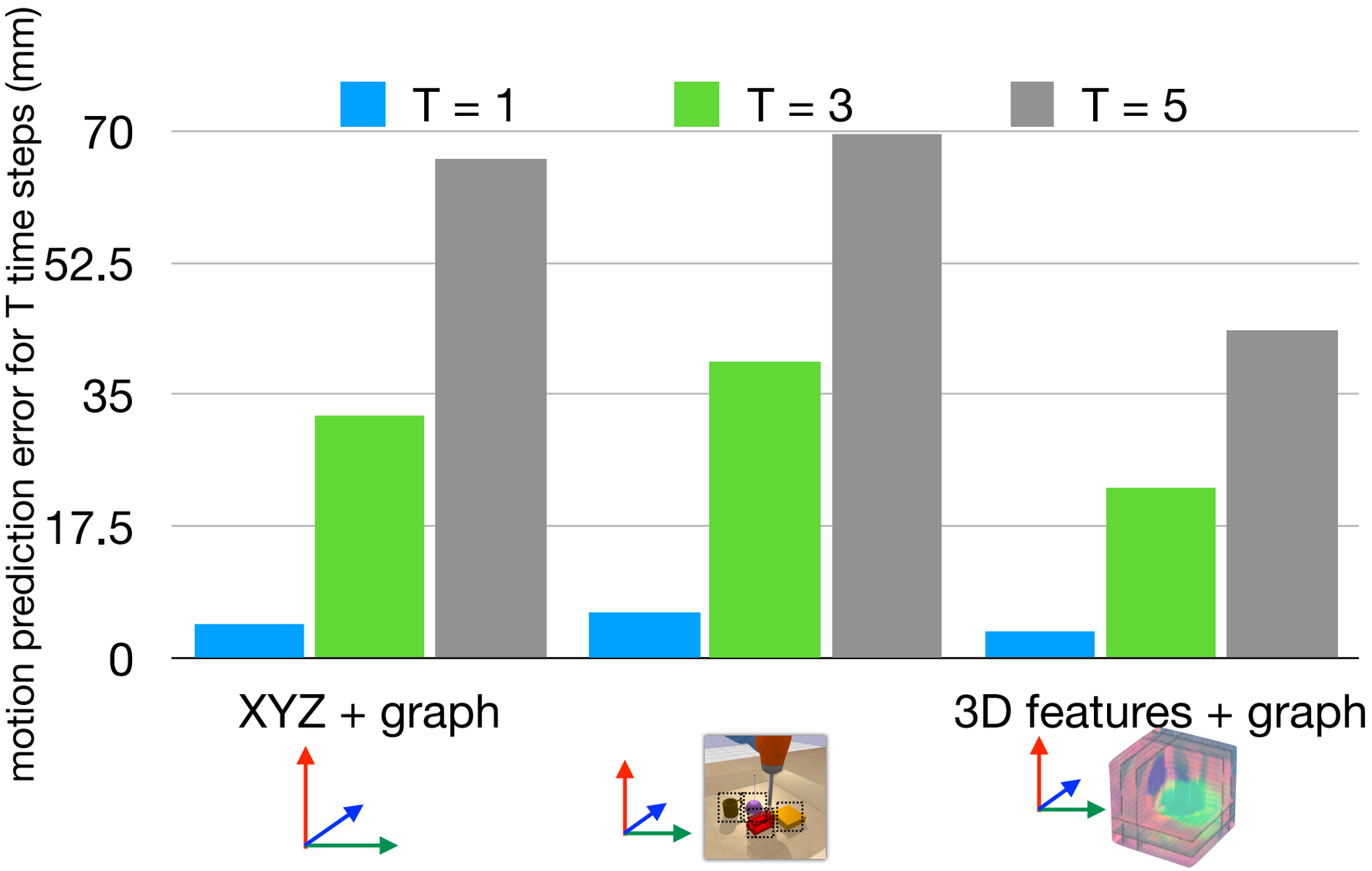
3D Object Factorized Environment Simulators

GT **Pred**

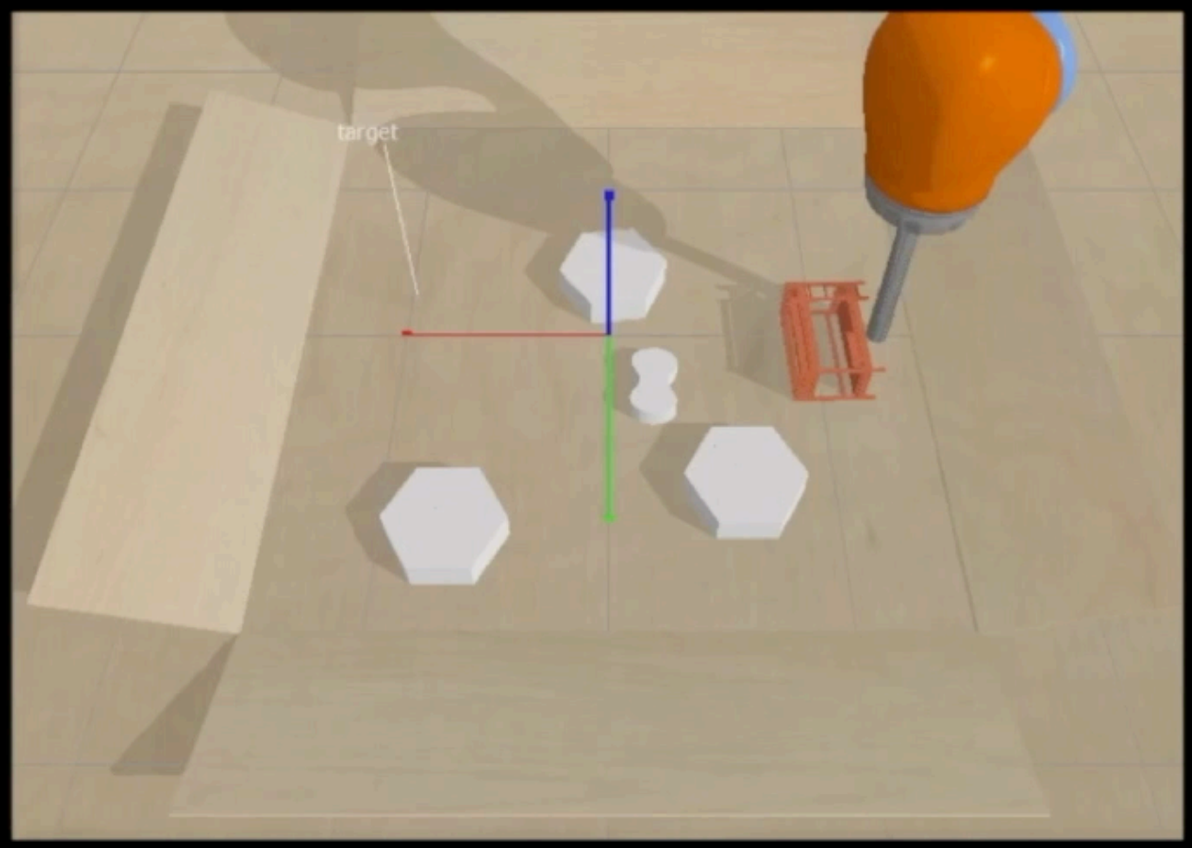


3D-OES: Viewpoint-Invariant Object-Factorized Environment Simulators, CoRL 2020

Comparisons



Obstacle Avoidance - Simulation Task 2



3D-OES: Viewpoint-Invariant Object-Factorized Environment Simulators, CoRL 2020

This talk

Learning amodal 3D feature representations

Applications in:

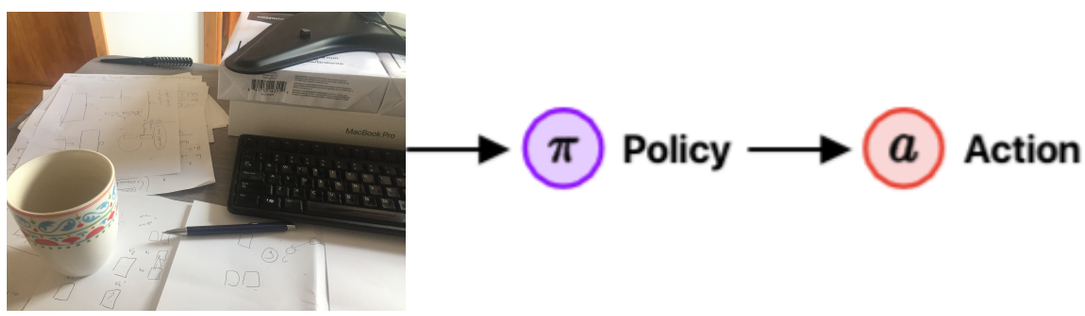
- visual tracking, 3D object detection, occupancy prediction
- intuitive physics, dynamics learning
- control
- language grounding

Vision2Action Mapping

Low dimensional object/
keypoint locations

★Data efficient
★Does not generalize across object/
scene appearances

images



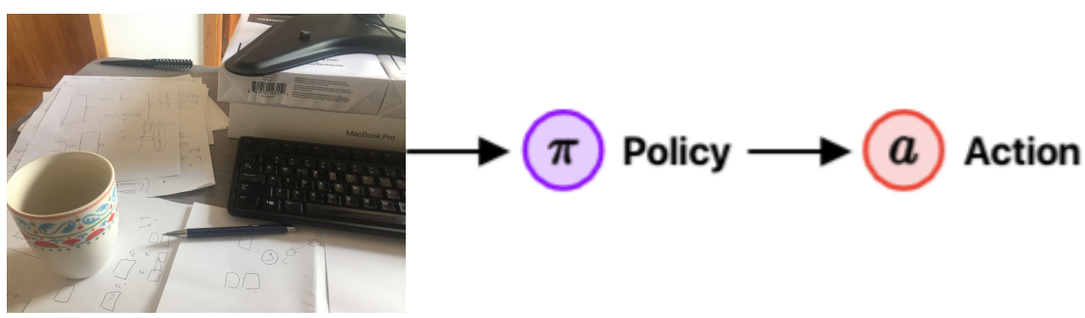
★ Data inefficient
★ Generalizes across object/scene
appearances

Vision2Action Mapping

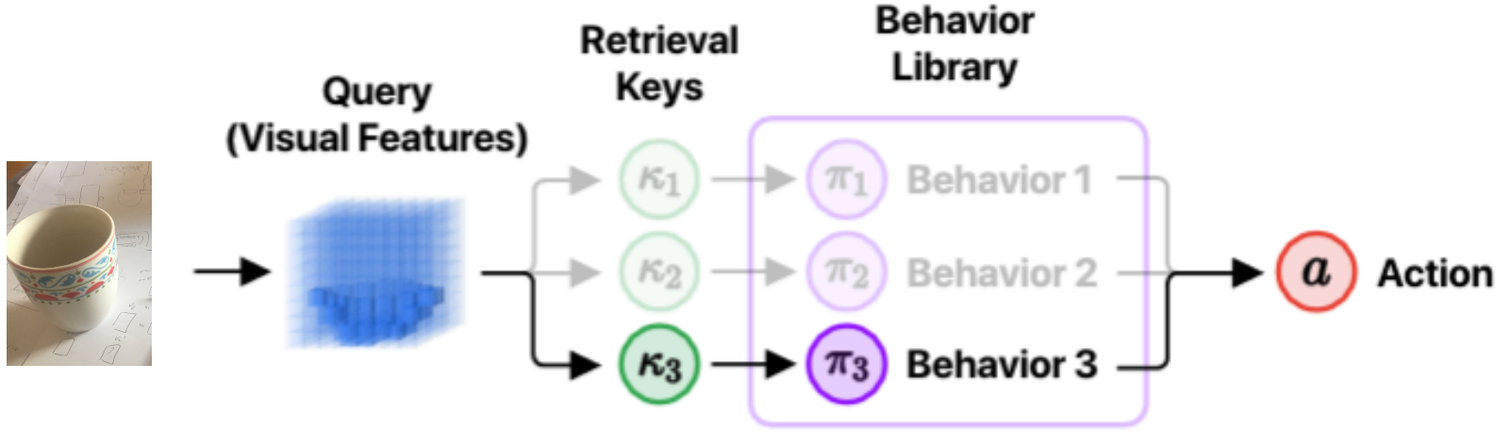
Low dimensional object/
keypoint locations

★Data efficient
★Does not generalize across object/
scene appearances

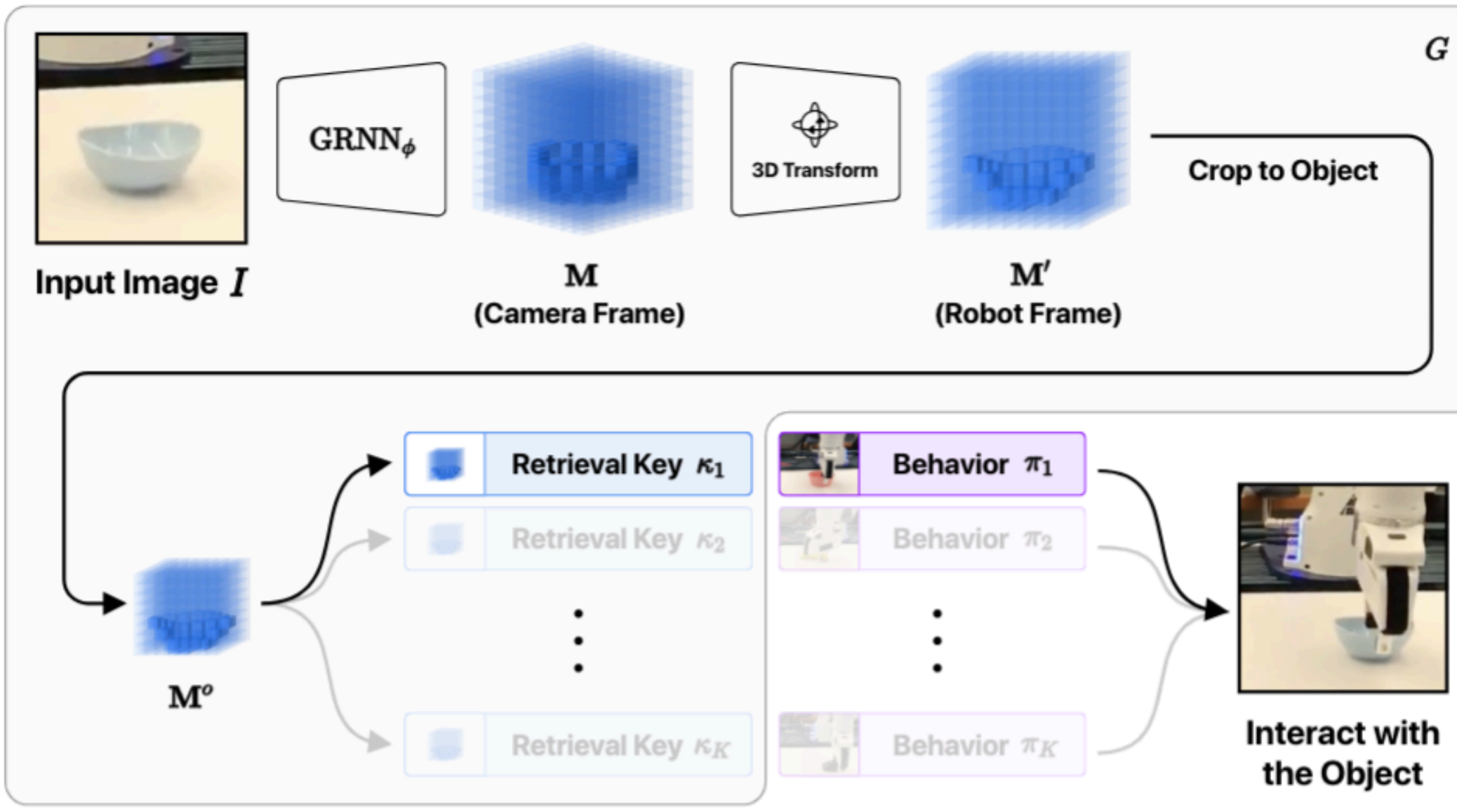
images



★ Data inefficient
★ Generalizes across object/scene
appearances

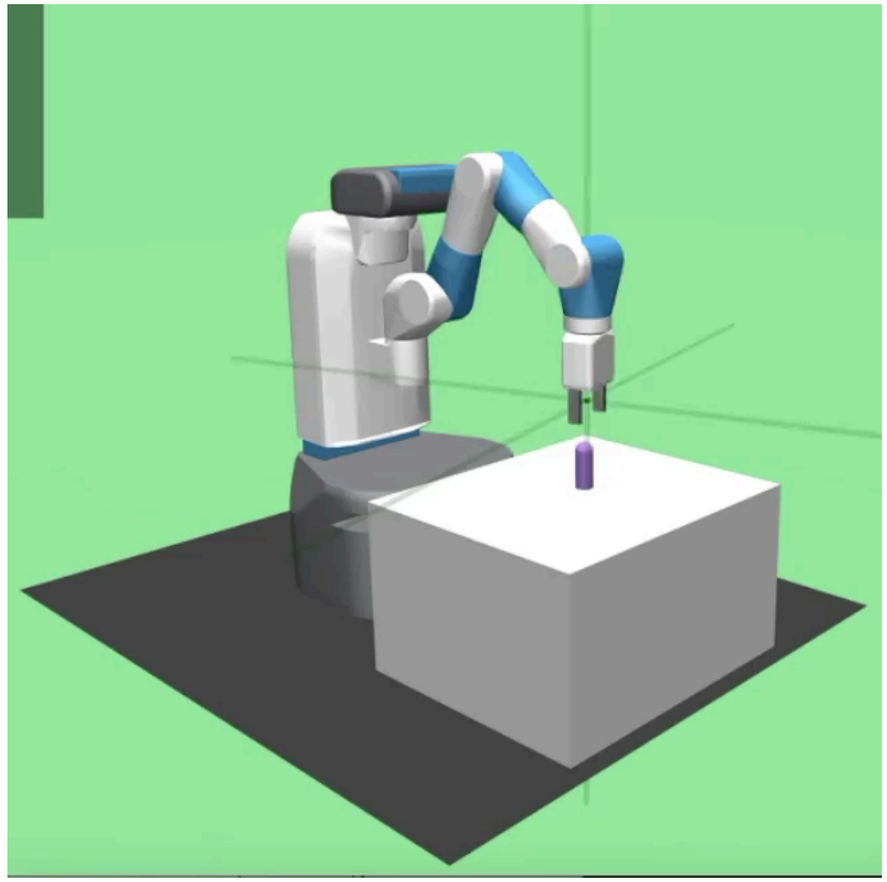


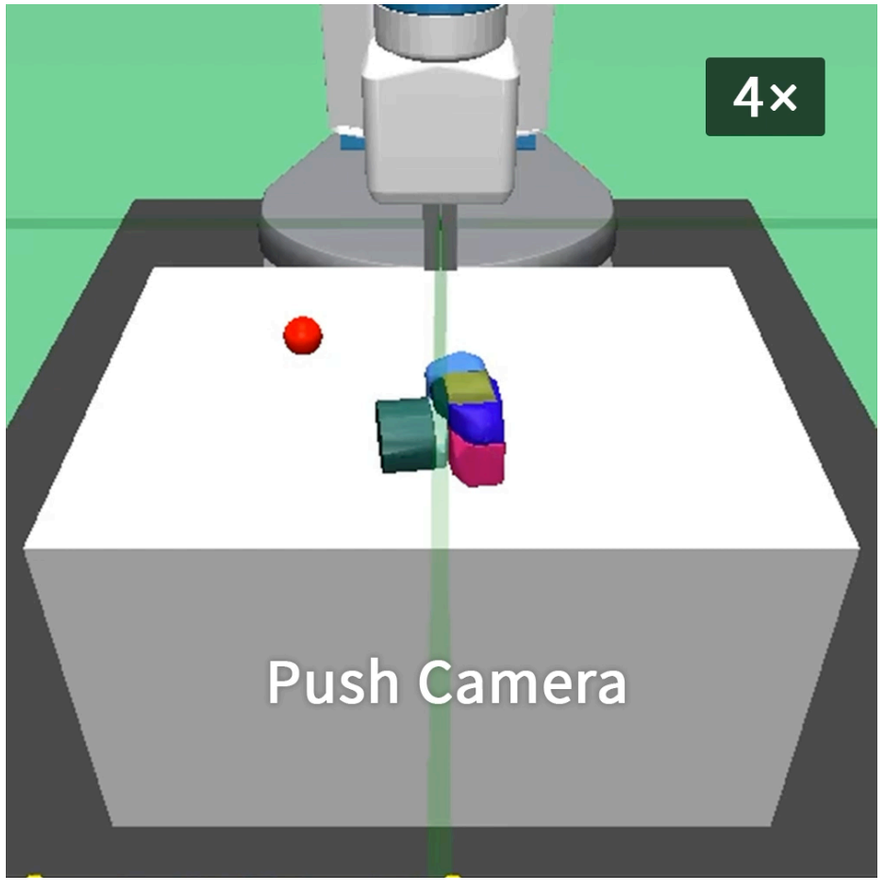
Visually-Grounded Library of Behaviours



Behaviour keys and scene features are trained by trial-and-error: bring close the scene-behaviors that were successful.

Experiments (grasping, pushing)

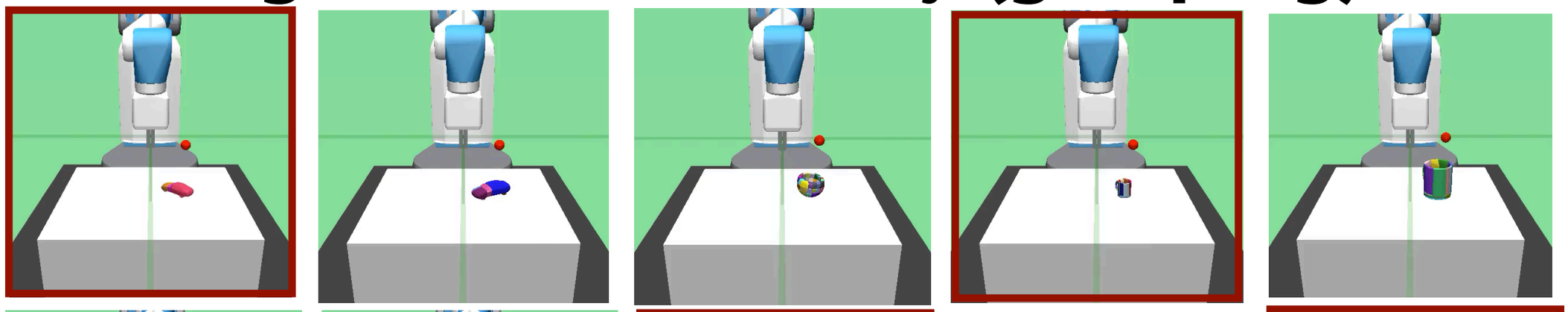




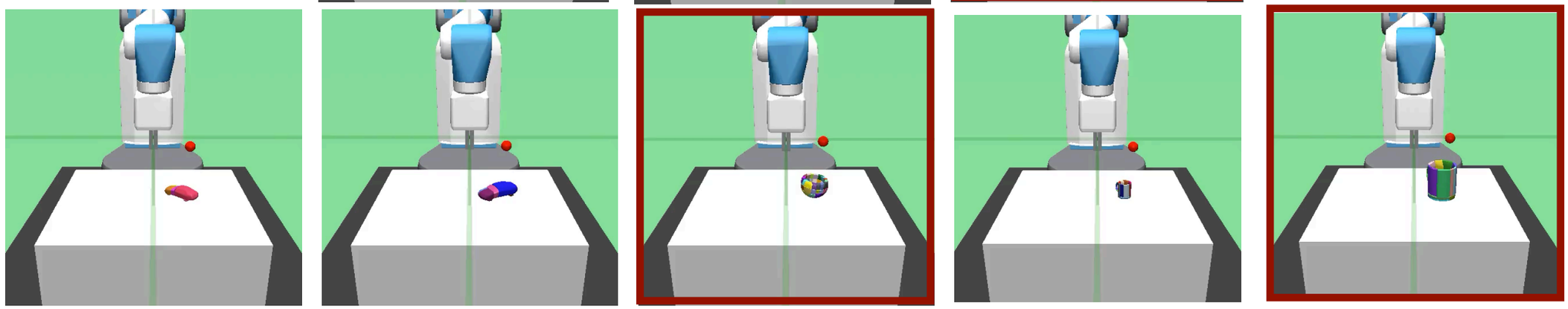
Tung*, Yang, Zhang*, Pathak, Pokle, Atkeson, Fragkiadaki

Building Behavior Library (grasping)

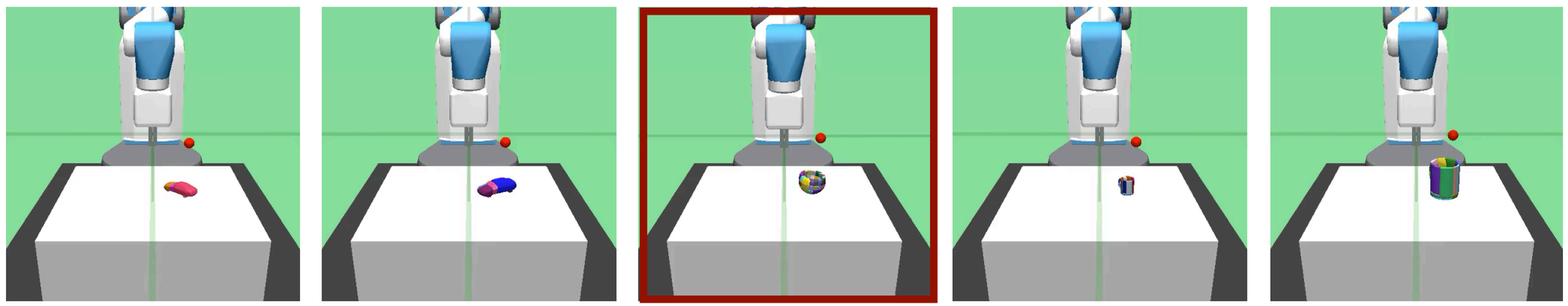
Behavior 1



Behavior 2

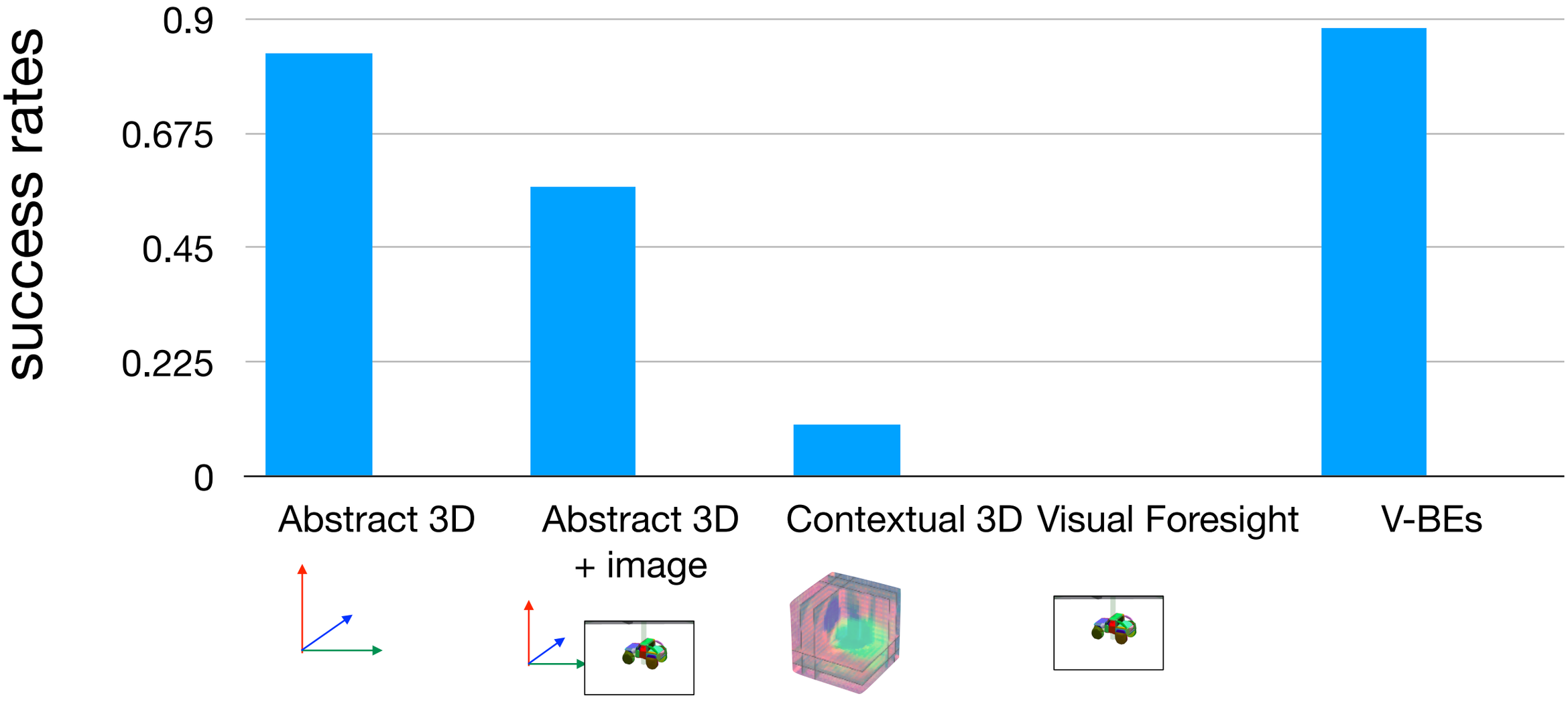


Behavior 3

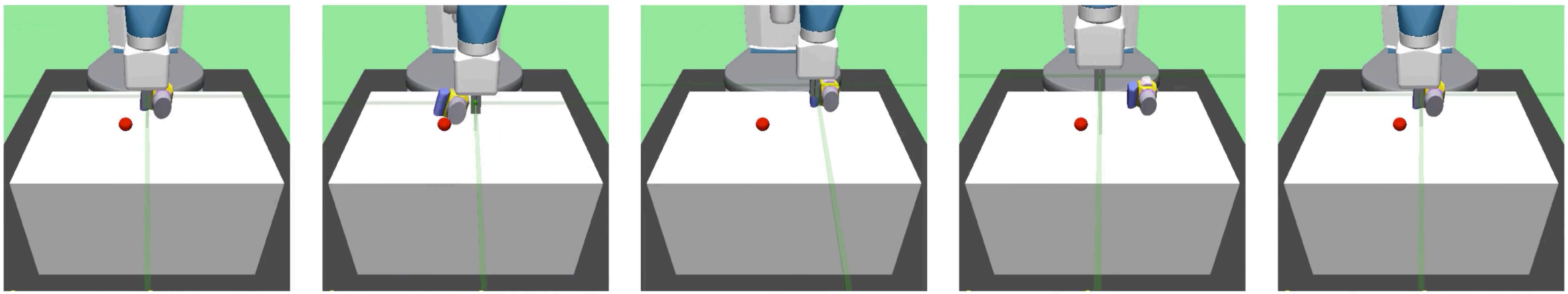


Tung*, Yang, Zhang*, Pathak, Pokle, Atkeson, Fragkiadaki

Results (pushing)



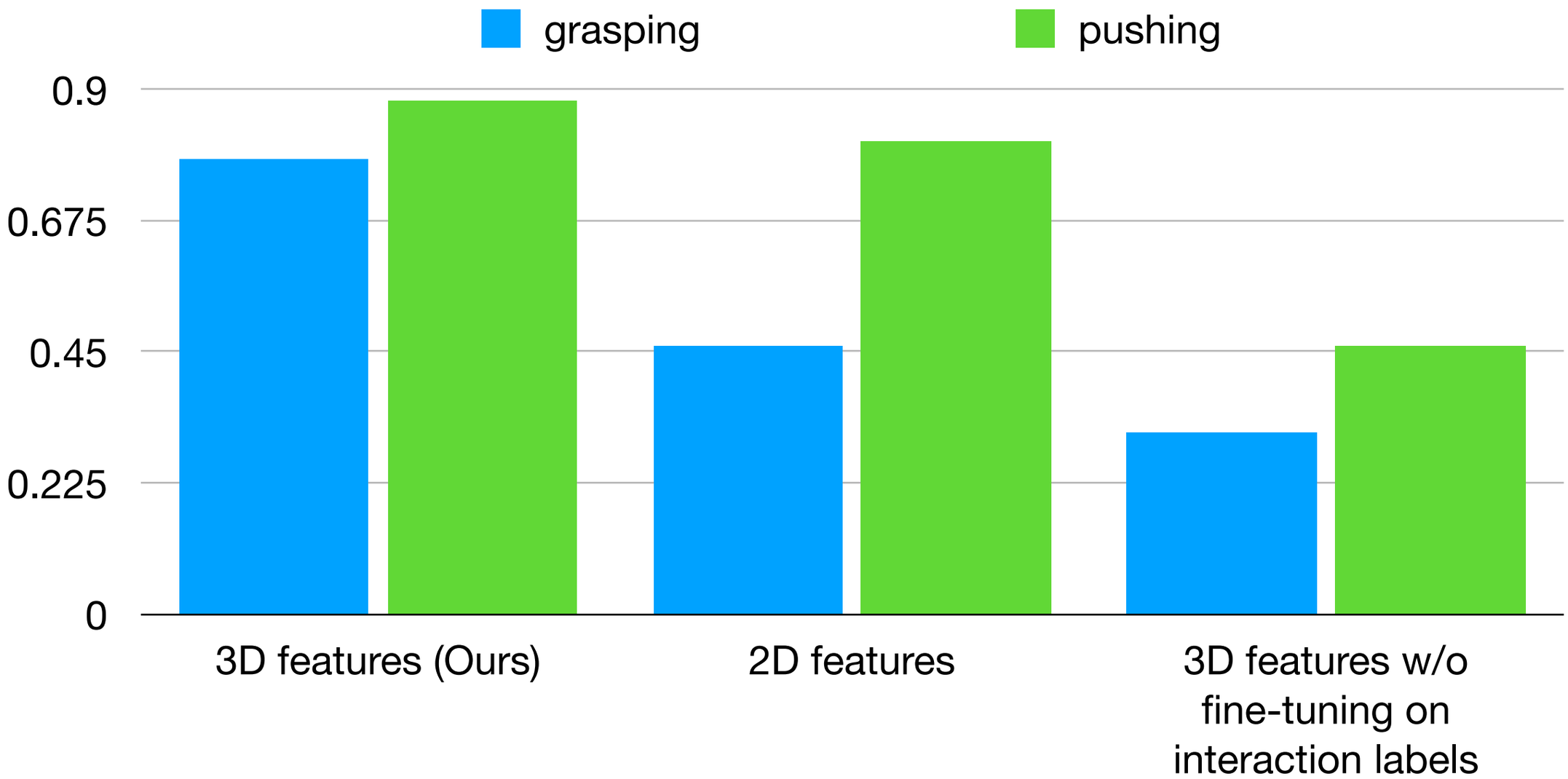
Results (pushing)



	Abstract 3D	Abstract 3D + Image	Contextual 3D	Visual Foresight	V-BEs (Ours)
success rates	0.83	0.57	0.10	0.00	0.88

Note: a blue border is shown on episode success.

Ablation Results — learning retrieval key with different visual representations



Reward learning using natural language

Goal: place *the can to the right of the bowl*

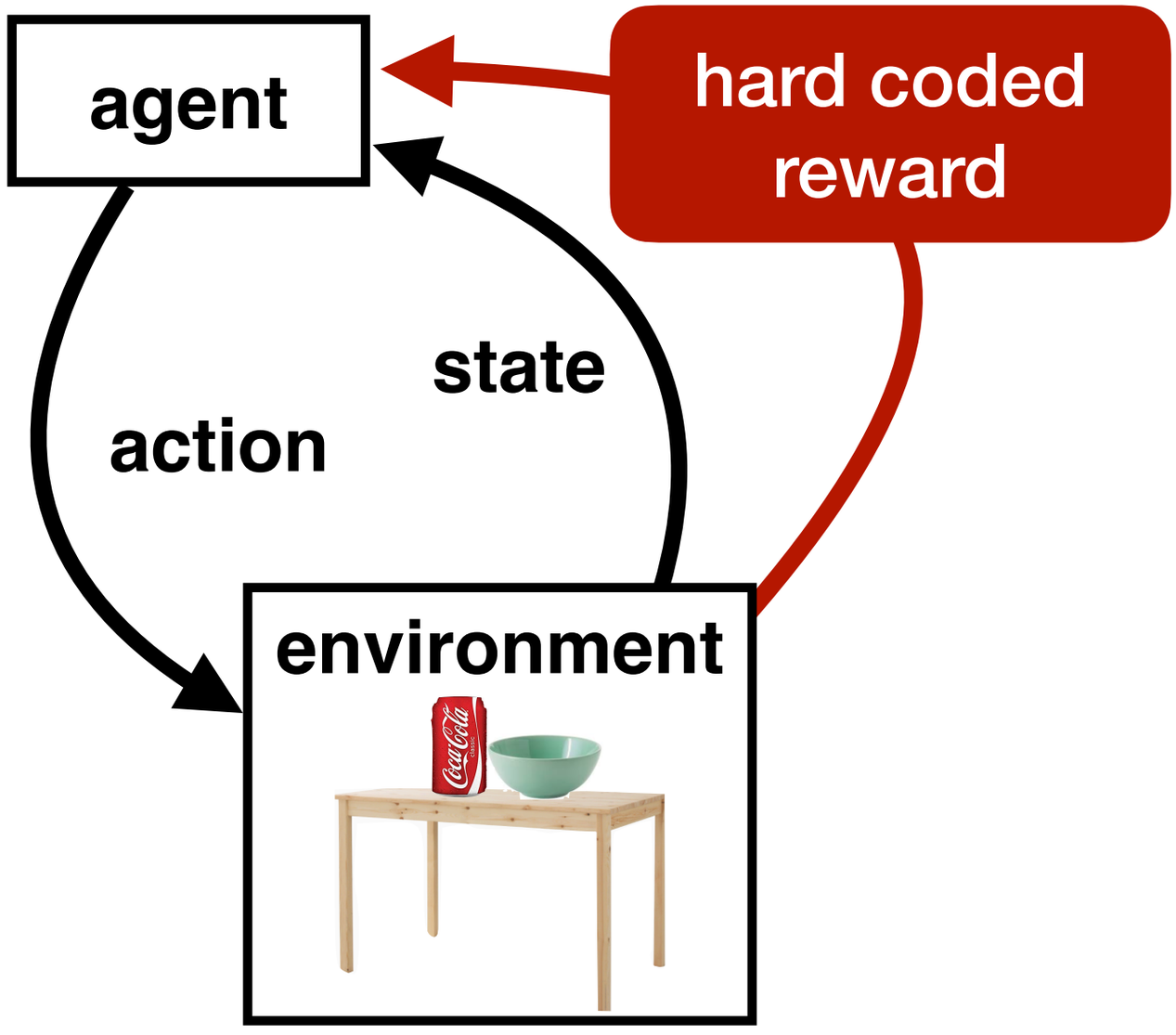
“Can is to the right of the bowl”

Use the learned visual detector to get rewards for policy learning

Tung et al. CVPR 2018

Reward learning using natural language

Goal: place the coca-cola to the left of the bowl

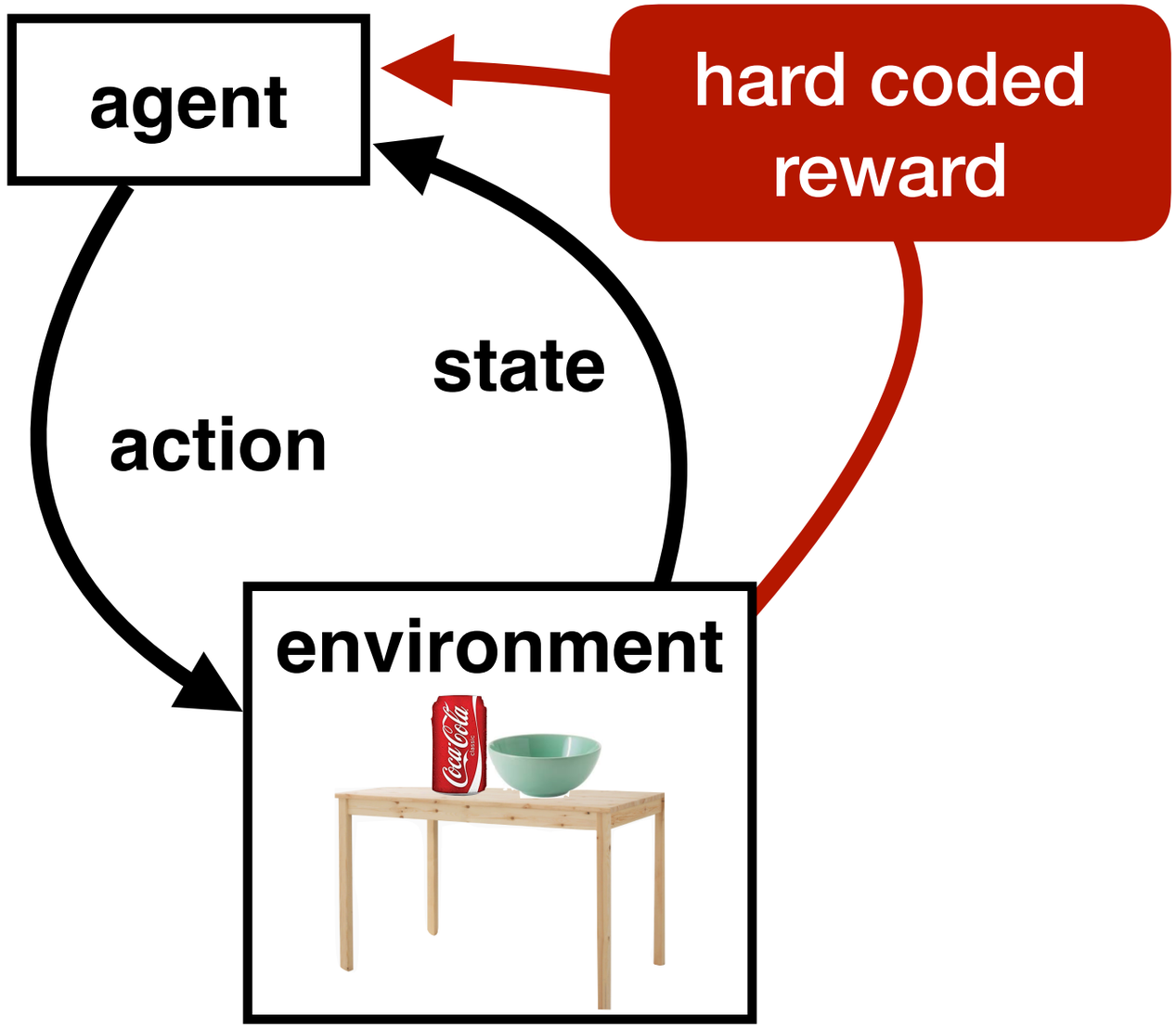


Manually code the reward in a simulated or instrumented environment

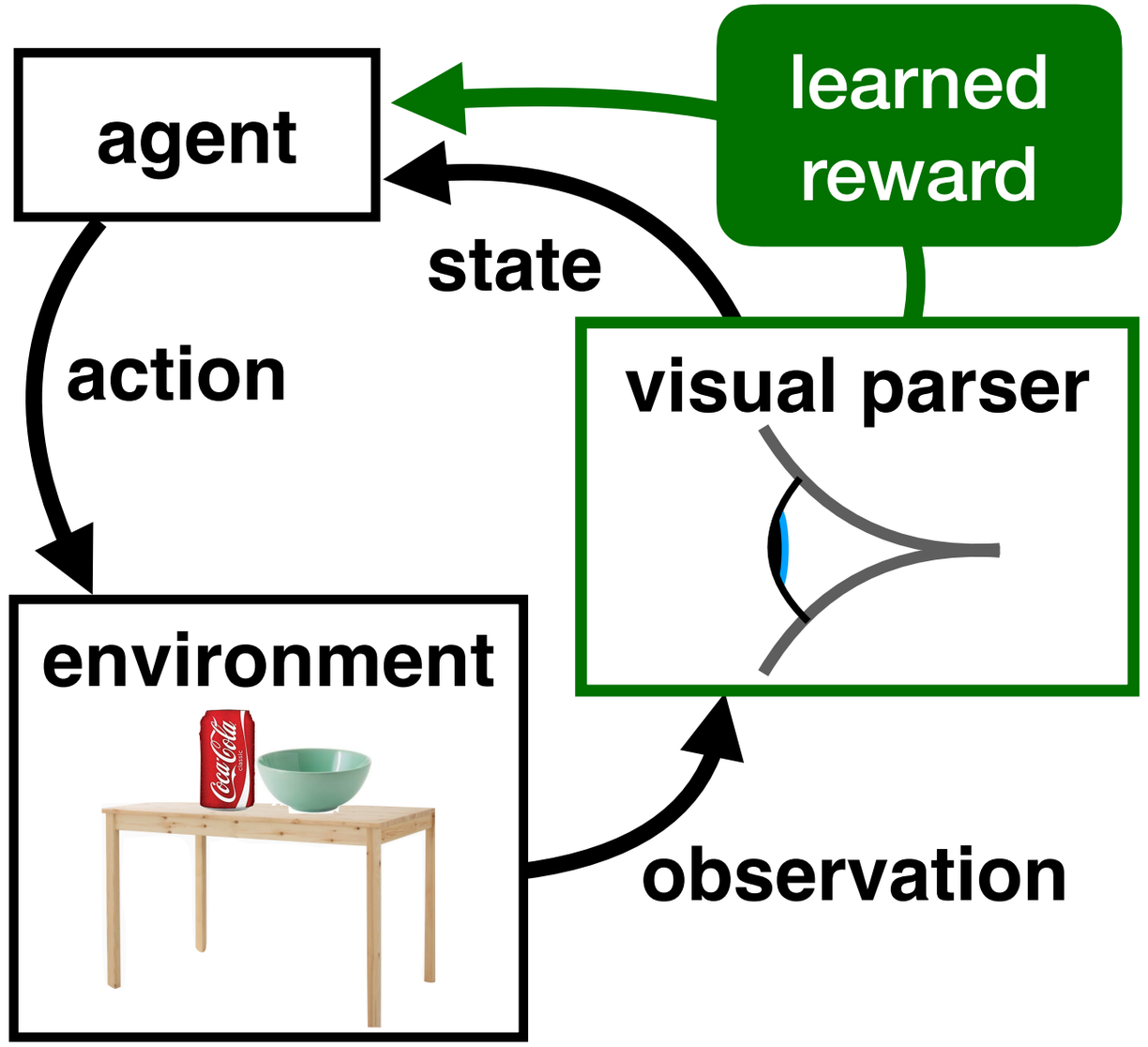
Tung et al. CVPR 2018

Reward learning using natural language

Goal: place the coca-cola to the left of the bowl



Manually code the reward in a simulated or instrumented environment



Learn to detect from an RGB image when the goal is achieved

Tung et al. CVPR 2018

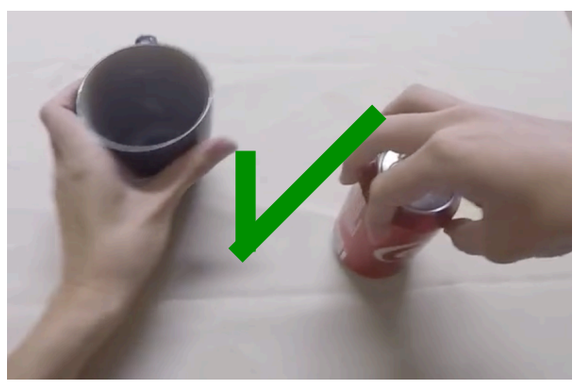
Reward learning using natural language

“*Can is to the right of the mug*”

Modeling Relationships in Referential Expressions with Compositional Modular Networks, Ronghang et al.

Reward learning using natural language

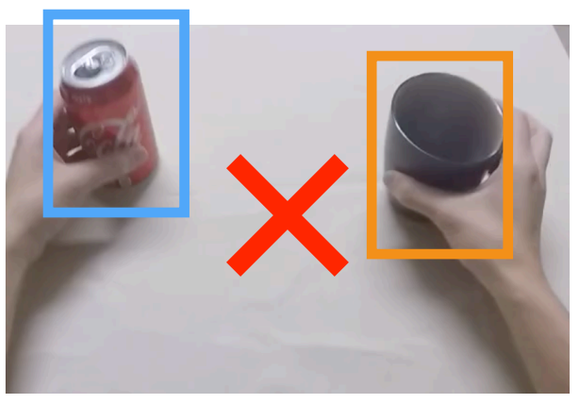
"Can *is to the right of the mug*"



Modeling Relationships in Referential Expressions with Compositional Modular Networks, Ronghang et al.

Reward learning using natural language

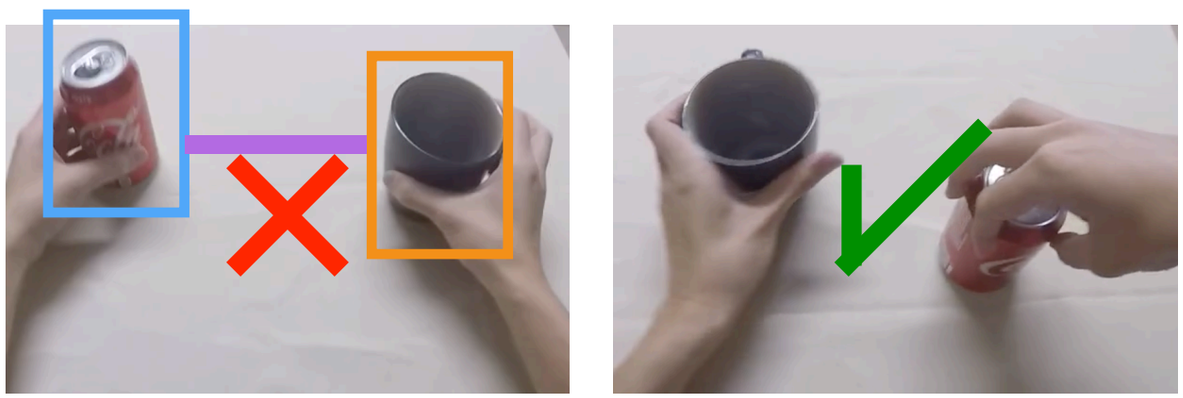
"*Can is to the right of the mug*"



Modeling Relationships in Referential Expressions with Compositional Modular Networks, Ronghang et al.

Reward learning using natural language

"*Can is to the right of the mug*"

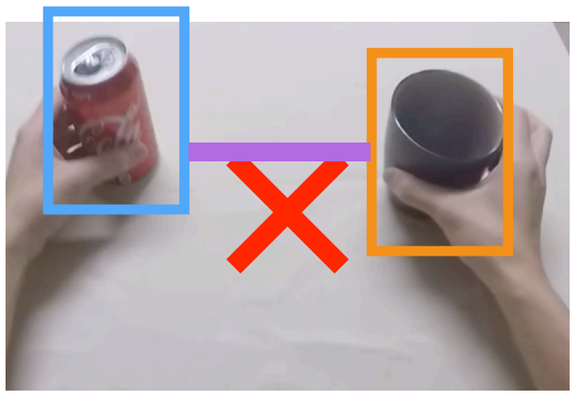


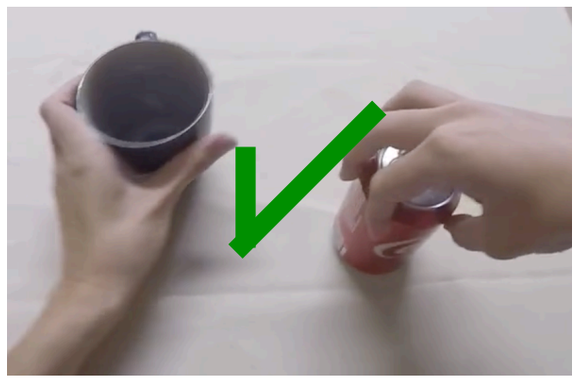
reward detector

Modeling Relationships in Referential Expressions with Compositional Modular Networks, Ronghang et al.

Reward learning using natural language

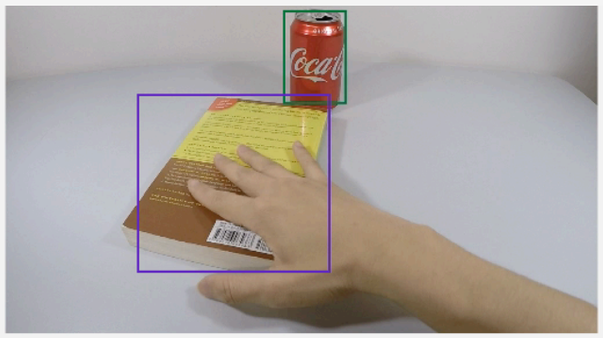
"*Can is to the right of the mug*"



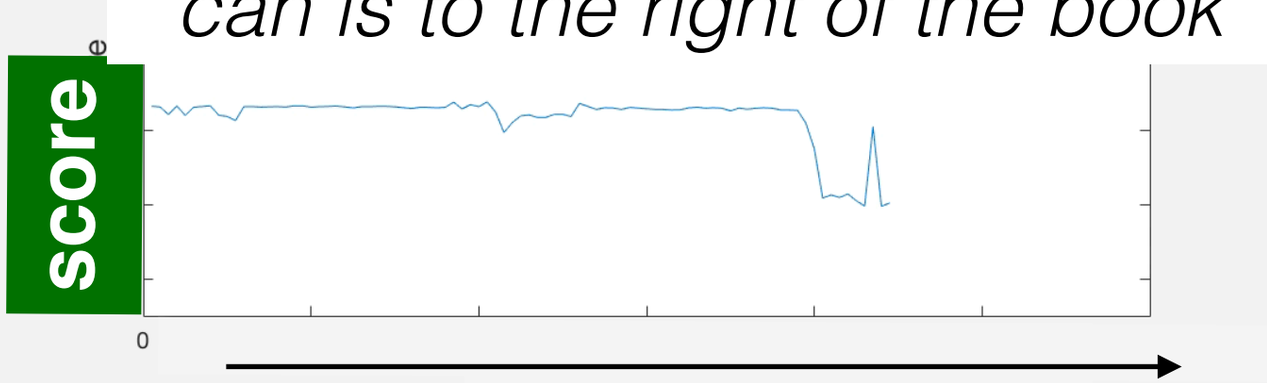


reward detector

Learned reward detector



``*can is to the right of the book*"



Learned policy

Reward learning using natural language

“Can is to the right of the mug”

Our conclusions are that the reward detector:

- could not effectively generalize across camera placements
- could not provide shaped rewards
- could not distinguish impossible goals for possible ones, e.g., “*the mug inside the coca cola”* versus “*the coca cola inside the mug”*

Tung et al. CVPR 2018

People can infer affordability of utterances.

- *“After wading barefoot in the lake, Erik used his shirt to dry his feet.”*
- *“After wading barefoot in the lake, Erik used his glasses to dry his feet.”*

Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories of Meaning, Glenberg and Robertson, 2000

People can infer affordability of utterances.

- *“He used the newspaper to protect his face from the wind.”*
- *“He used the matchbox to protect his face from the wind.”*

Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories of Meaning, Glenberg and Robertson, 2000

Computational models of language and vision

…cannot infer affordability of language utterances

- *“The bowl inside the cube”*
- *“The cube inside the bowl”*

Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories of Meaning, Glenberg and Robertson, 2000

Simulation Semantics

We understand utterances by simulating their content, using similar constructs to perception and control

Embodiment, simulation and meaning, Bergen, How reading comprehension is embodied and why that matters , Glenberg, Grounding language in action, Glenberg and Kaschak, Grounding Meaning in Affordances, Glenberg

2D boxes or 2D CNN activations do not have any affordance information attached.

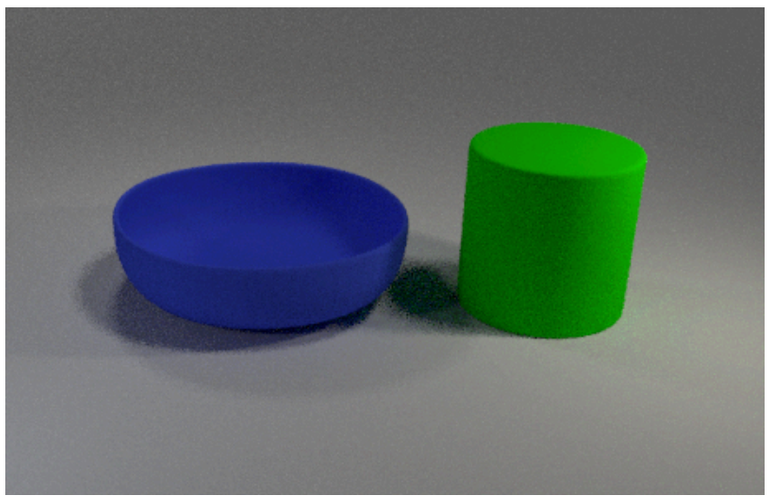
They are themselves **ungrounded :-(**

Affordable visual representations

We seek visual feature representations to ground NL onto that obey basic spatial common sense constraints:

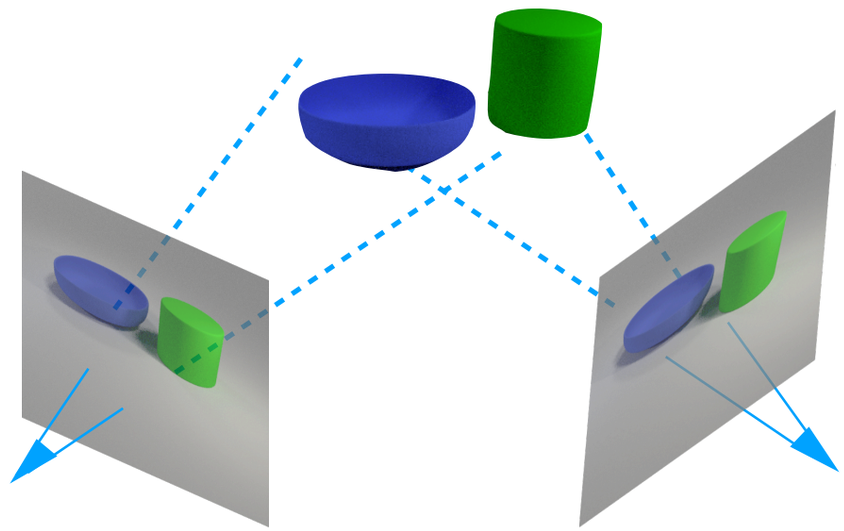
- Objects have 3D extent
- Objects do not interpenetrate in 3D
- Objects come in regular sizes
- Objects persist over time

“The green rubber cylinder is on the right of the blue bowl”



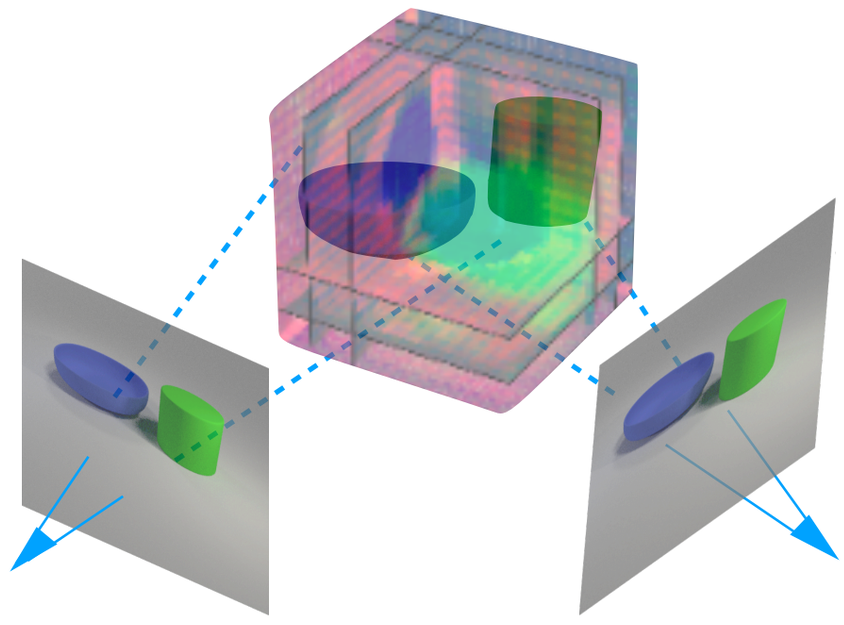
1. We consider an embodied agent that can see a scene from multiple viewpoints

“The green rubber cylinder is on the right of the blue bowl”



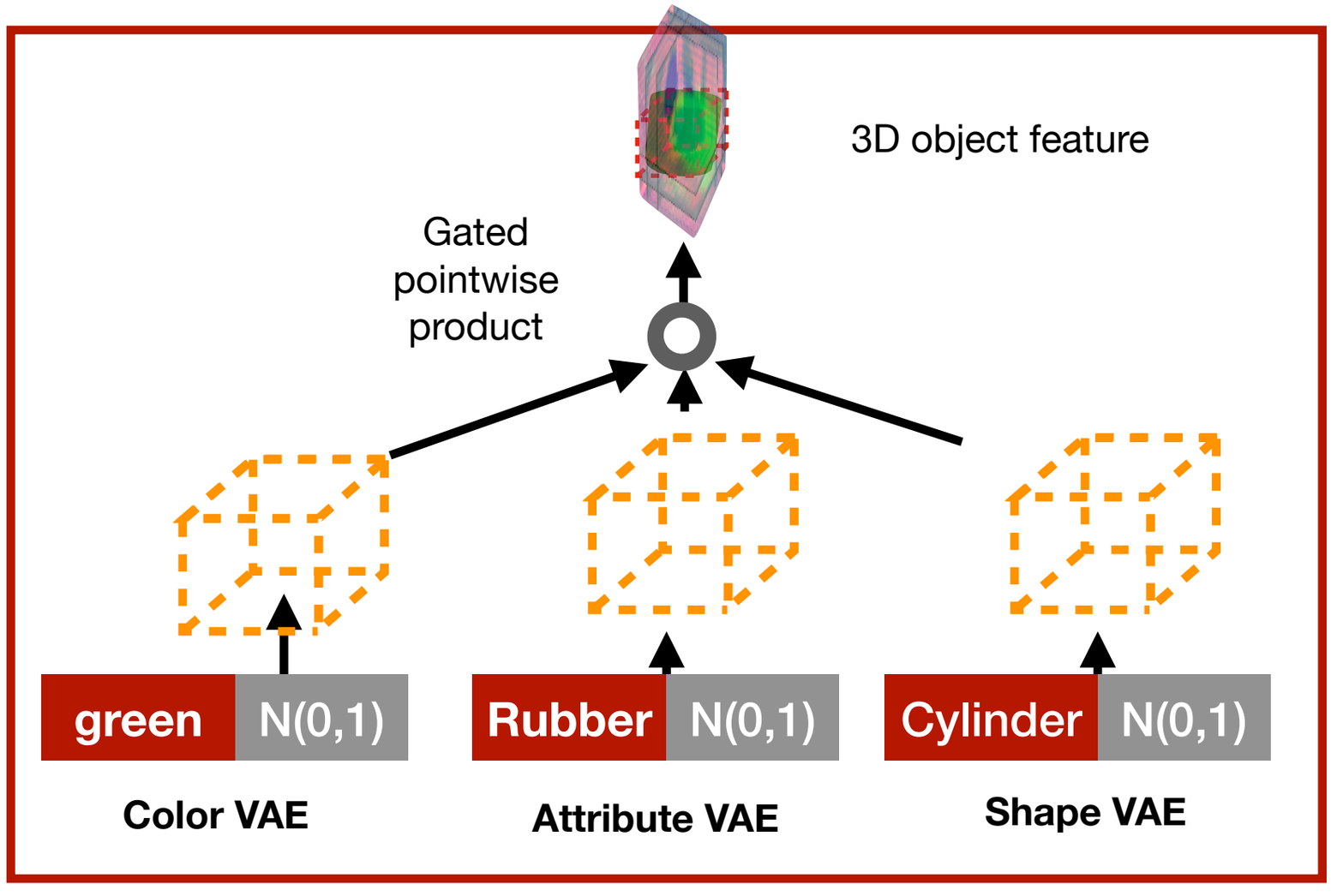
1. We consider an embodied agent that can see a scene from multiple viewpoints

“The green rubber cylinder is on the right of the blue bowl”



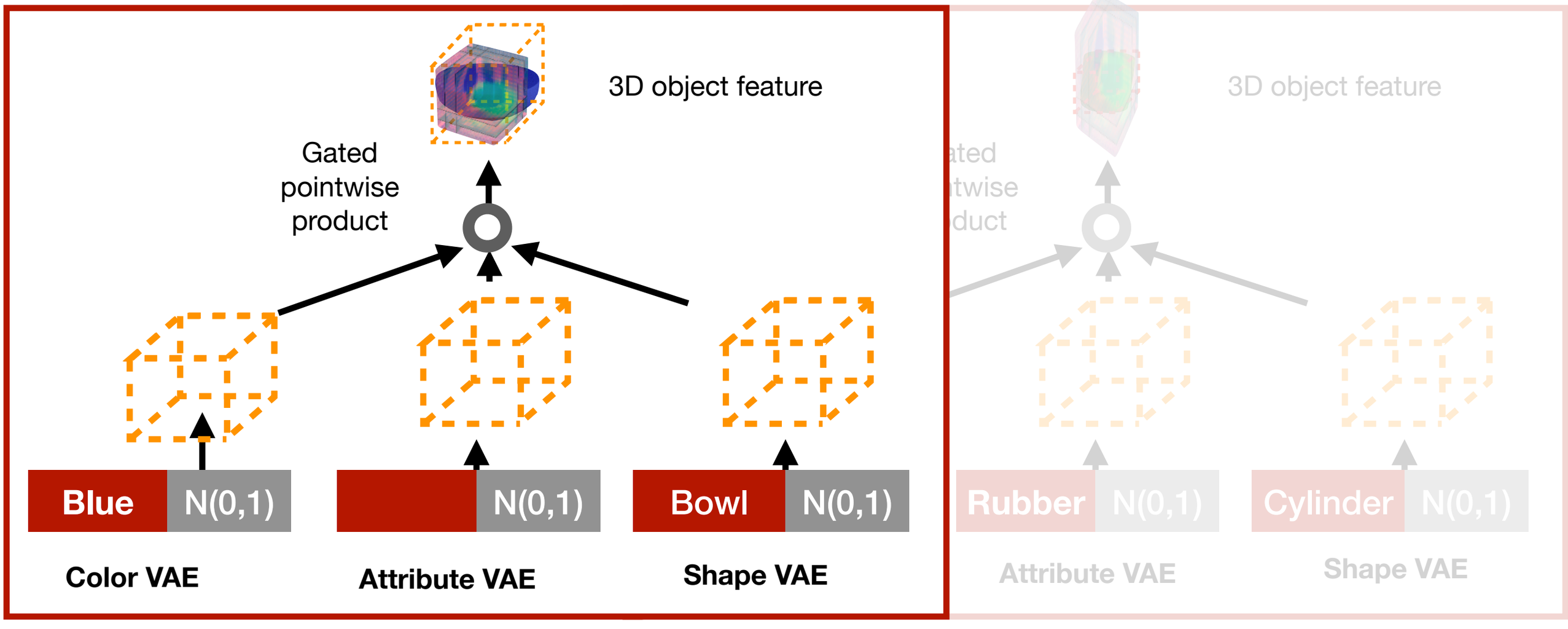
2. Our agent learns to map an RGB image to a set of 3D feature maps by training GRNNs to predict views

*"The **green rubber cylinder** is on the right of the blue bowl"*



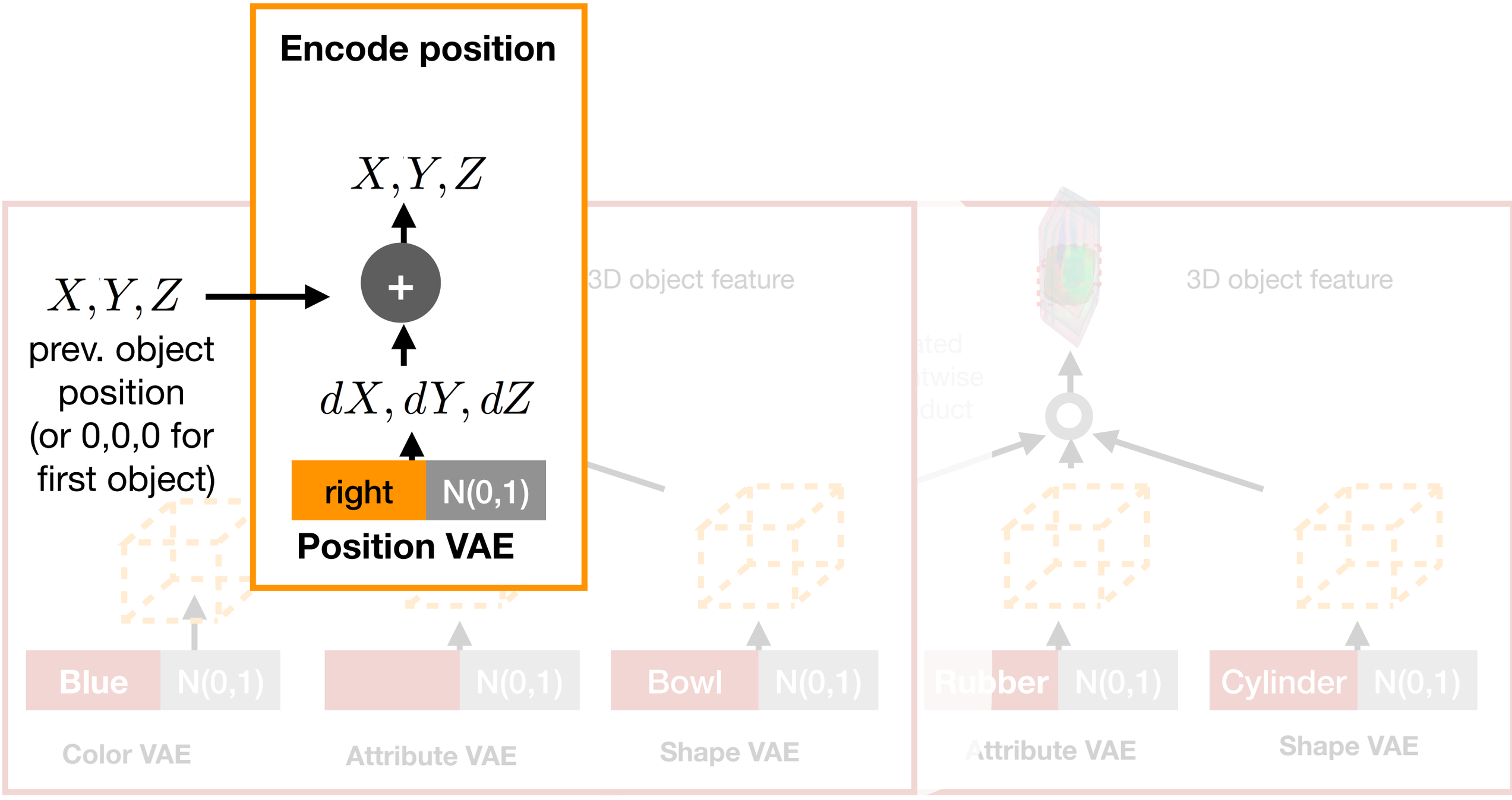
3. Our agent **maps noun phrases to object-centric 3D feature maps**

*"The green rubber cylinder is on the right of the **blue bowl**"*



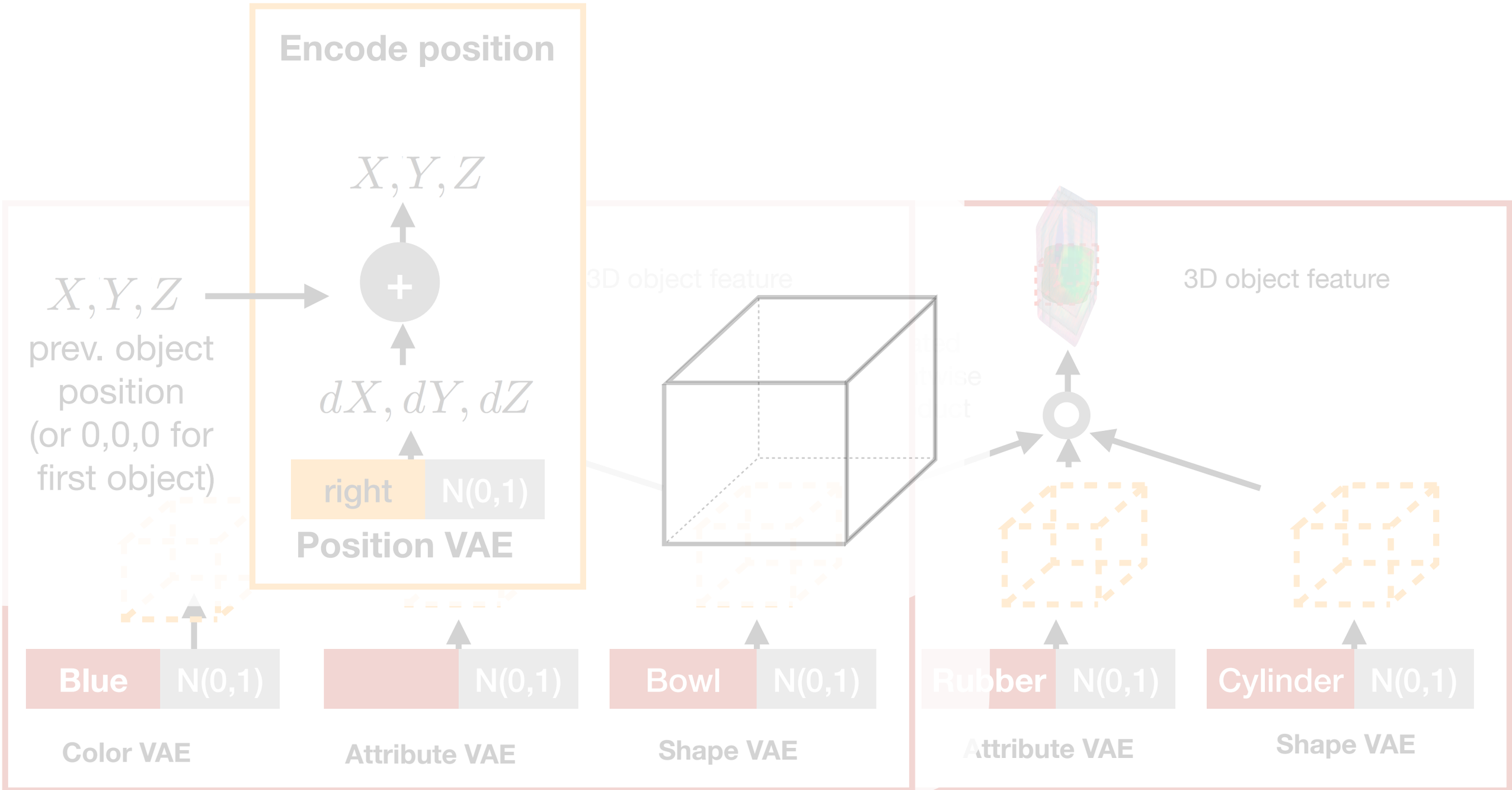
3. Our agent **maps noun phrases to object-centric 3D feature maps**

"The green rubber cylinder is
***on the right of** the blue bowl"*



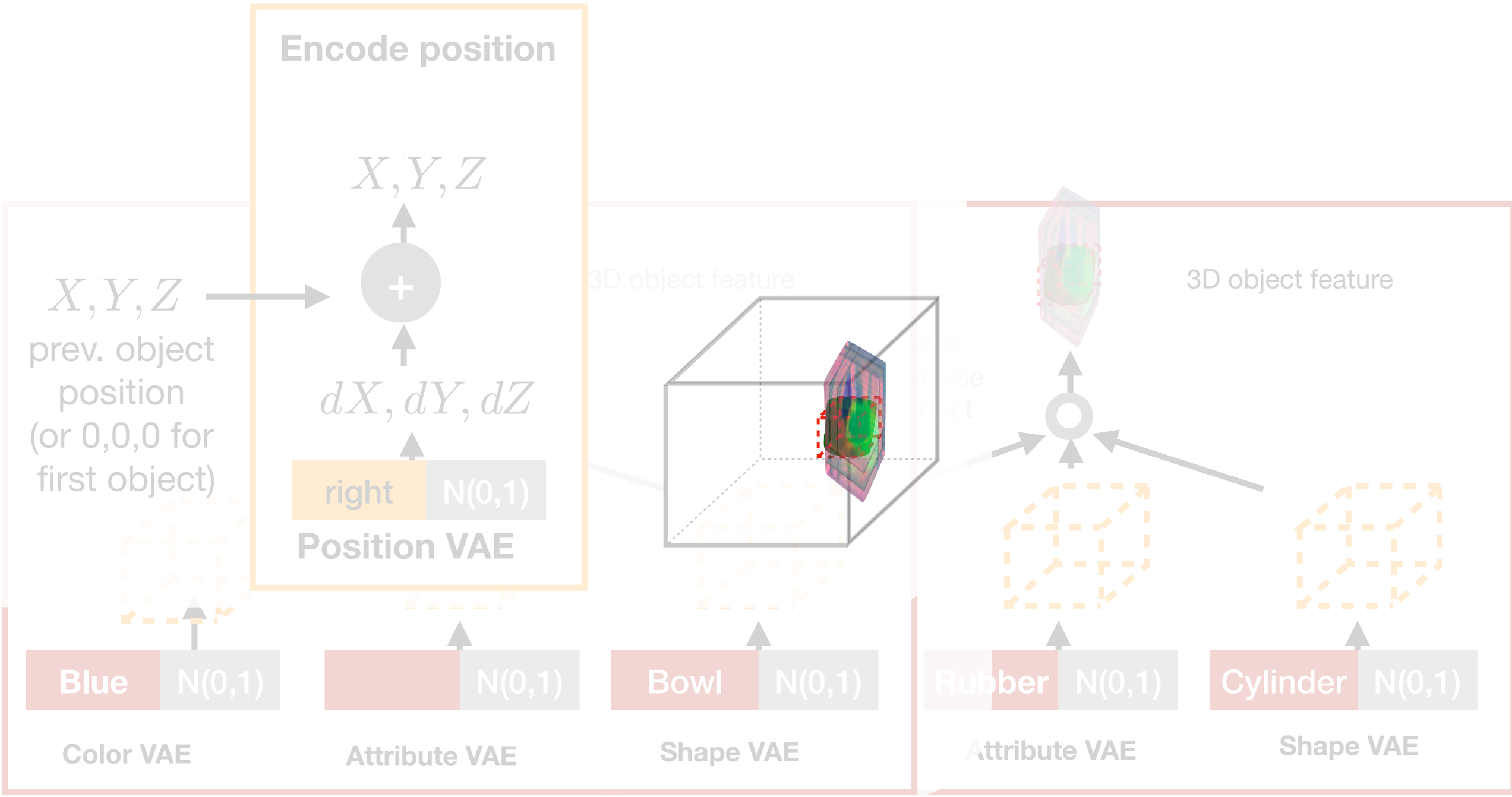
4. Our agent **maps spatial expressions to relative 3D offsets**

“The green rubber cylinder is on the right of the blue bowl”



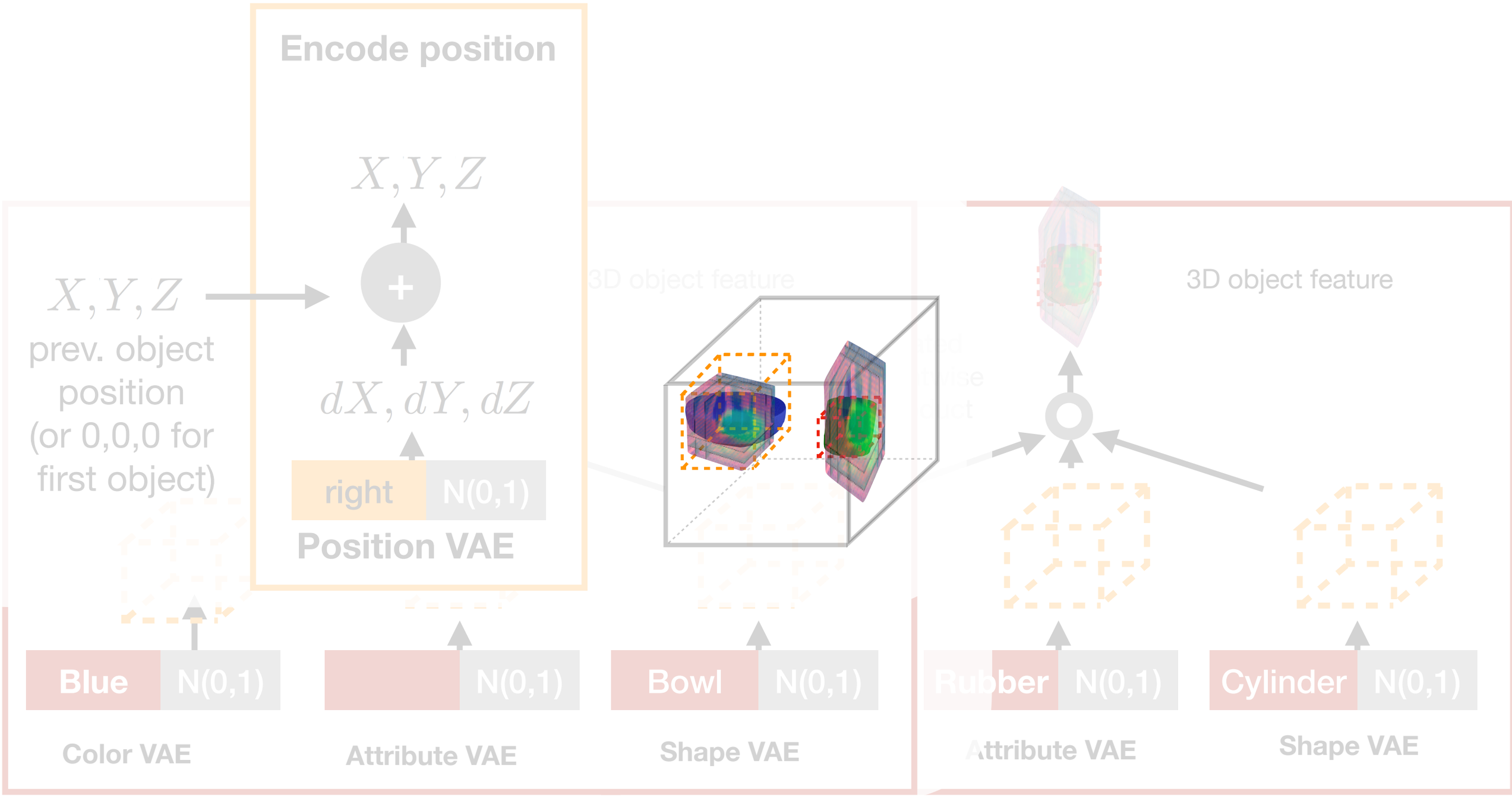
5. Our agent populates a 3D canvas with the predicted object tensors and their relative offsets

"The green rubber cylinder is on the right of the blue bowl"



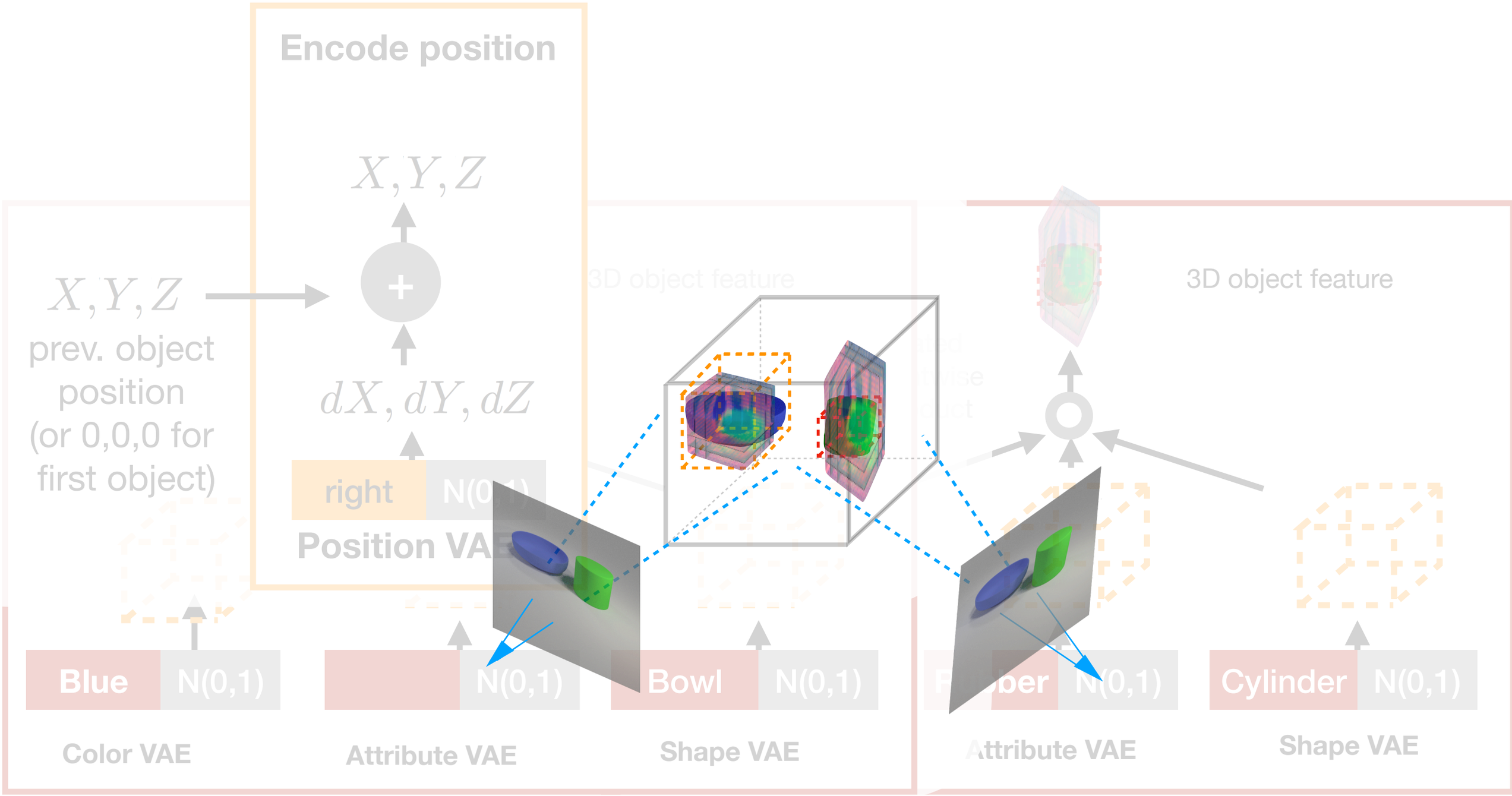
5. Our agent populates a 3D canvas with the predicted object tensors and their relative offsets

"The green rubber cylinder is on the right of the blue bowl"



5. Our agent populates a 3D canvas with the predicted object tensors and their relative offsets

"The green rubber cylinder is on the right of the blue bowl"



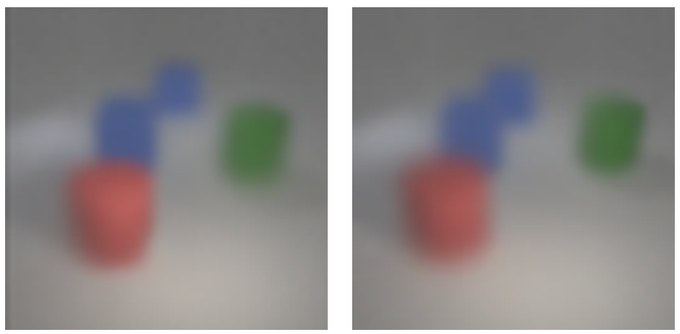
6. The generated canvas when projected should match the RGB image views

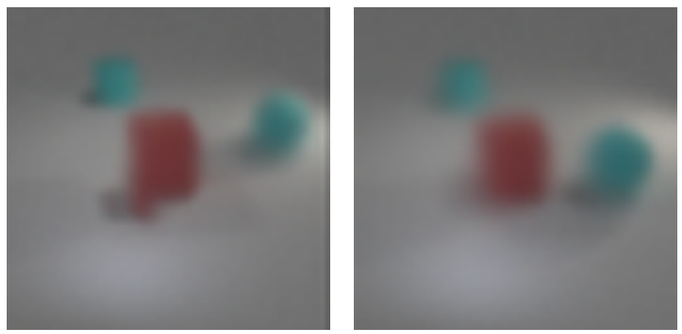
Scene imagination

“Red Rubber Cylinder to the left front of Blue Rubber Cube to the left front of Green Rubber Cylinder to right front of Blue Rubber Cube”

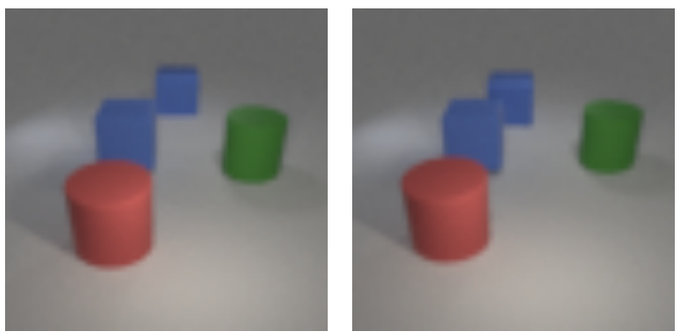
“Red Rubber Cube to the left front of the Blue Rubber Sphere to the right front of Cyan Metal Cylinder”

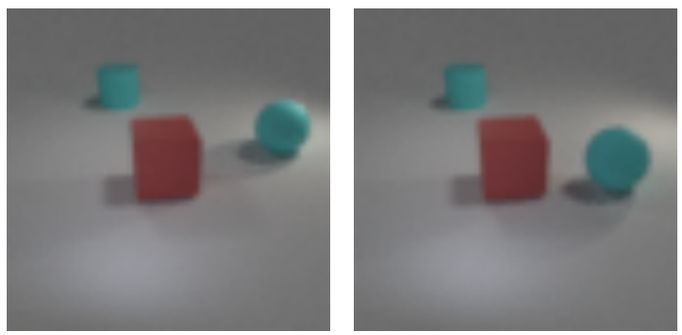
Neural rendering





Blender rendering





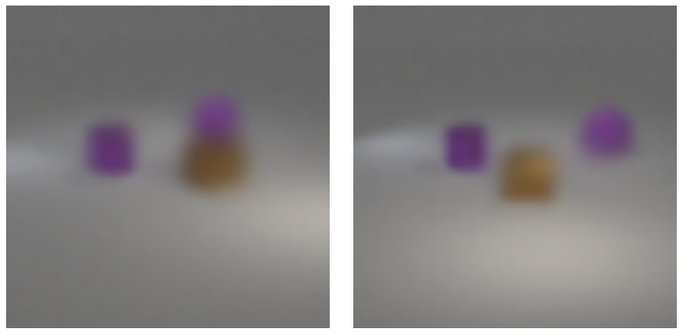
- **Neural rendering**: project the 3D feature maps using our learned project+RGB decoder neural module
- **Blender rendering**: use the object-centric 3D feature maps to retrieve nearest 3D mesh neighbors from a training set, then arrange the retrieved meshes based on predicted 3D spatial offsets

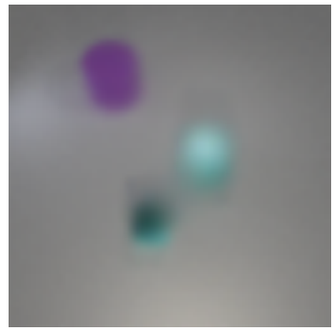
Scene imagination

“Purple Cylinder to the left behind of Brown Cube to the left front of Purple Sphere”

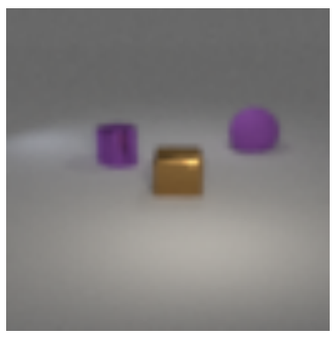
“Purple Cylinder to the left behind of Cyan Cube to the left front of Cyan Cube”

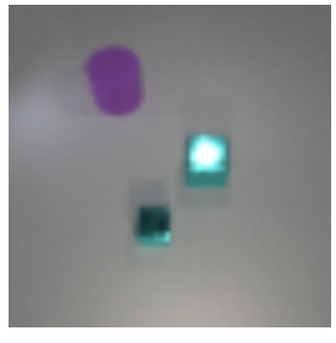
Neural rendering





Blender rendering





- **Neural rendering**: project the 3D feature maps using our learned project+RGB decoder neural module
- **Blender rendering**: use the object-centric 3D feature maps to retrieve nearest 3D mesh neighbors from a training set, then arrange the retrieved meshes based on predicted 3D spatial offsets

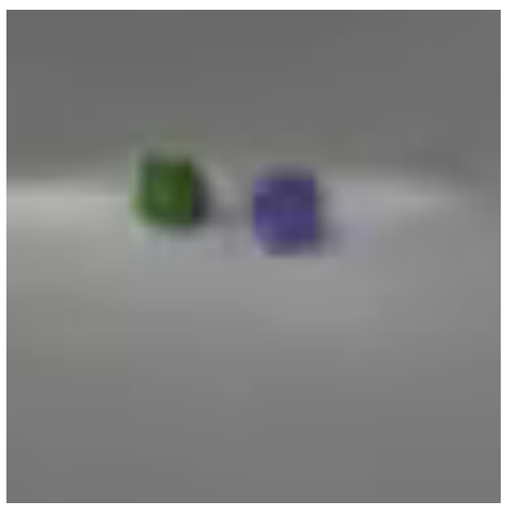
Scene imagination

“cyan sphere red cube”

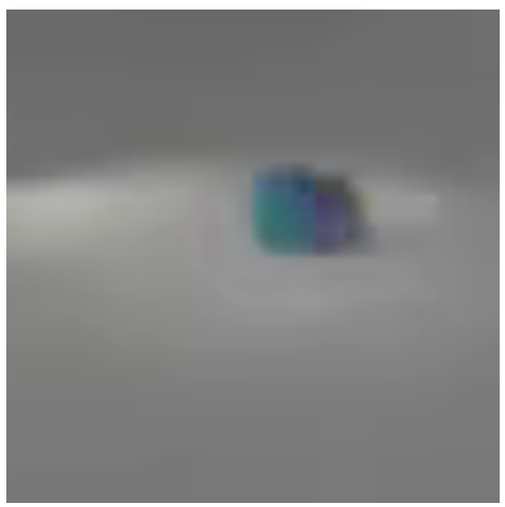
“red cylinder red sphere blue sphere”

“cyan cylinder red sphere green sphere”

“blue sphere green cube”



“cyan cylinder red sphere green sphere”



“cyan cylinder yellow sphere green sphere blue sphere gray cylinder red sphere”

Grounding arbitrarily long utterances

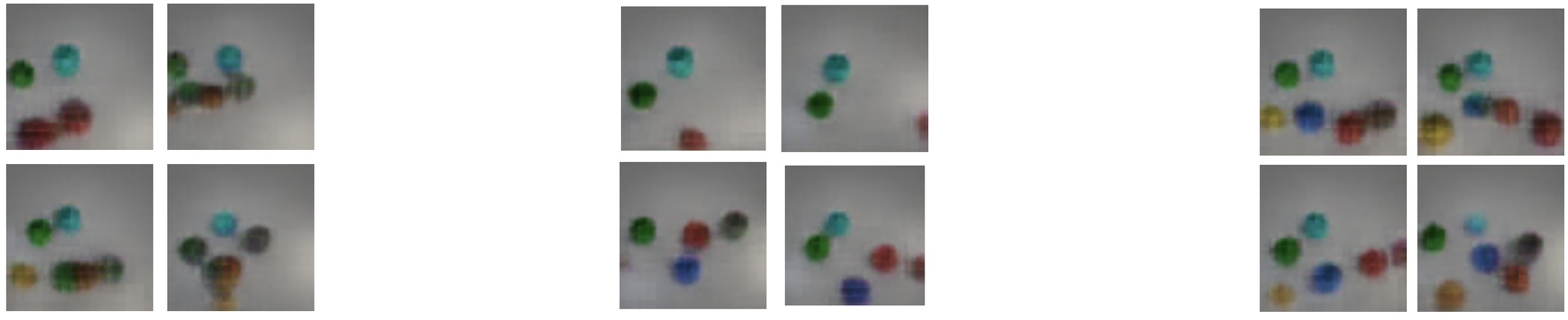
“yellow sphere to the left front of green sphere to the left behind of blue sphere to the left front of blue cylinder to the left behind of red cube to the left front of gray cube”

IOU > 0.1

Object Out of Camera View

IOU= 0

Top View



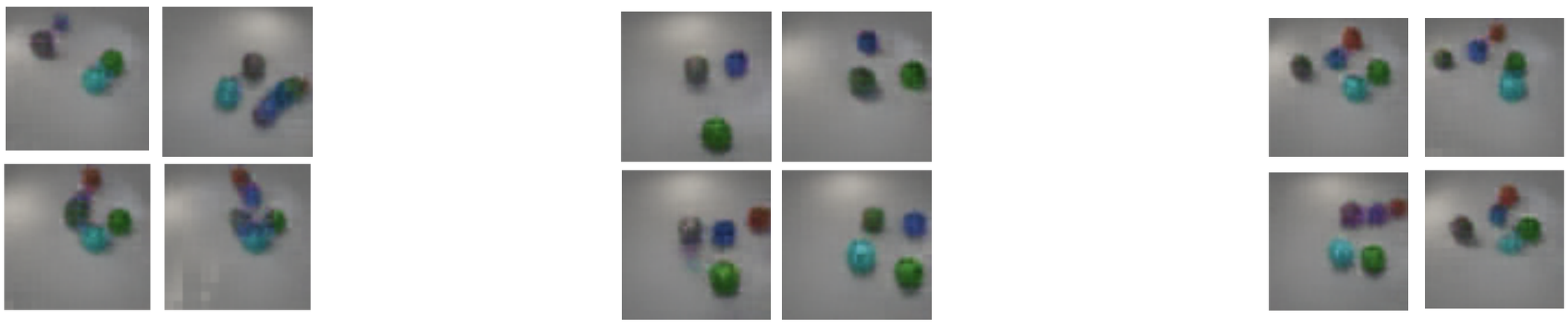
“gray sphere to the left front of blue sphere to the left front of red sphere to the left behind of cyan sphere to the left behind of green sphere”

IOU > 0.1

Object Out of Camera View

IOU= 0

Top View

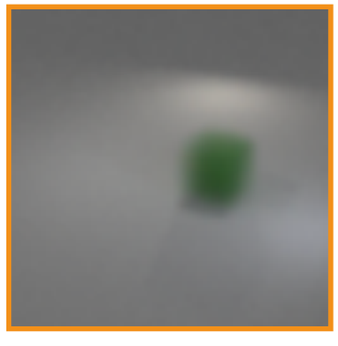


Scene alteration

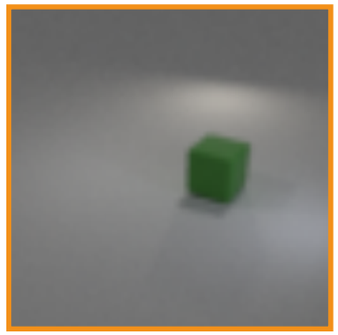
“blue sphere to the right behind of green cube”

“green cube to the left front of cyan cylinder”

Neural rendering



Blender rendering



Condition on both an image and a NL description:

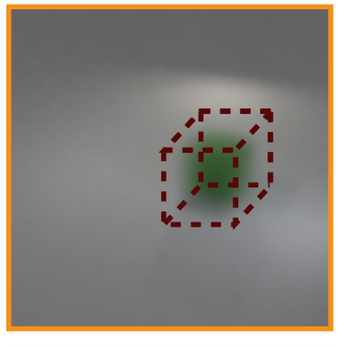
1. we predict the 3D feature tensors from NL description,

Scene alteration

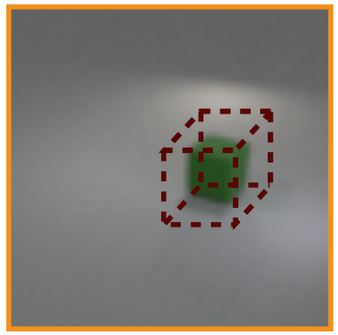
“blue sphere to the right behind of green cube”

“green cube to the left front of cyan cylinder”

Neural rendering



Blender rendering



Condition on both an image and a NL description:

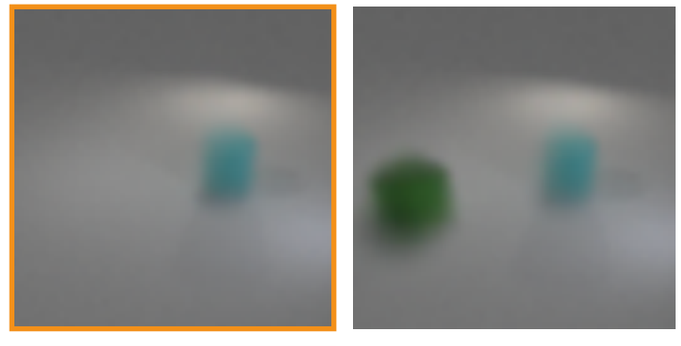
1. we predict the 3D feature tensors from NL description,
2. we extract the object 3D feature tensors by running our object detector
3. we compare them to ground NL referents to actual objects in the scene.

Scene alteration

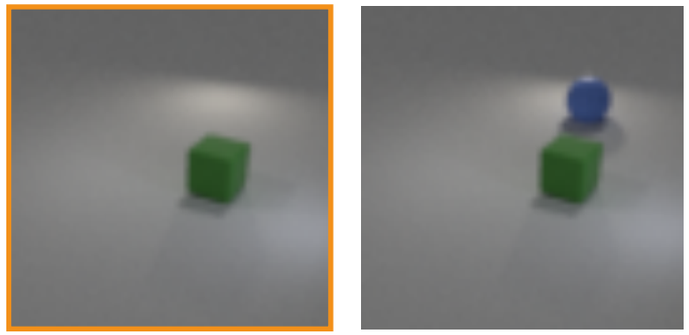
“blue sphere to the right behind of green cube”

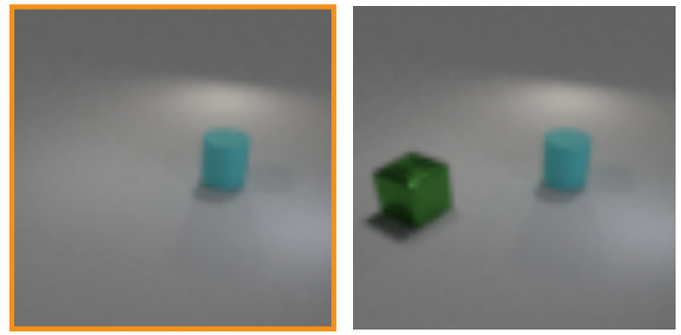
“green cube to the left front of cyan cylinder”

Neural rendering



Blender rendering





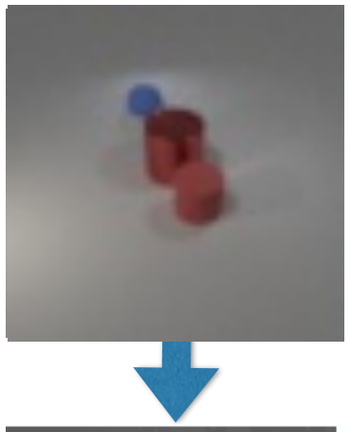
Condition on both an image and a NL description:

1. we predict the 3D feature tensors from NL description,
2. we extract the object 3D feature tensors by running our object detector
3. we compare them to ground NL referents to actual objects in the scene.

Then, we alter the image by adding objects to the scene according to the NL description

3D referential object detection

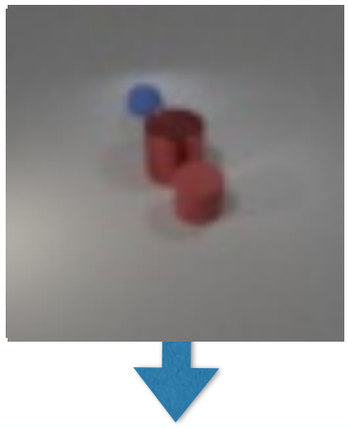
query *"find red metal cylinder to the left behind of red rubber cylinder"*



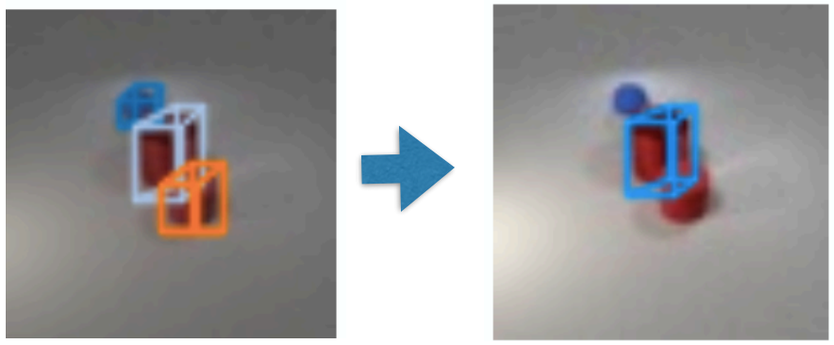
3D RPN

3D referential object detection

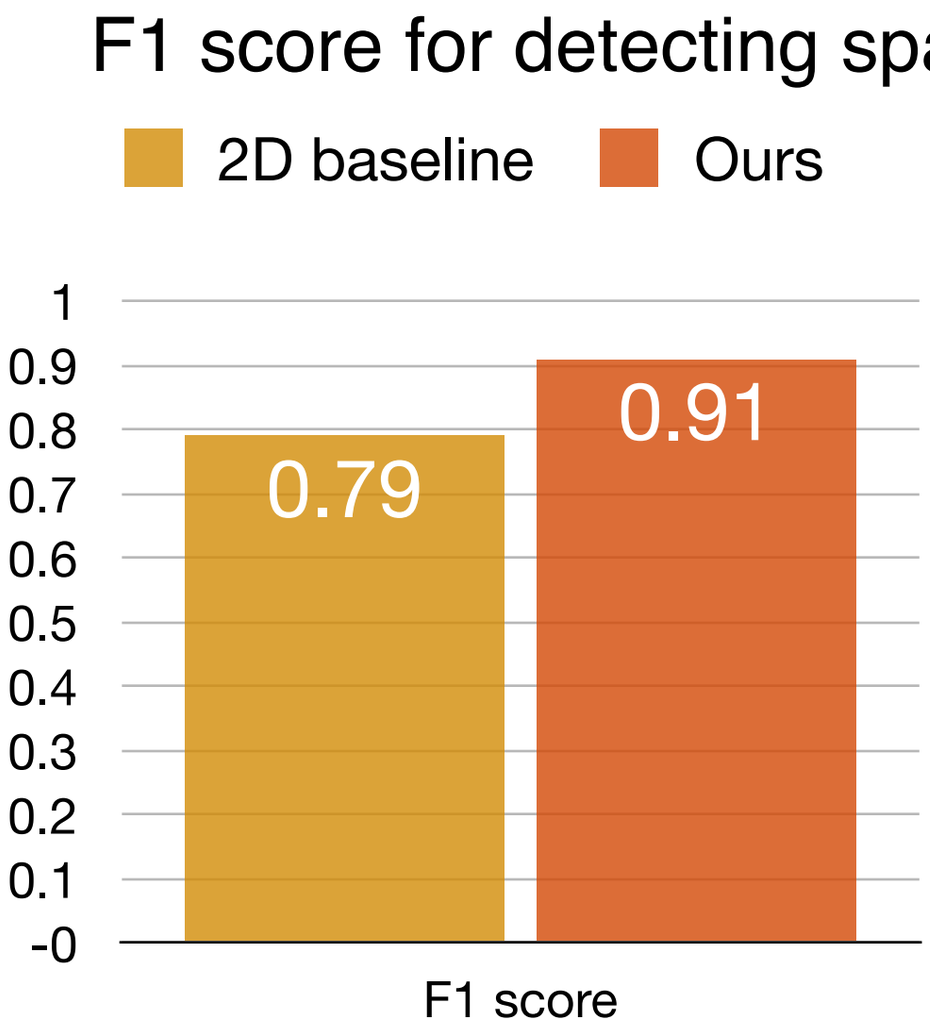
query *"find red metal cylinder to the left behind of red rubber cylinder"*

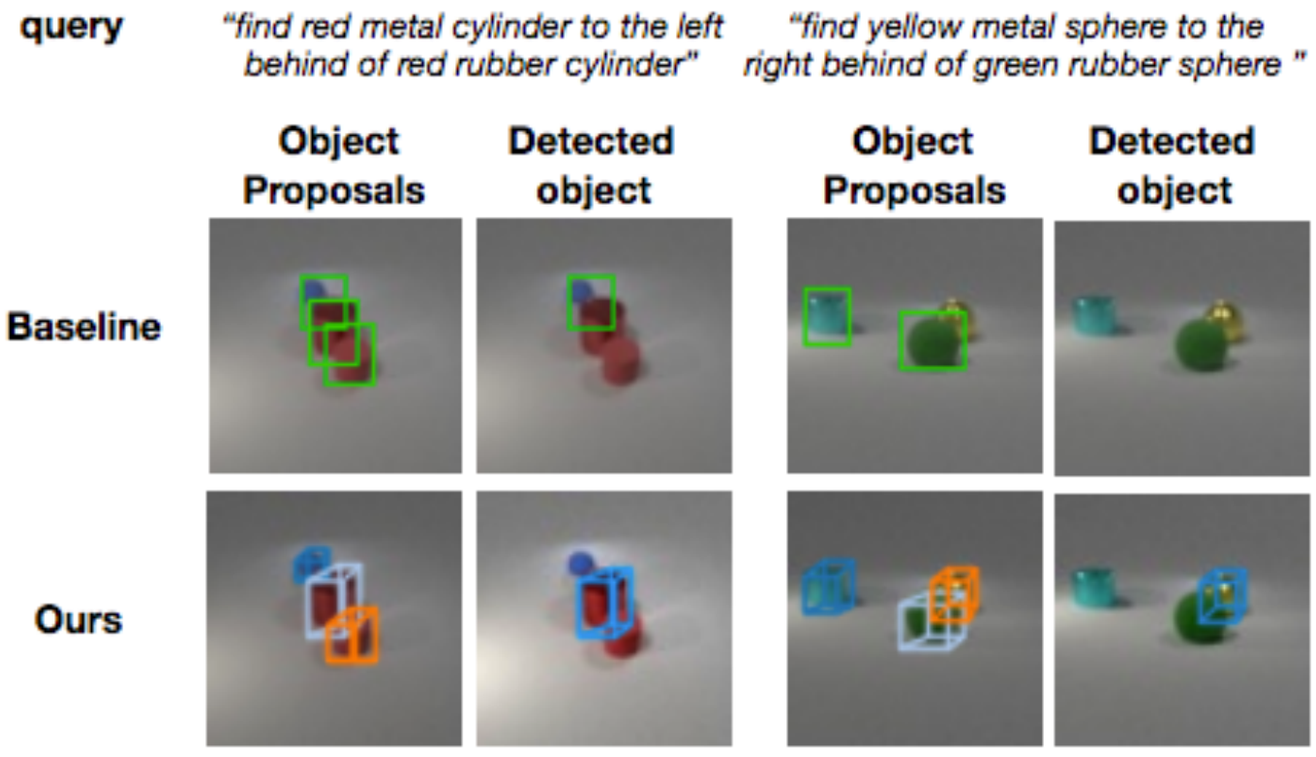


3D RPN



3D referential object detection





Ronghang Hu et al. Modeling relationships in referential expressions with compositional modular networks

Instruction Following

``*put the cube on the right of the bowl*”

1. Referential 3D object detection
2. Goal imagination: Predict relative 3D desired location for the object
3. Use LQR with Euclidean distance of current to goal location as the cost.