UNIVERSITY of WASHINGTON

Introduction to Robot Learning

CSE 571

Jiafei Duan, 3 June 2025

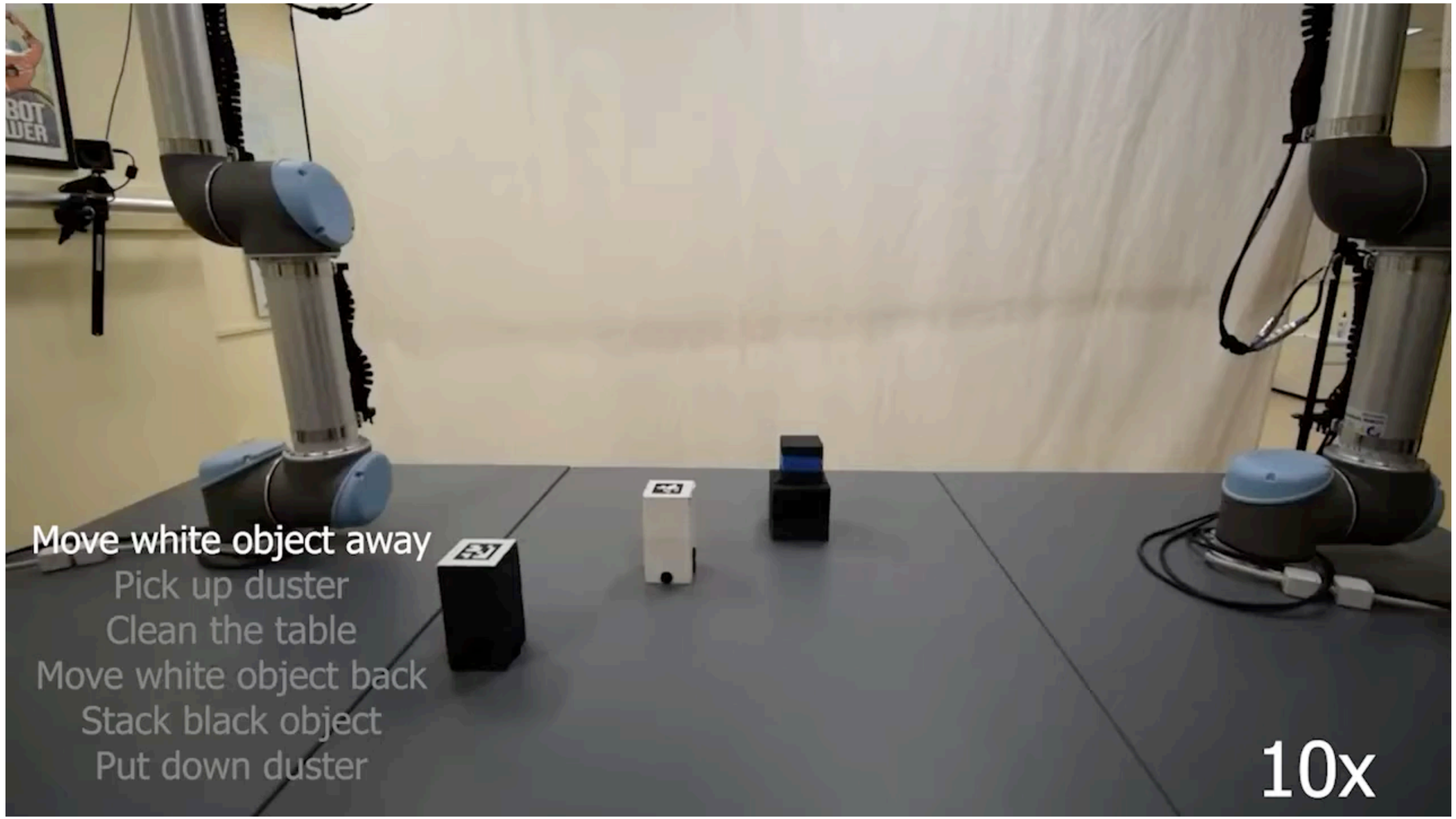
Used Materials

- Acknowledgement: Some of the material and slides for the lecture were borrowed from Shuran Song, Wenlong Huang, and Li Yi.

Things to cover today:

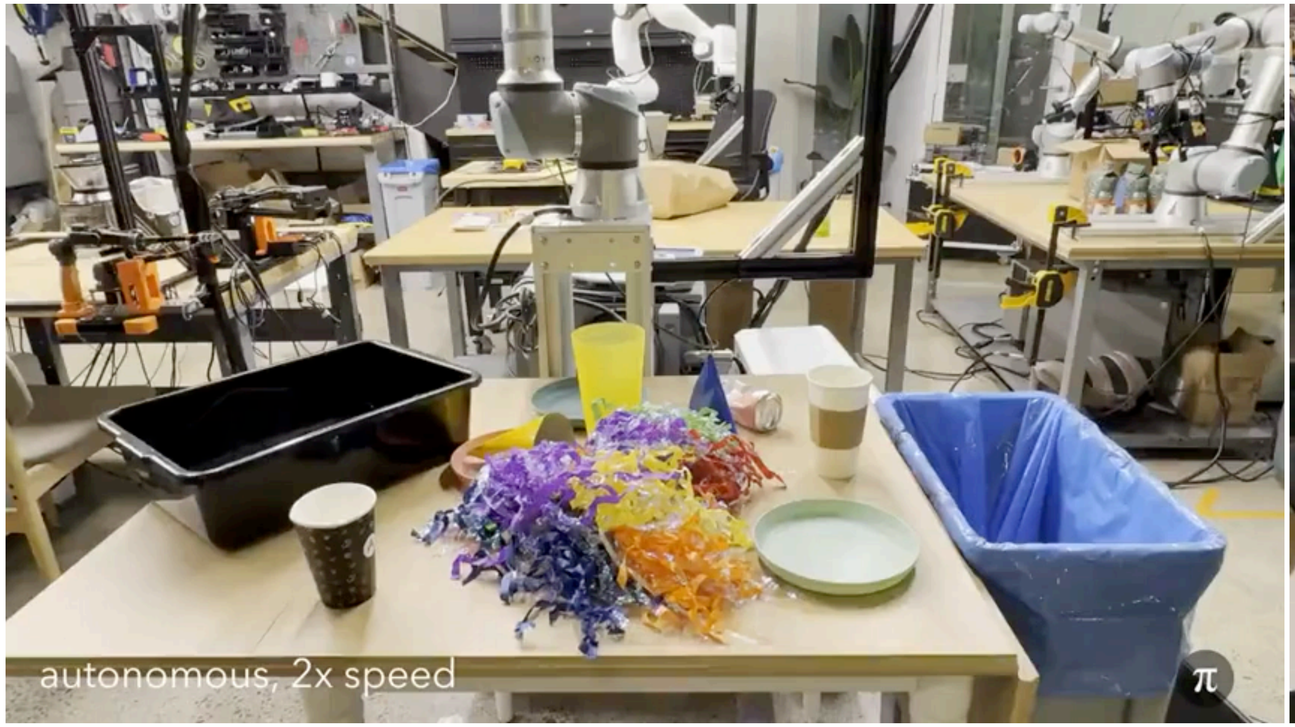
- What is empowered by robot learning?
- IBC
- Visuomotor Policy Learning
- Representational learning for Robotic Manipulation
- Vision-language-action Models

What is empowered by robot learning?



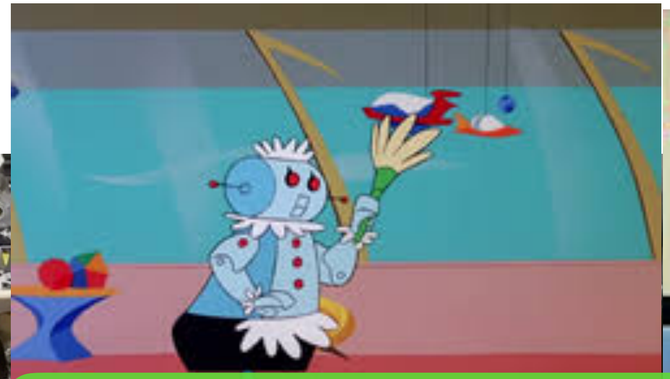
What is good and bad about this approach?

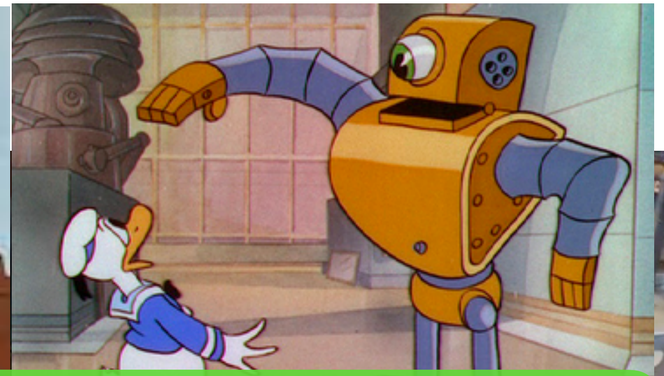
What is empowered by robot learning?

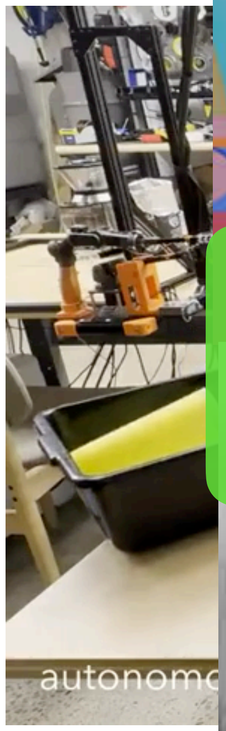


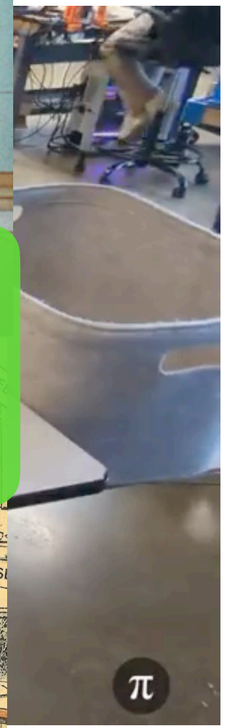
What is empowered by robot learning?

What is empowered by robot learning?

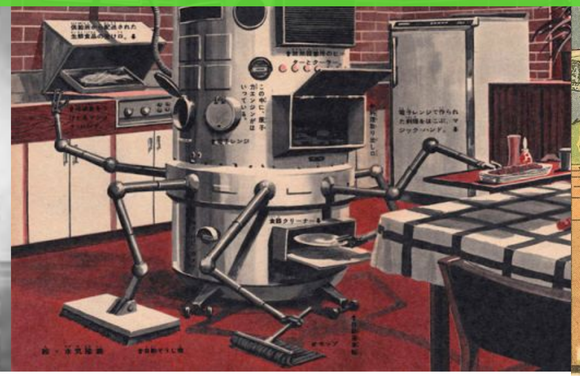


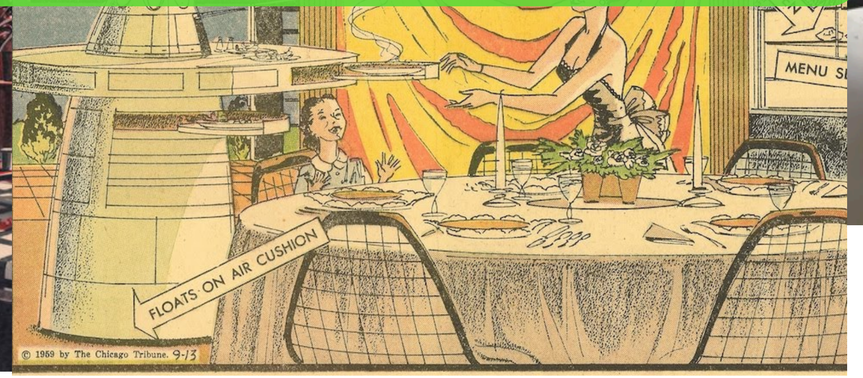






Robot Learning is the computational study of algorithms that enables robotics agents to acquire skills and adapt their behavior through environmental interaction and data-driven experience rather than explicit, hard-coded programming.





A quick recap of Behavior Cloning

We assume a dataset of expert demonstrations:

$$\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$$

where

- $s_i \sim d^{\pi_E}(s)$ (expert state distribution)
- $a_i = \pi_E(s_i)$

We parameterize a policy $\pi_\theta(a \mid s)$

Behavior Cloning solves:

$$\theta^* = \arg\max_\theta \sum_{(s,a)\in\mathcal{D}} \log \pi_\theta(a \mid s)$$

This is the Maximum Likelihood Estimation (MLE)

A quick recap of Behavior Cloning

We assume a dataset of expert demonstrations:

$$\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N}$$

where

- $s_i \sim d^{\pi_E}(s)$ (expert state distribution)
- $a_i = \pi_E(s_i)$

We parameterize a policy $\pi_\theta(a \mid s)$

Behavior Cloning solves:

$$\theta^* = \arg\max_\theta \sum_{(s,a)\in\mathcal{D}} \log \pi_\theta(a \mid s)$$

This is the Maximum Likelihood Estimation (MLE)

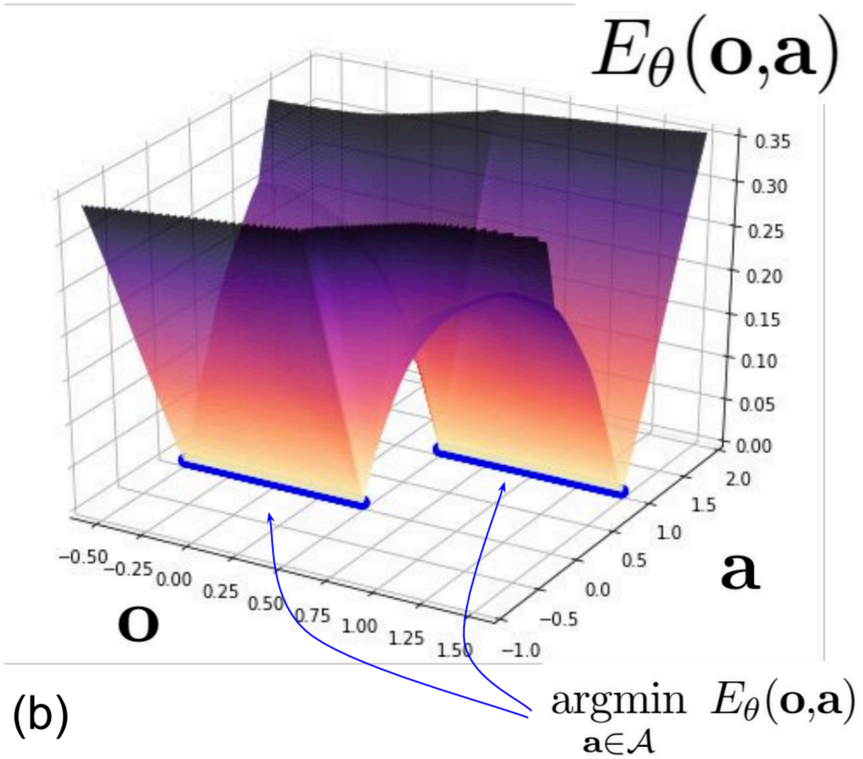
Compounding errors, covariate shift, ….

-> Collect on-policy data, get expert to label (DAgger)

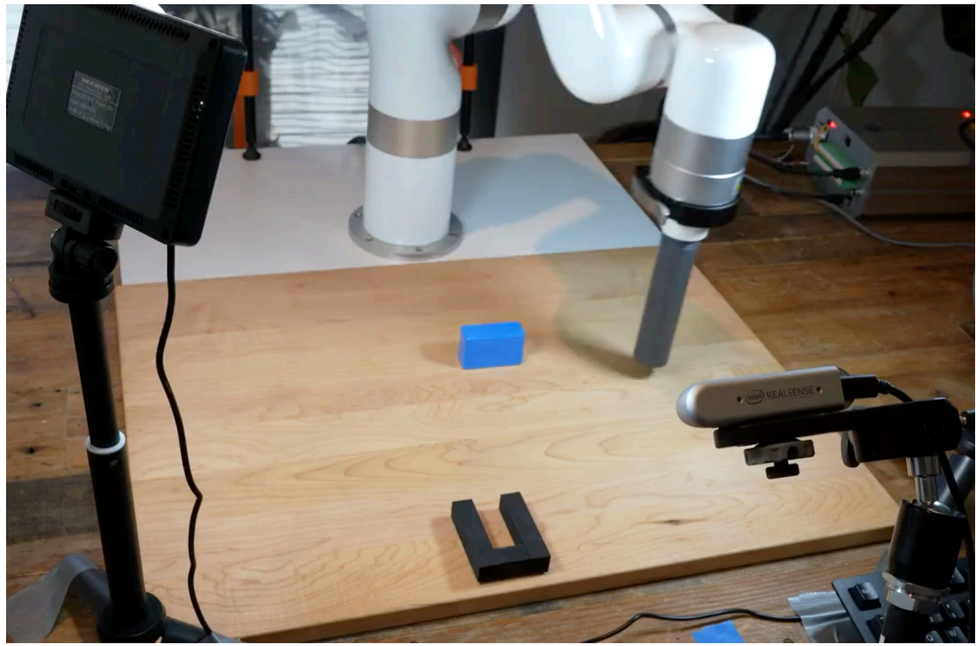
-> Collect on-policy data, distribution-match (GAIL)

-> Collect reward labels for demonstrator data (Offline RL)

Implicit Behavior Cloning

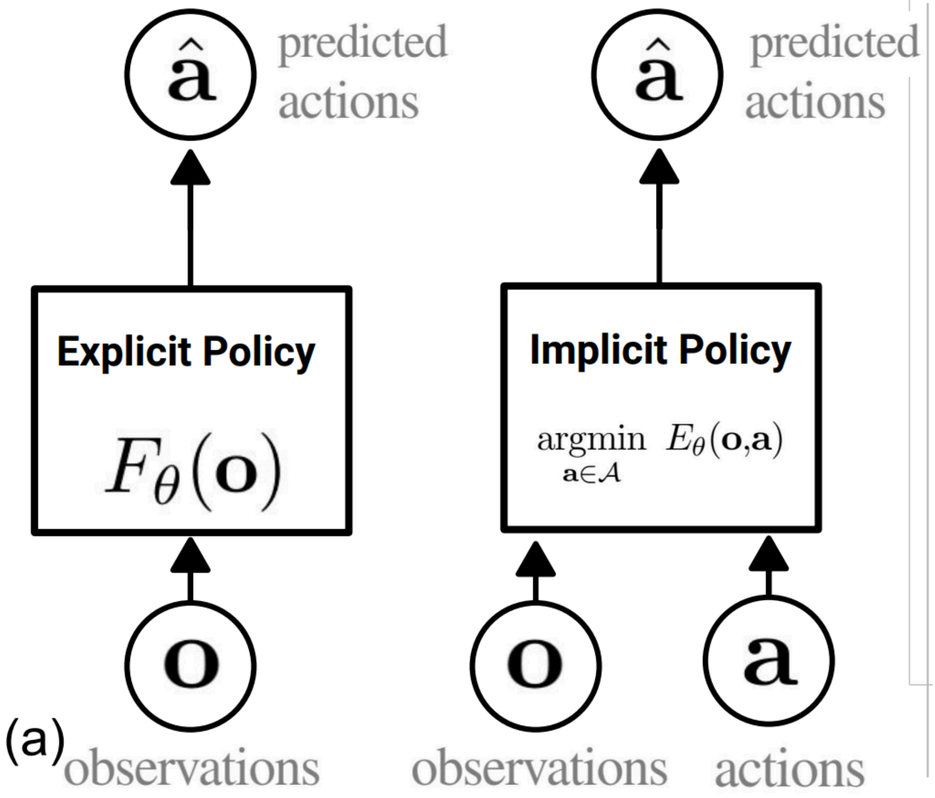


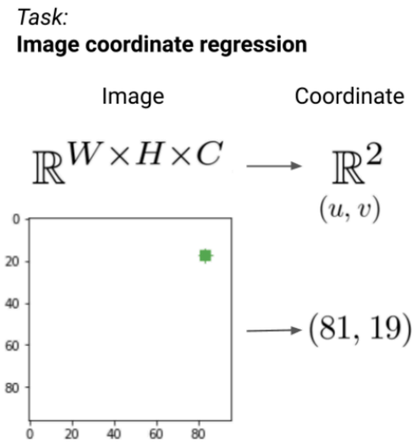
Discontinuous

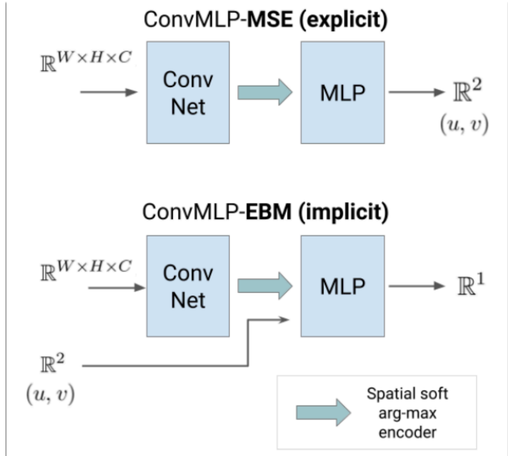


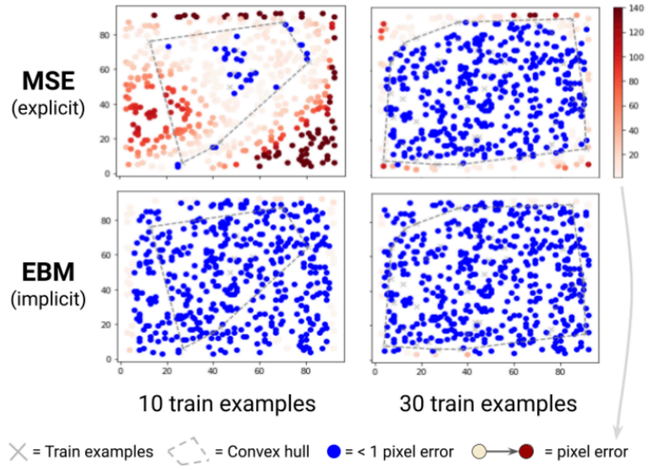
Multimodal distribution

Implicit Behavior Cloning

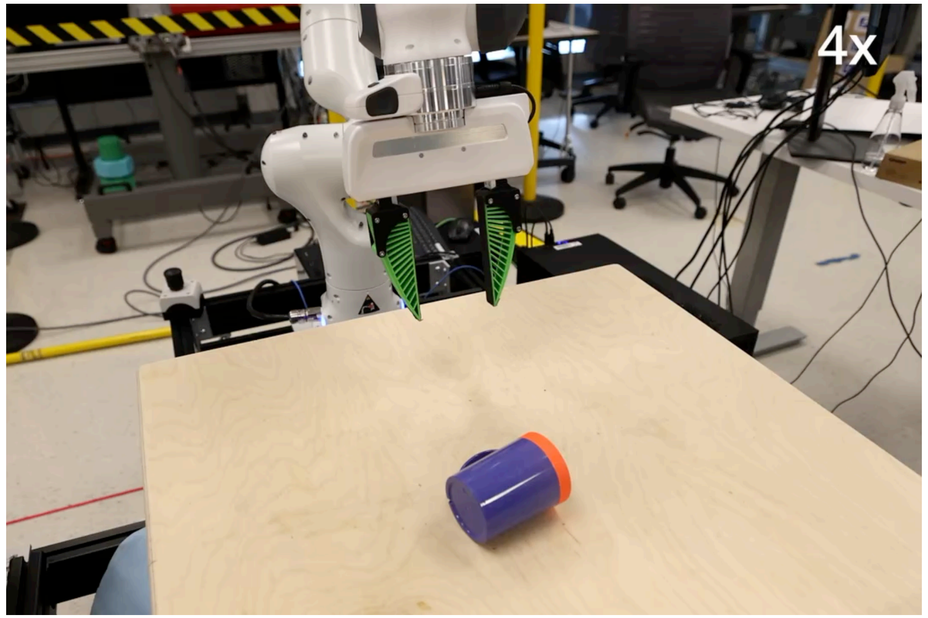




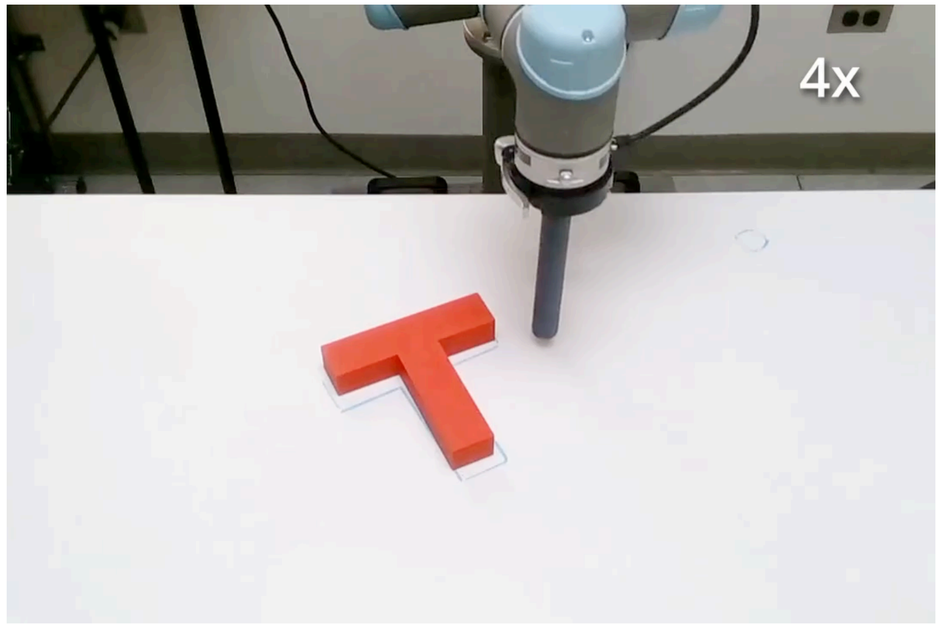


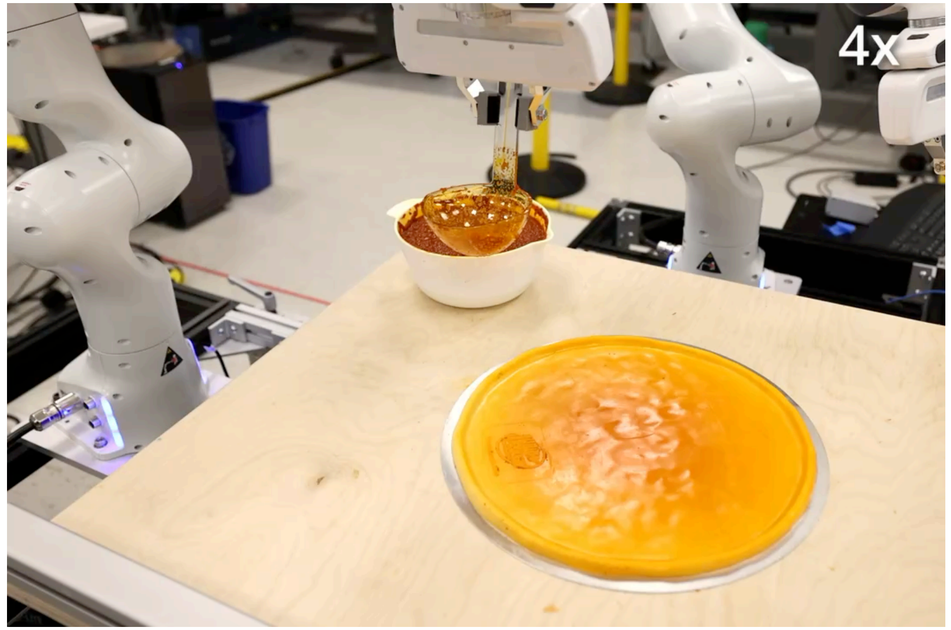


Visuomotor Policy Learning- Diffusion Policy



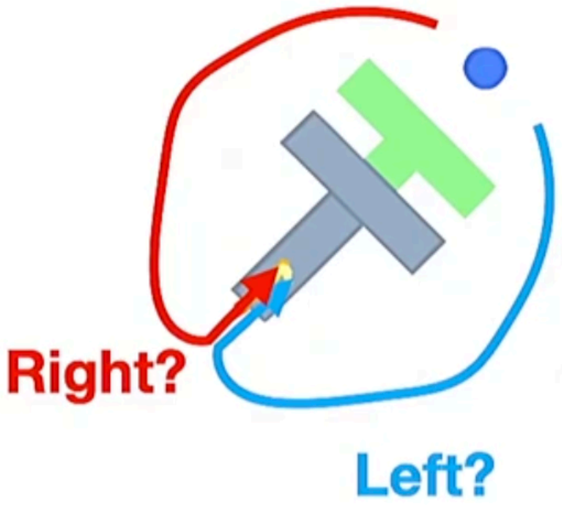
Switch between actions & recover from failures.

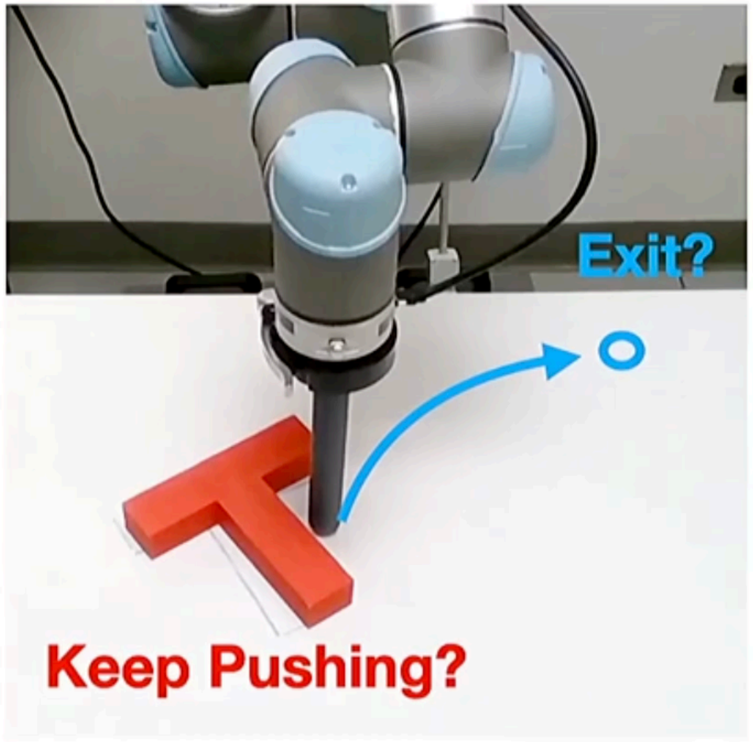


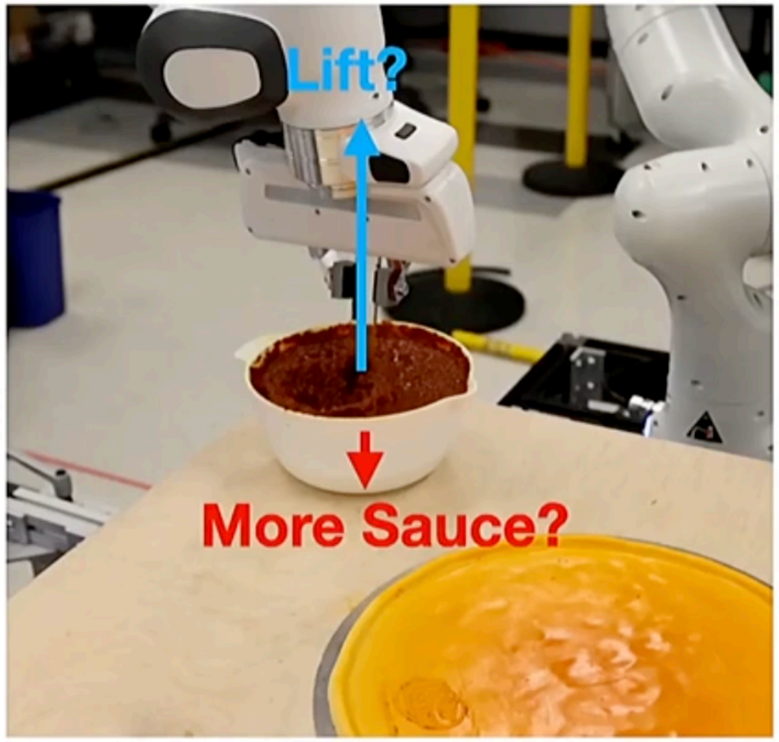


Visuomotor Policy Learning- Diffusion Policy

Action Multimodality





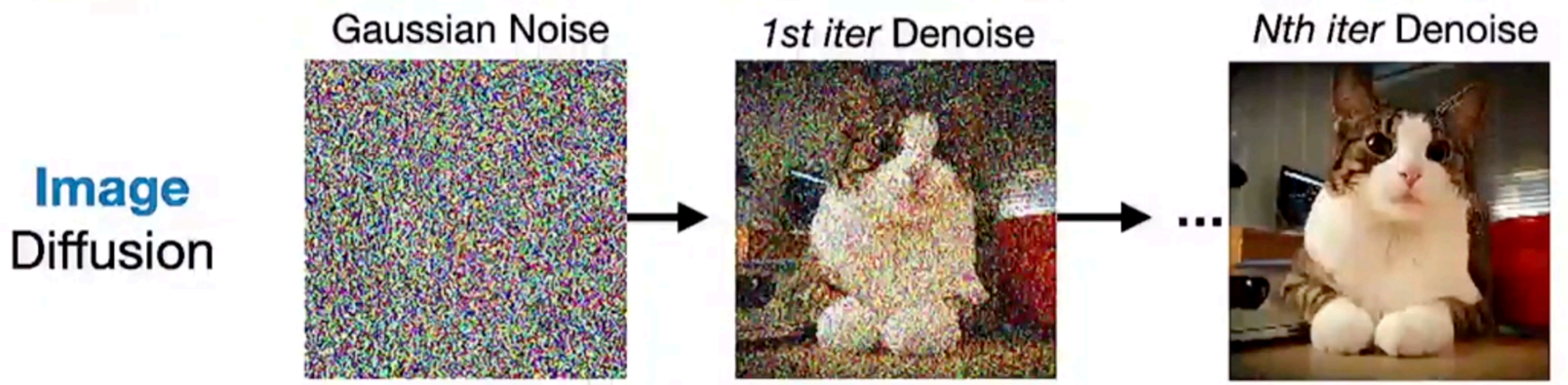


Key challenge for learning from demonstration

Slides from Shuran Song

Visuomotor Policy Learning- Diffusion Policy

Generative model for robot actions, learned from demonstrations



Forward Process (Add Noise):

$$q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\, x_{t-1}, \beta_t \mathbf{I}\right) \quad x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \quad \text{where } \epsilon \sim \mathcal{N}(0,\mathbf{I})$$

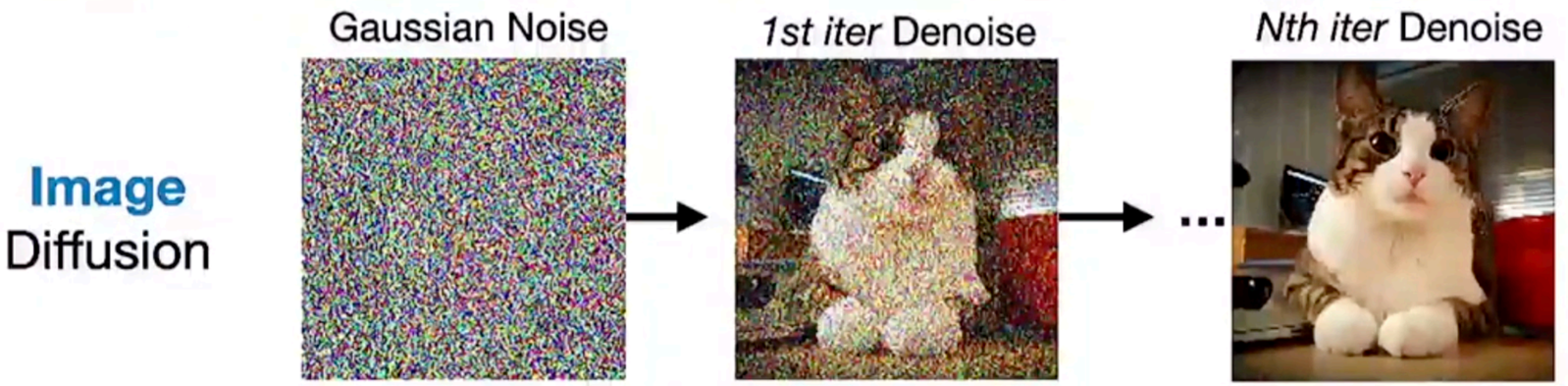
Reverse Process (Learn to Denoise):

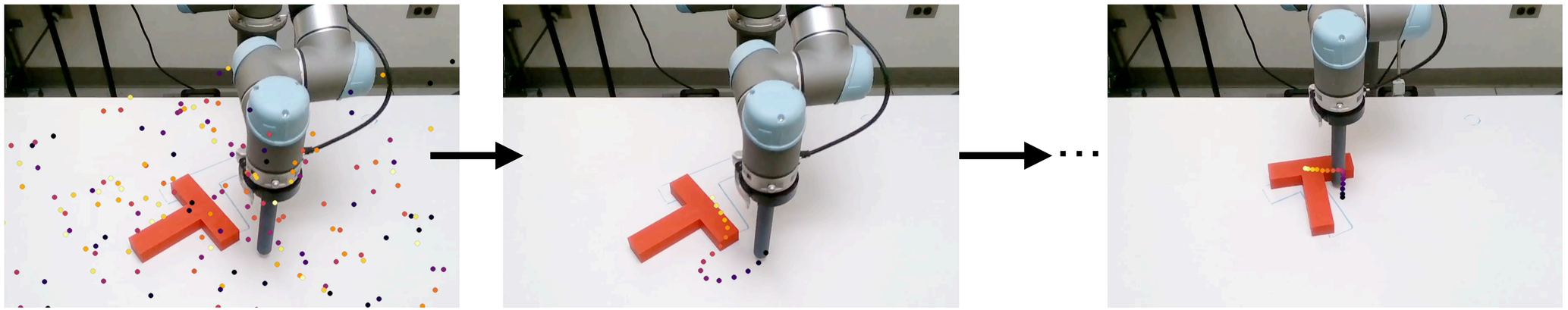
$$p_\theta(x_{t-1} \mid x_t) = \mathcal{N}\left(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)\right)$$

Slides from Shuran Song

Visuomotor Policy Learning- Diffusion Policy

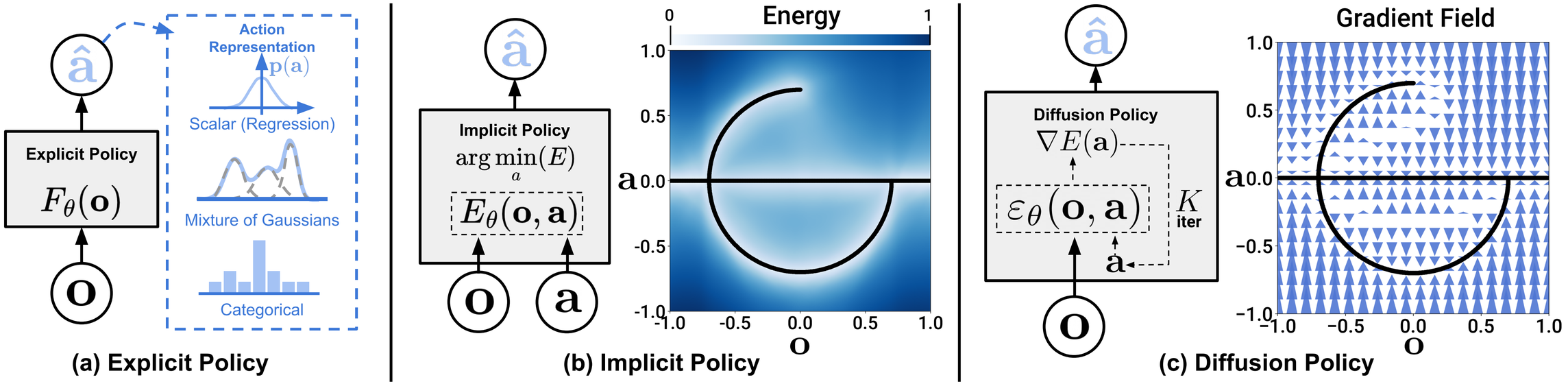
Generative model for robot actions, learned from demonstrations





Slides from Shuran Song

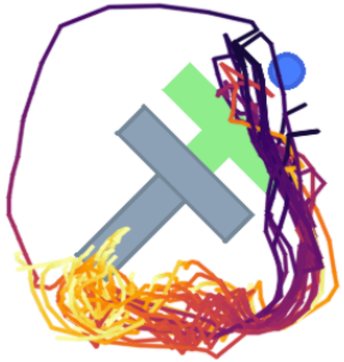
Visuomotor Policy Learning- Diffusion Policy



Slides from Shuran Song

Visuomotor Policy Learning- Diffusion Policy

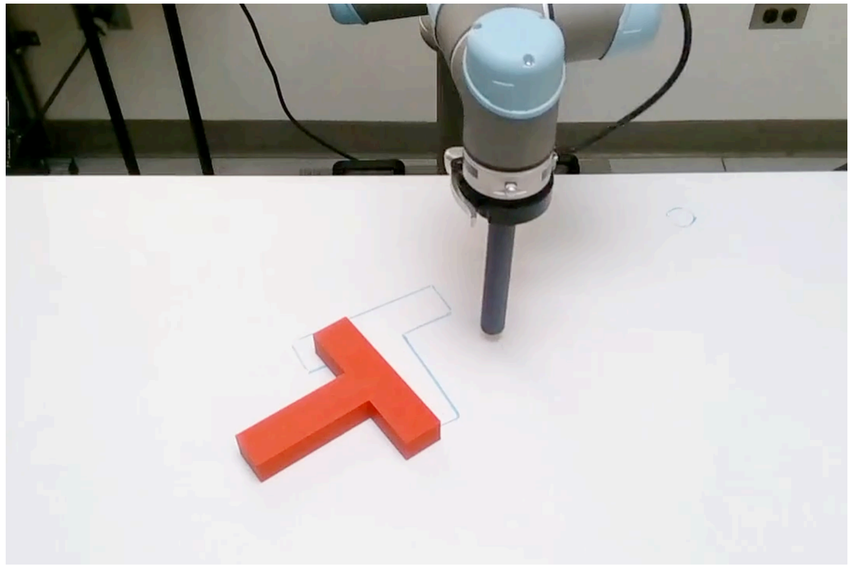
Diffusion Policy



LSTM-GMM

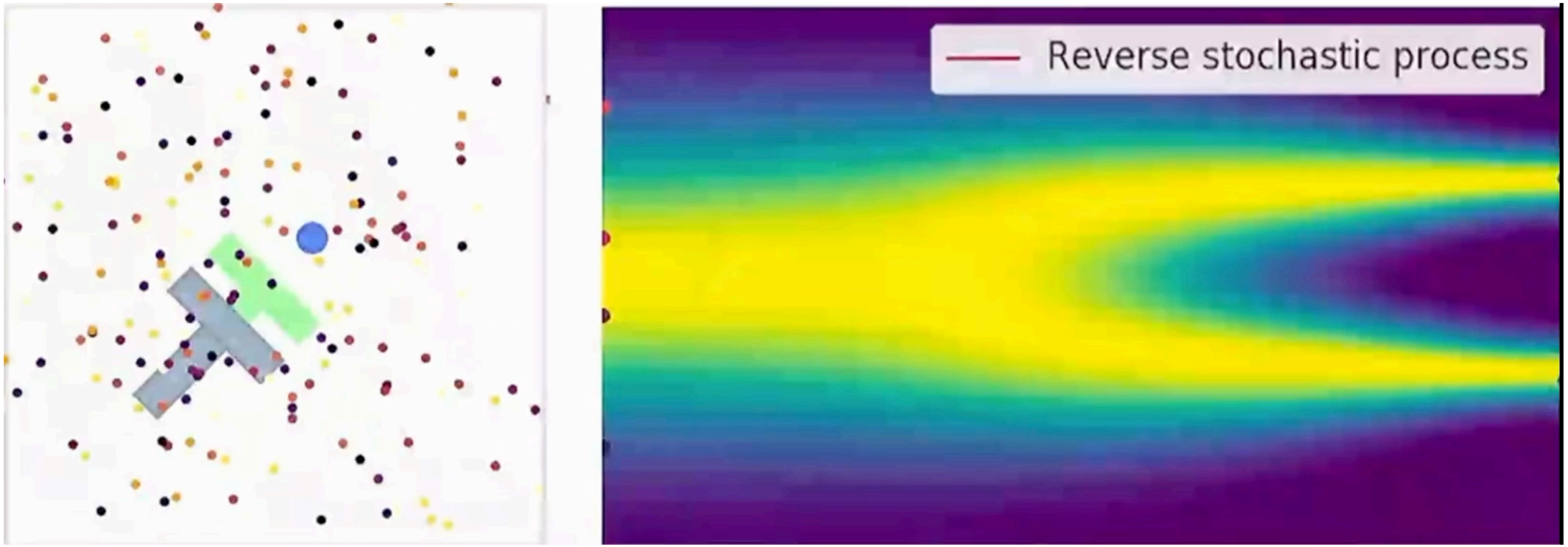
BET

IBC



Slides from Shuran Song

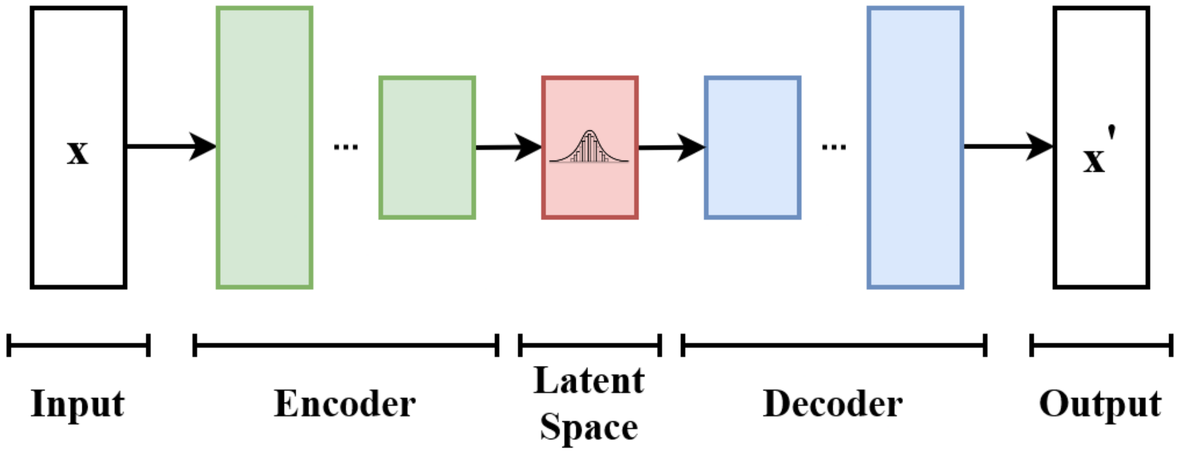
Visuomotor Policy Learning- Diffusion Policy



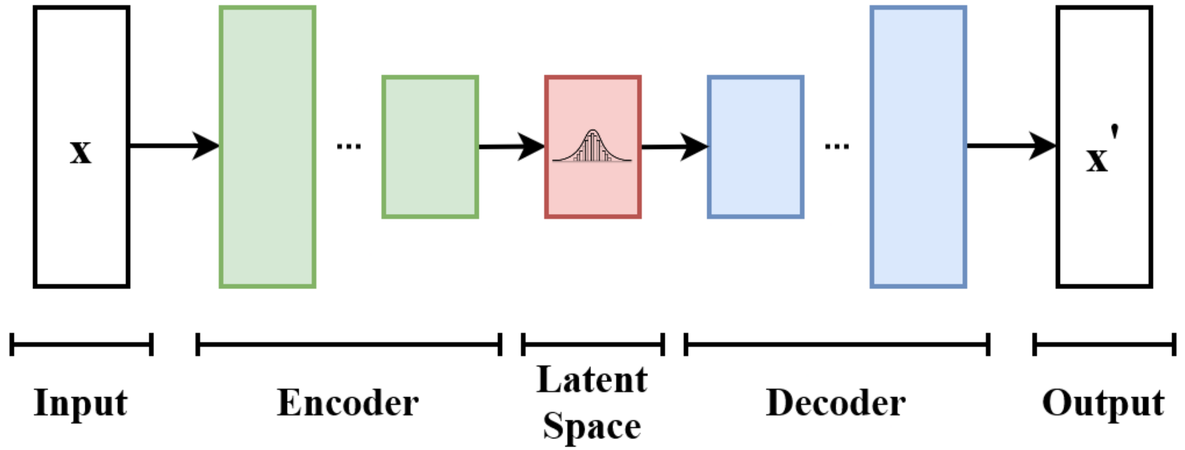
Predicted gradient field can have any number of local minimal, each captures a mode.

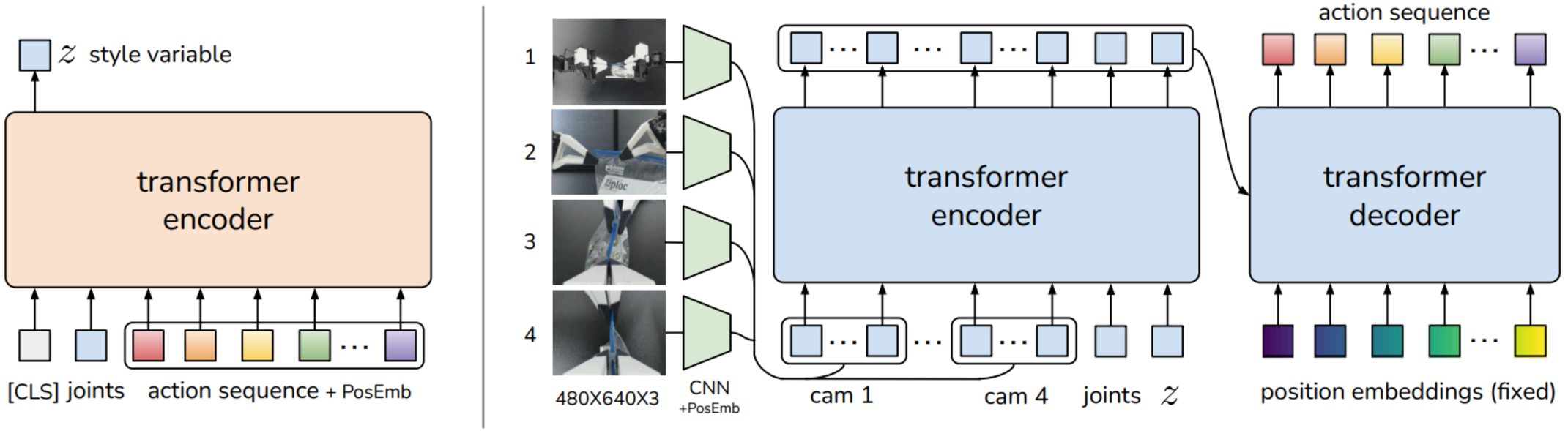
Slides from Shuran Song

Visuomotor Policy Learning- Action Chunking Transformer



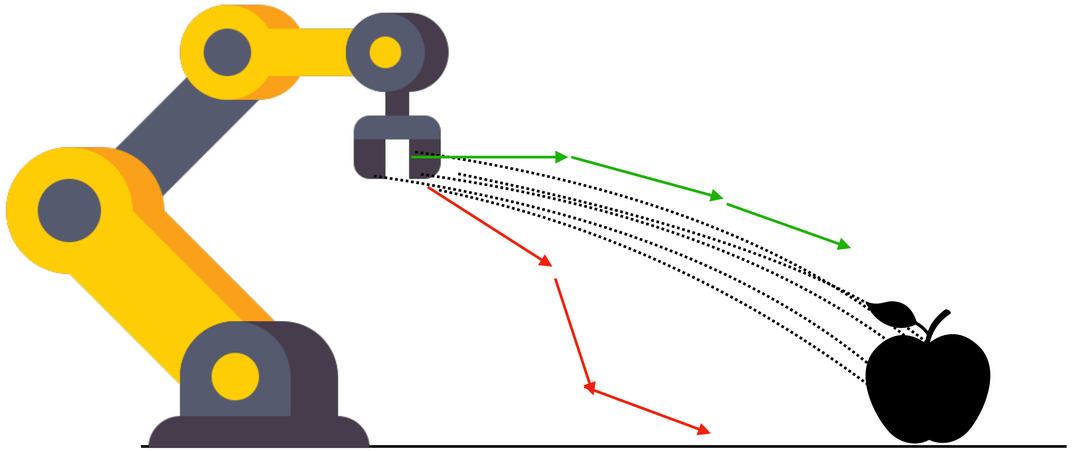
Visuomotor Policy Learning- Action Chunking Transformer



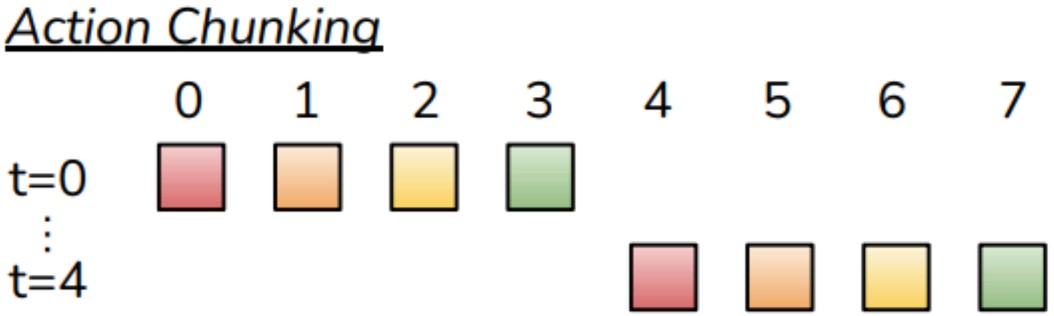


Visuomotor Policy Learning- Action Chunking Transformer

How does this fix covariate shift?

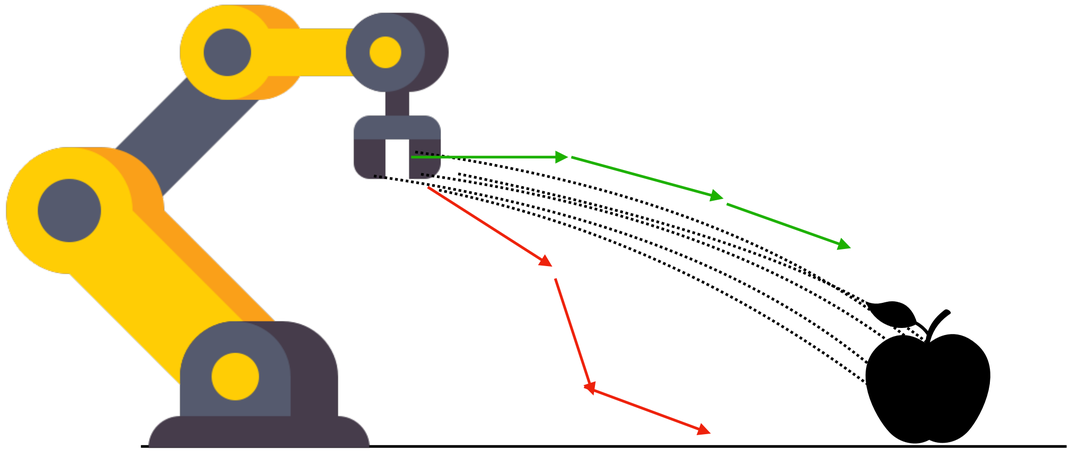


What if we don’t just predict 1 action every step?

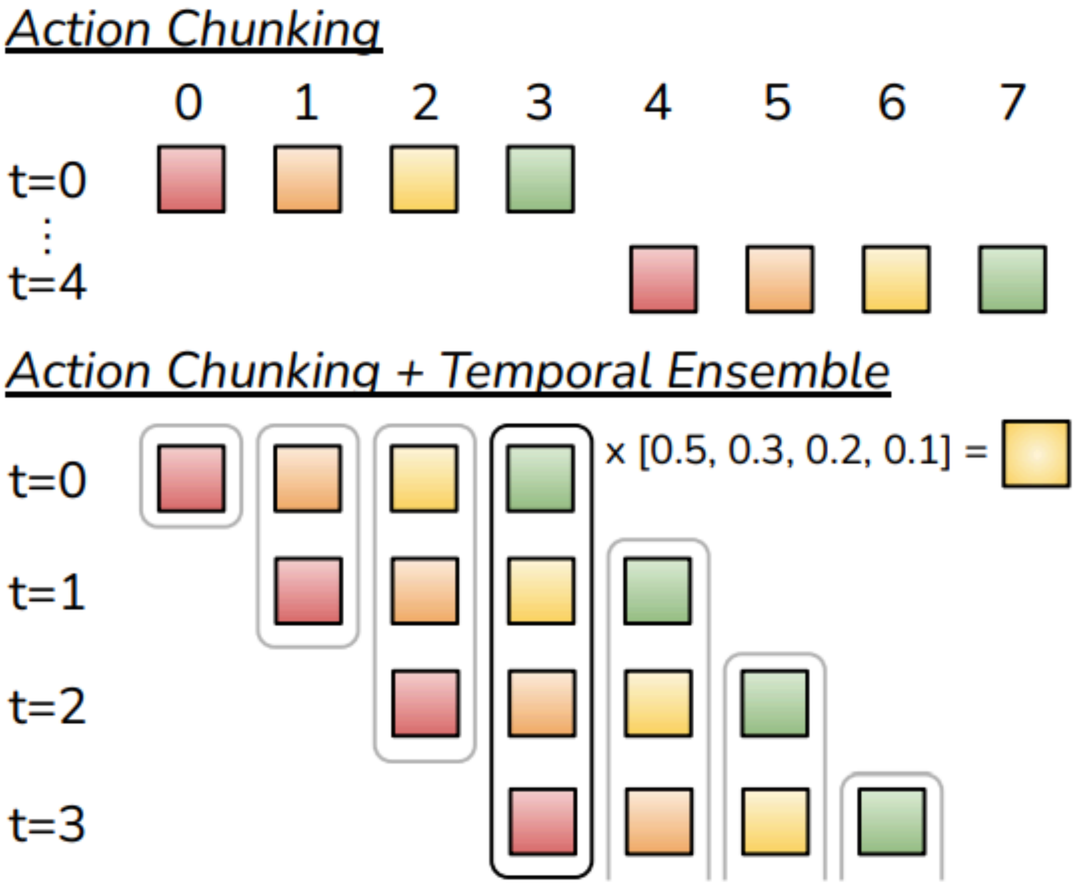


Visuomotor Policy Learning- Action Chunking Transformer

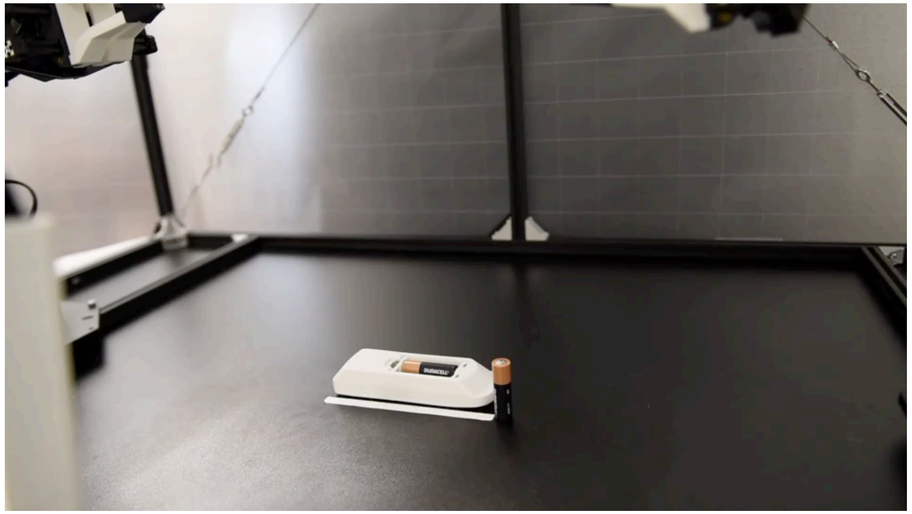
How do temporal ensemble?

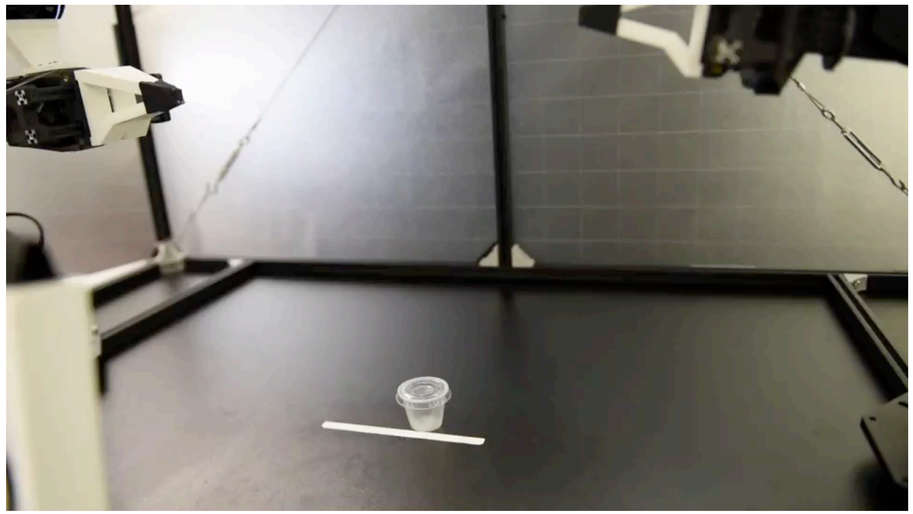


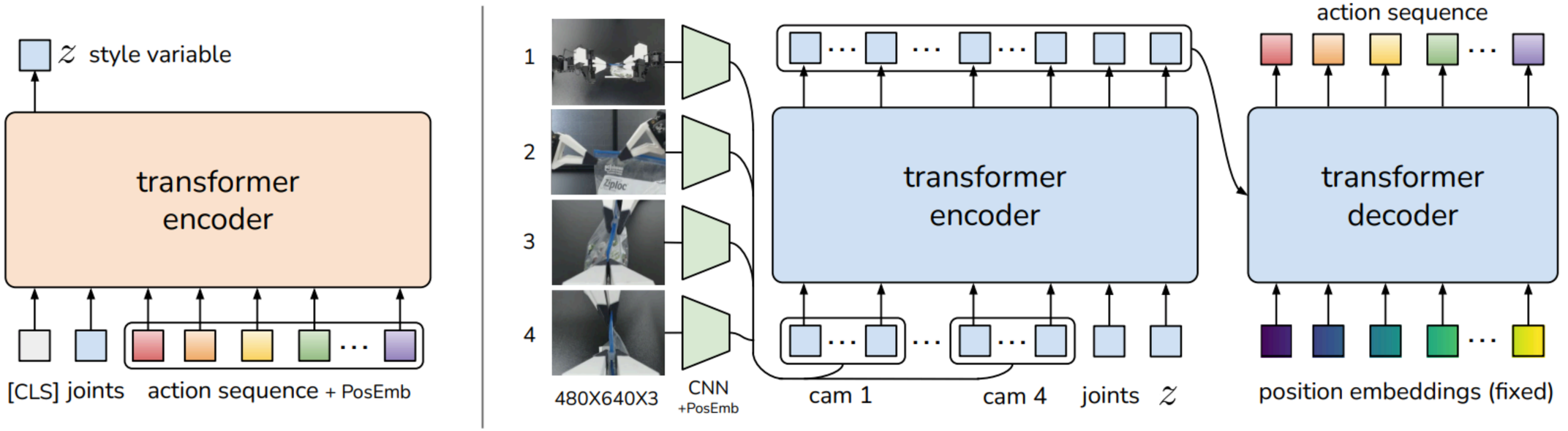
What if we don’t just predict 1 action every step?



Visuomotor Policy Learning- Action Chunking Transformer







The first idea of end-to-end visuomotor policy.

Journal of Machine Learning Research 17 (2016) 1-40 Submitted 10/15; Published 4/16

End-to-End Training of Deep Visuomotor Policies

Sergey Levine† SVLEVINE@EECS.BERKELEY.EDU
Chelsea Finn† CBFINN@EECS.BERKELEY.EDU
Trevor Darrell TREVOR@EECS.BERKELEY.EDU
Pieter Abbeel PABBEEL@EECS.BERKELEY.EDU
Division of Computer Science
University of California
Berkeley, CA 94720-1776, USA
†These authors contributed equally.

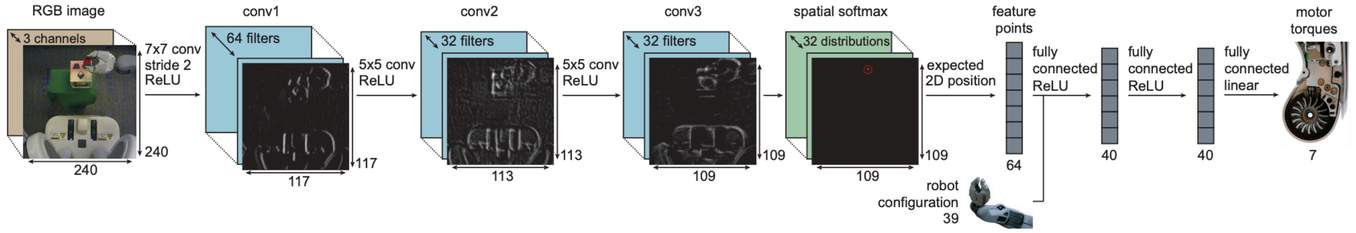
Editor: Jan Peters

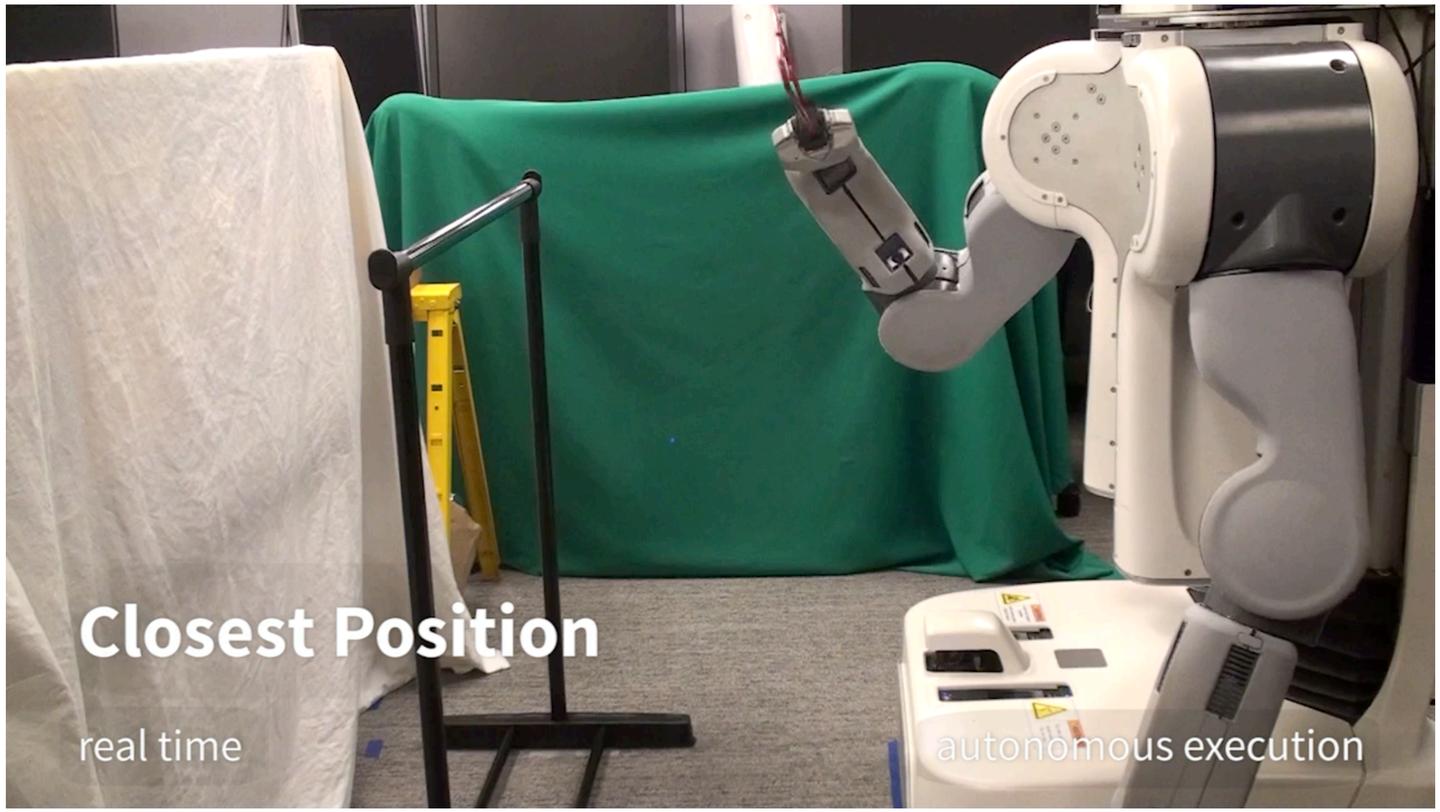
Abstract

Policy search methods can allow robots to learn control policies for a wide range of tasks, but practical applications of policy search often require hand-engineered components for perception, state estimation, and low-level control. In this paper, we aim to answer the following question: does training the perception and control systems jointly end-to-end provide better performance than training each component separately? To this end, we develop a method that can be used to learn policies that map raw image observations directly to torques at the robot's motors. The policies are represented by deep convolutional neural networks (CNNs) with 92,000 parameters, and are trained using a guided policy search method, which transforms policy search into supervised learning, with supervision provided by a simple trajectory-centric reinforcement learning method. We evaluate our method on a range of real-world manipulation tasks that require close coordination between vision and control, such as screwing a cap onto a bottle, and present simulated comparisons to a range of prior policy search methods.

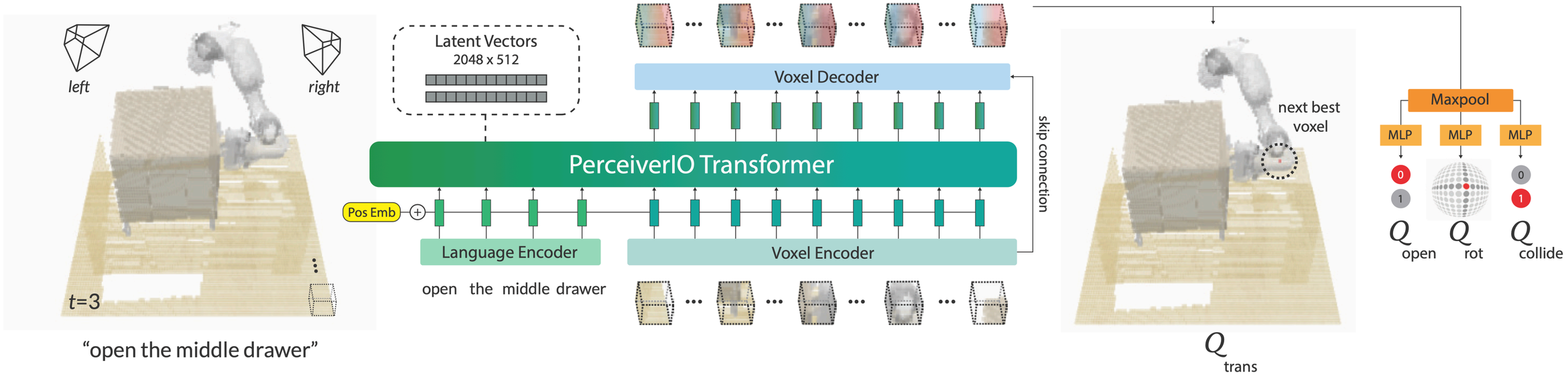
Keywords: Reinforcement Learning, Optimal Control, Vision, Neural Networks

92K Parameter

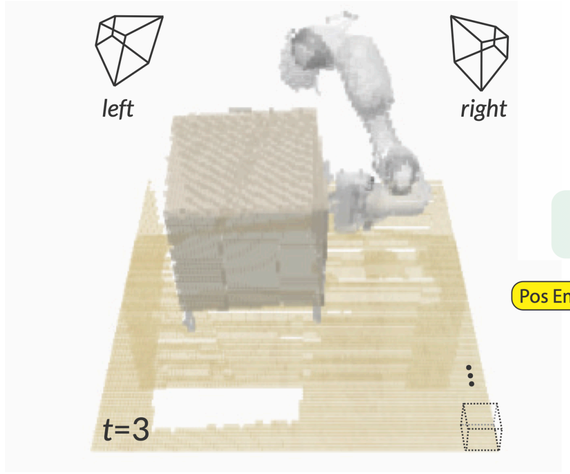




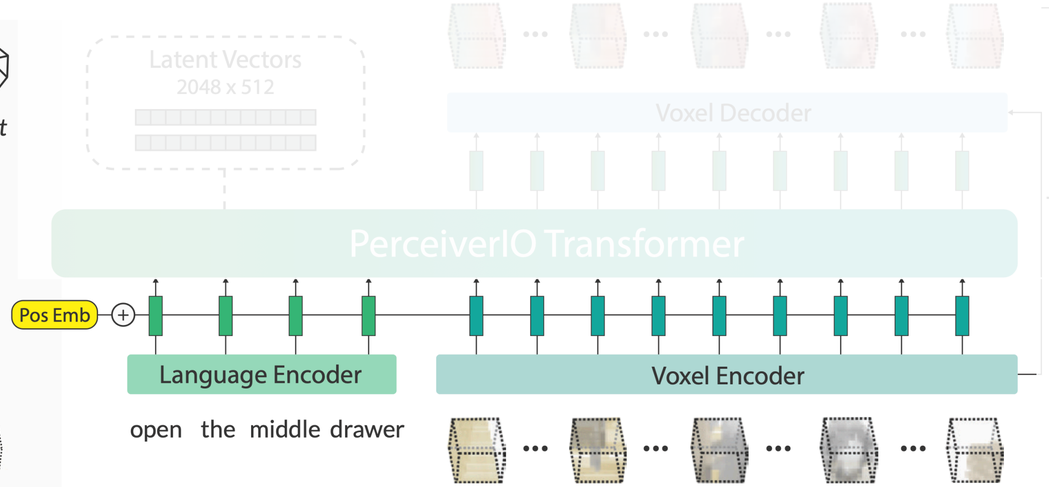
Representational learning- Perceiver-Actor

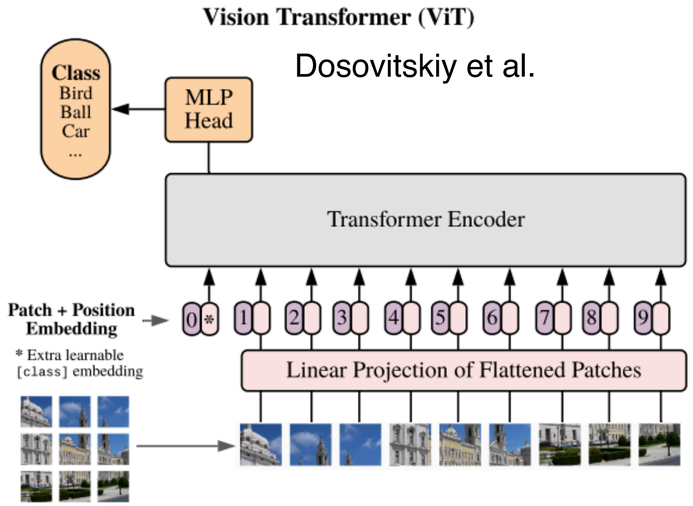


Perceiver-Actor



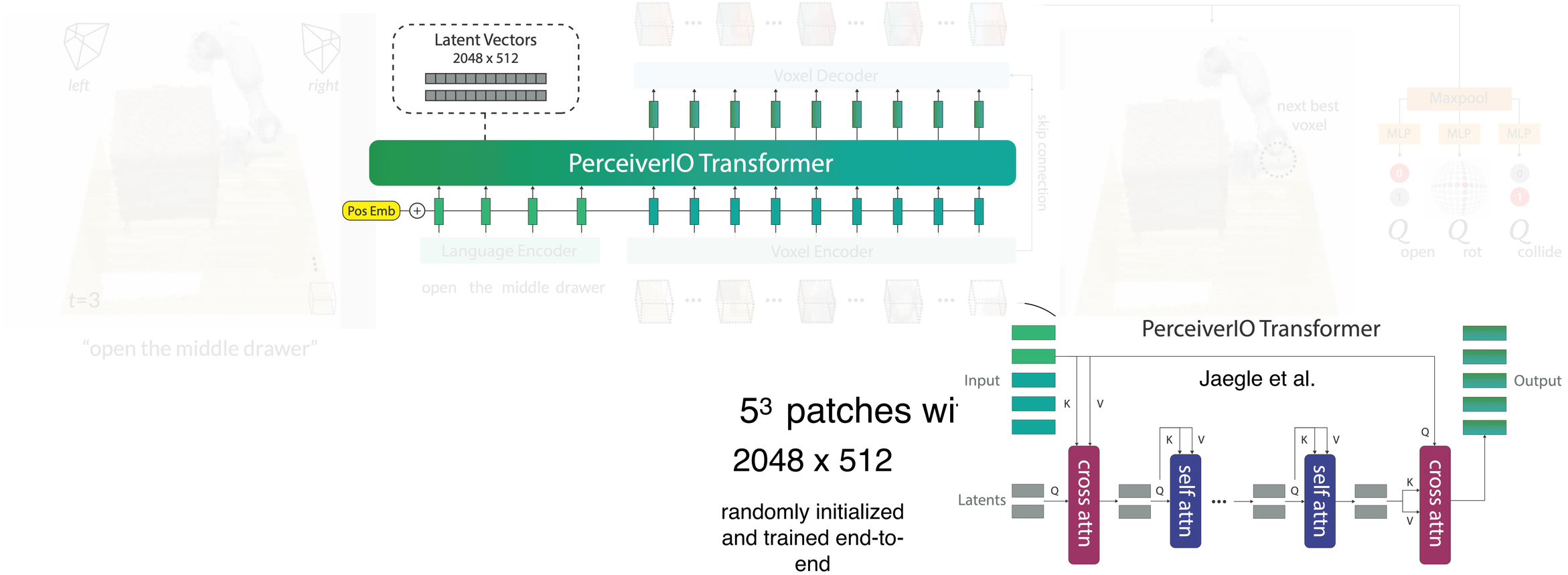
"open the middle drawer"



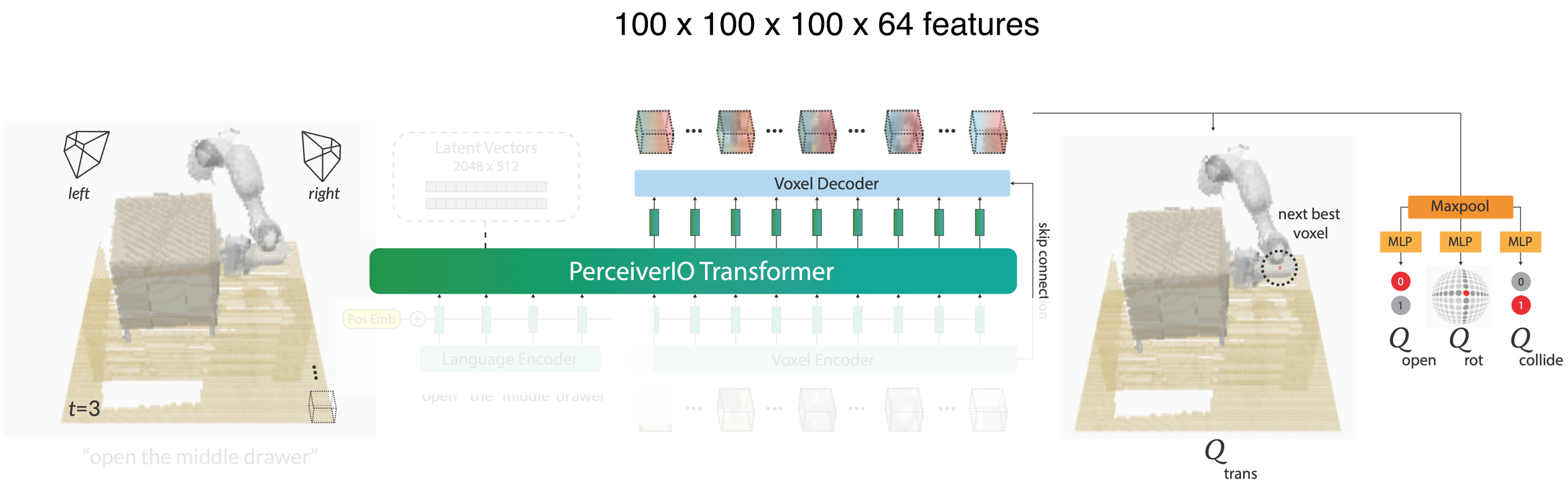


ViT : 2D image patches
PerAct : 3D voxel patches

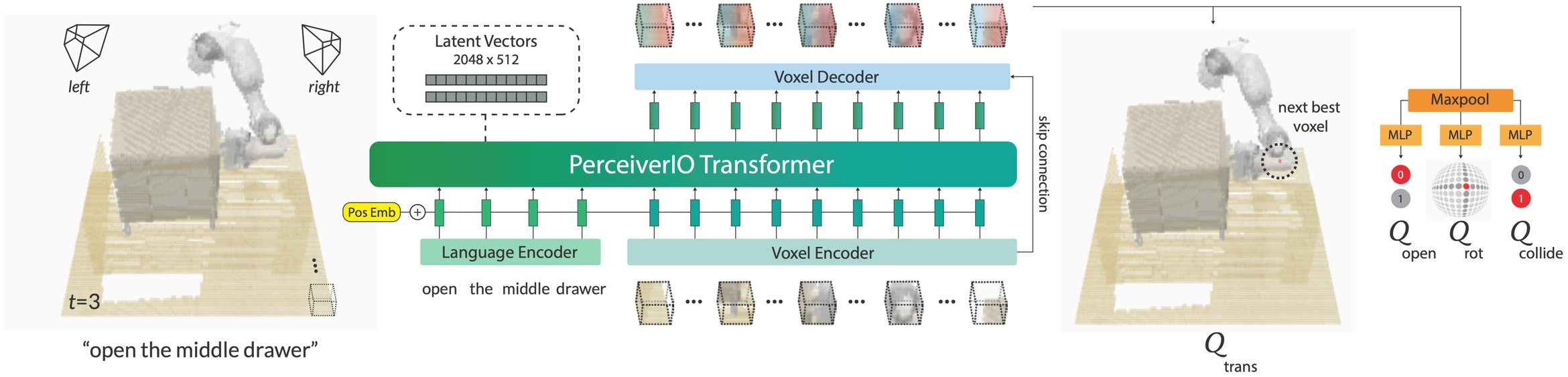
Perceiver-Actor



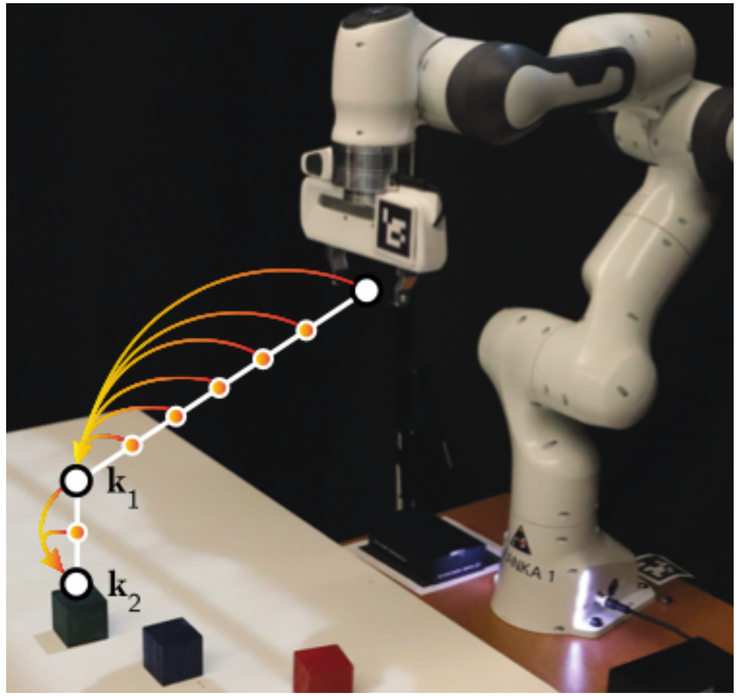
Perceiver-Actor



Perceiver-Actor



Dataset Setup



Predict the “next best **k**eyframe action” classification task

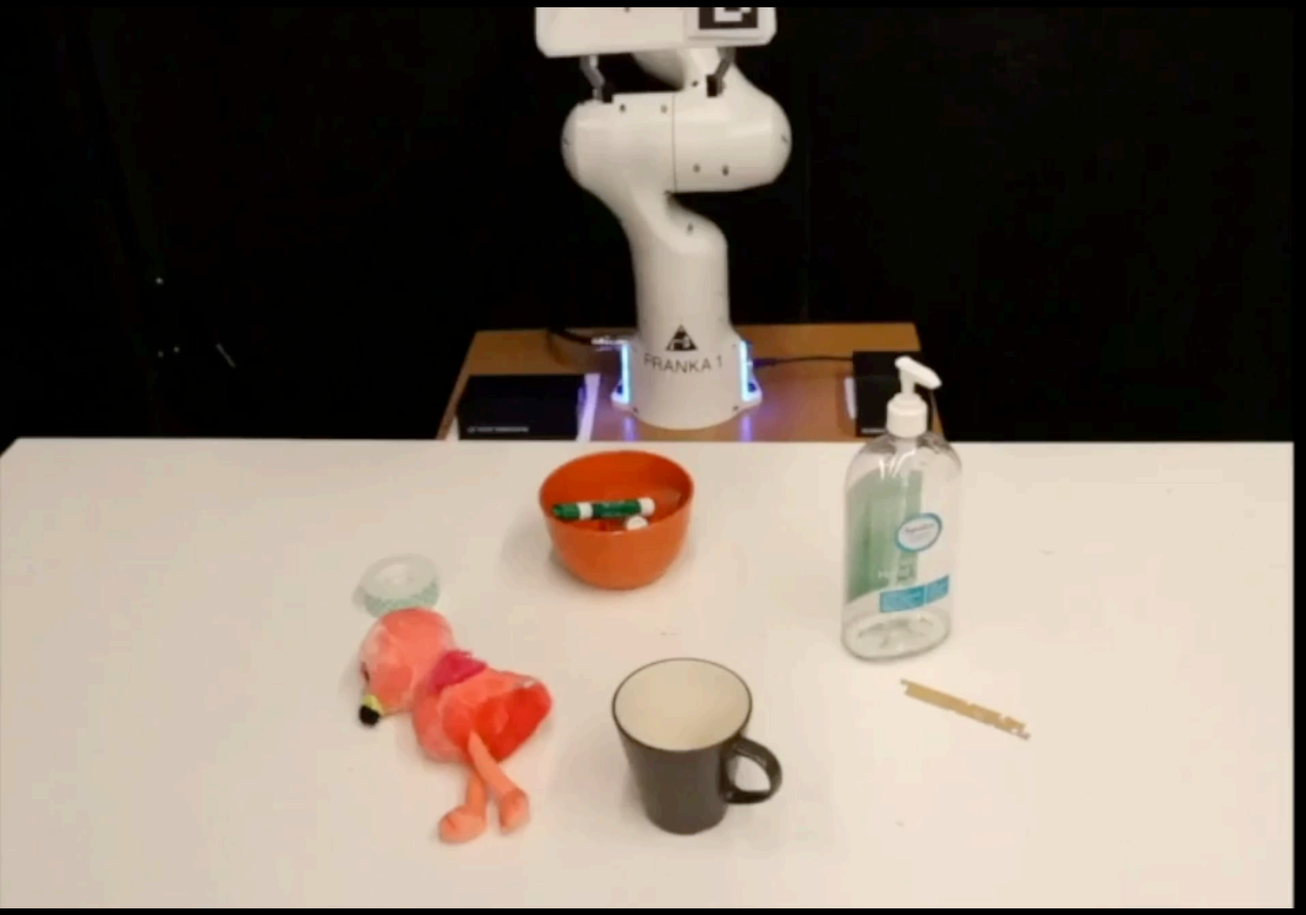
James et al

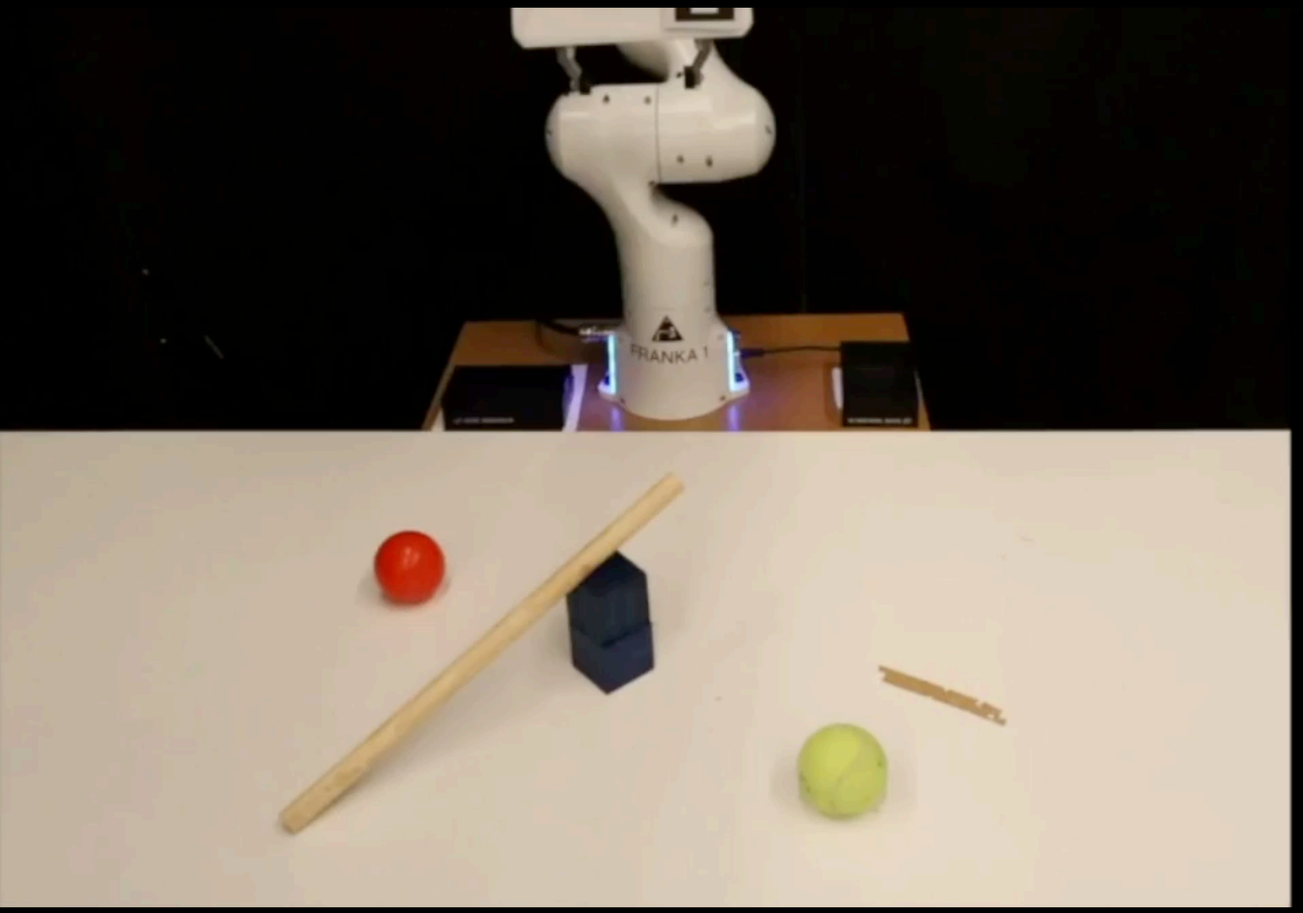
Heuristic for Keyframe Extraction:

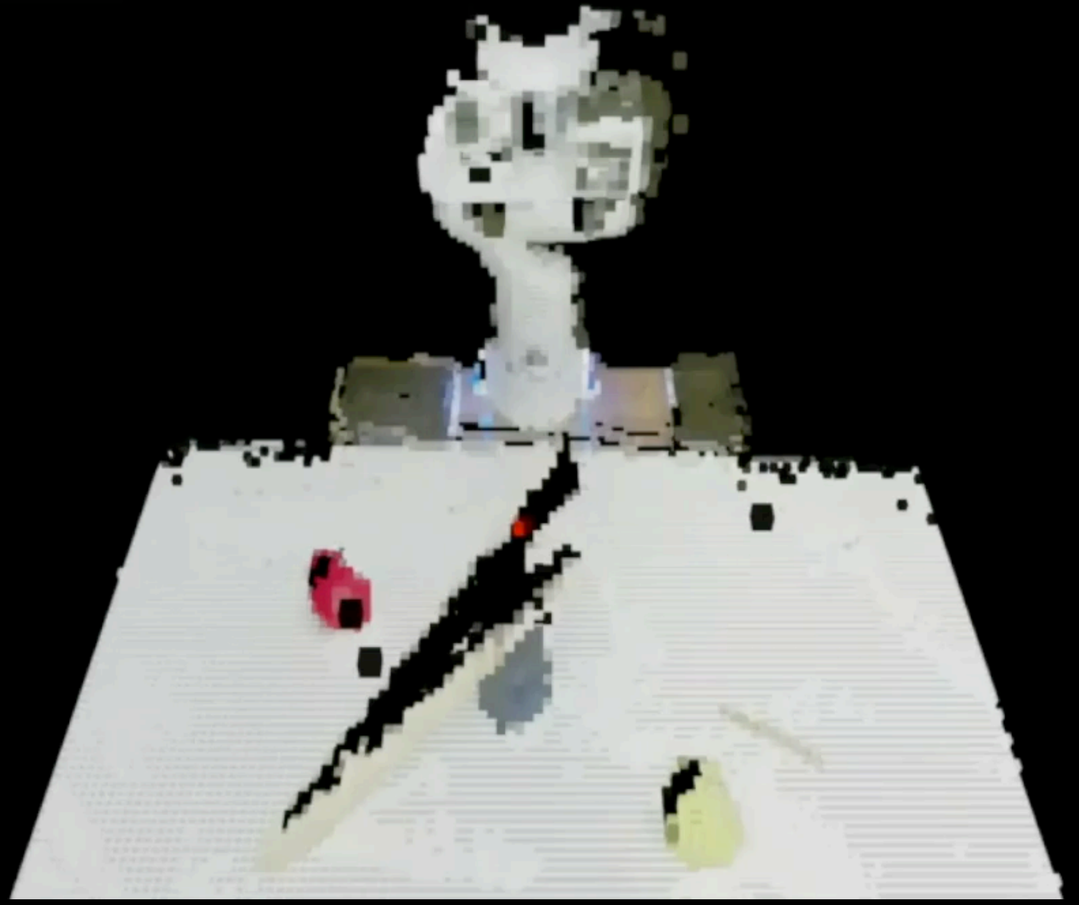
(1) Joint velocities are near zero &

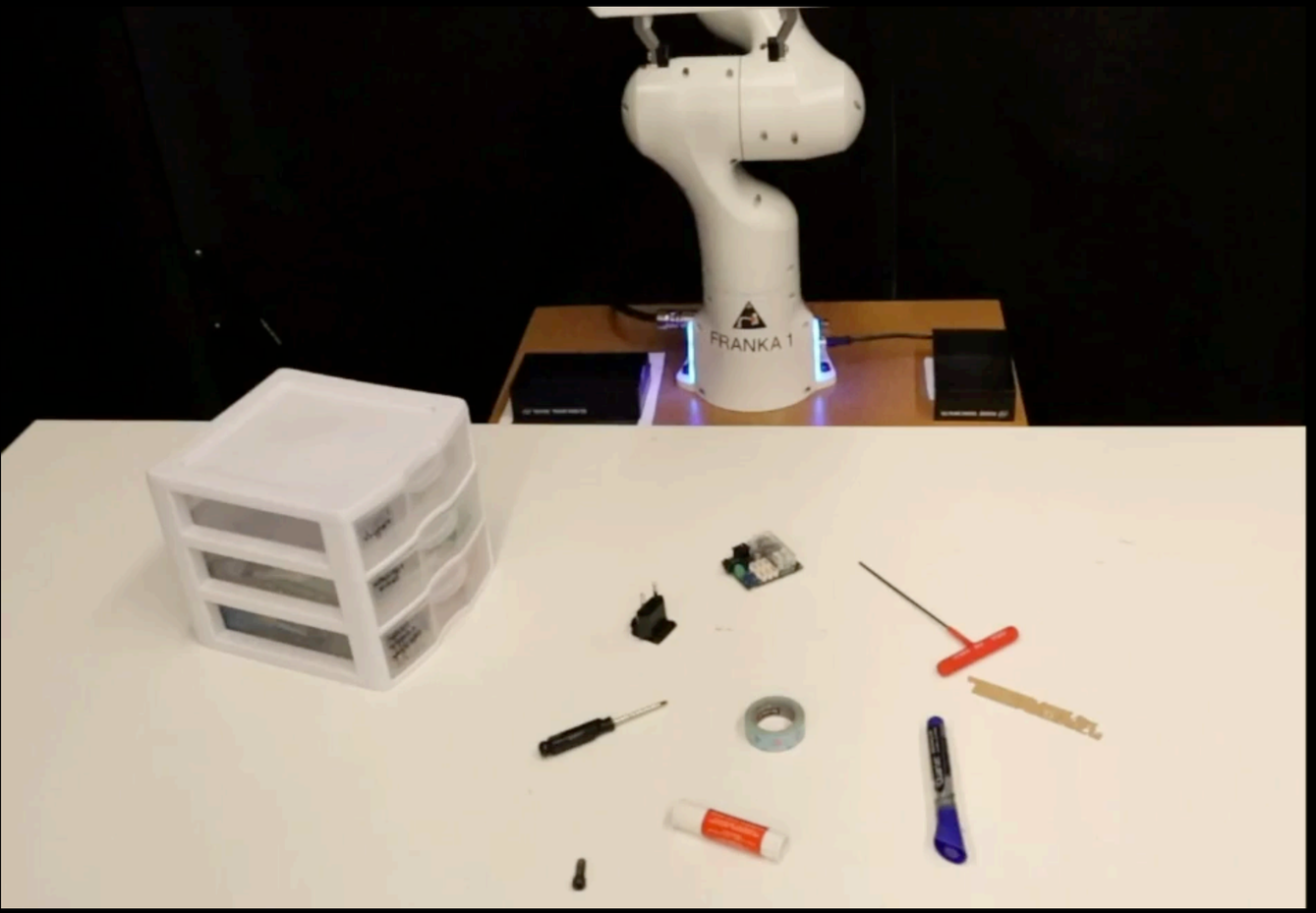
(2) Gripper open state has not changed

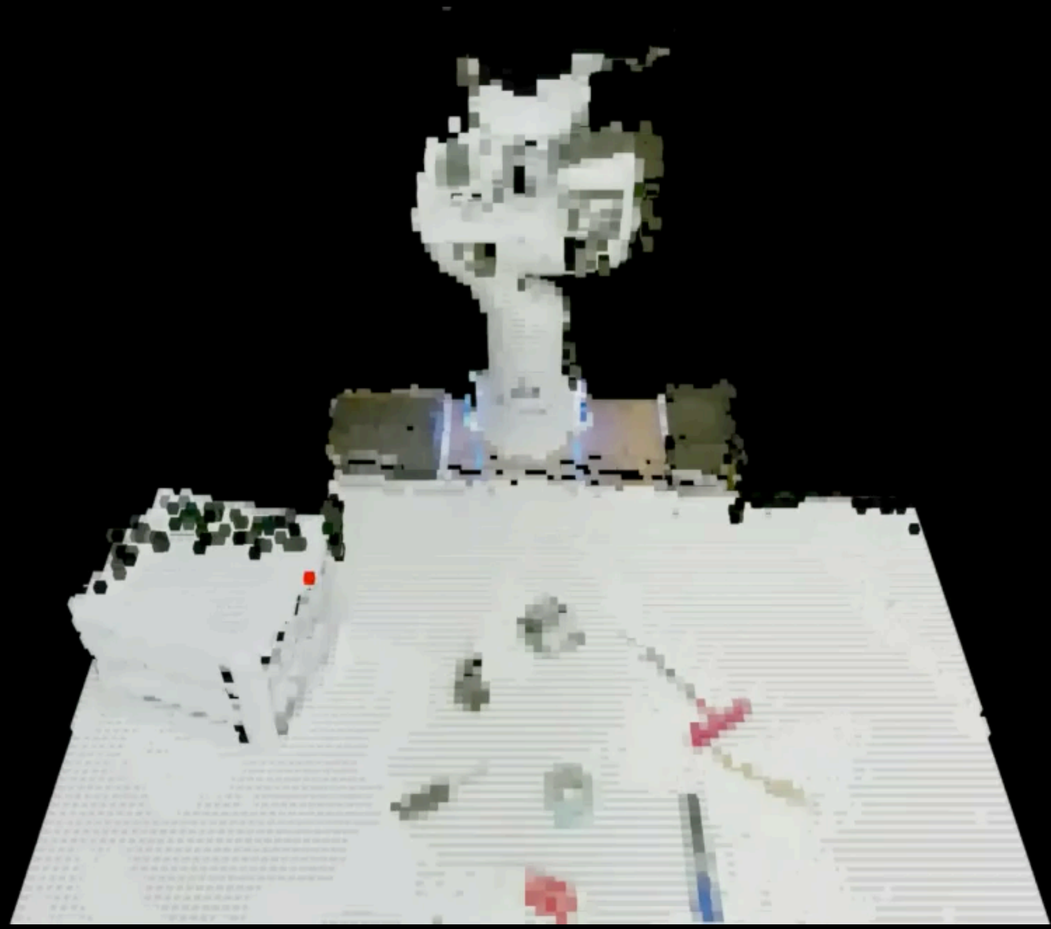

2x


FRANKA 1


FRANKA 1



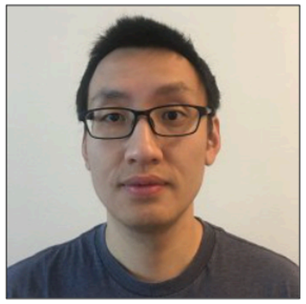

FRANKA 1

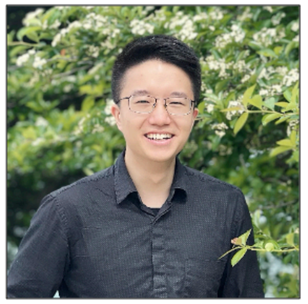


VoxPoser: Composable 3D Value Maps for Robotic Manipulation with Language Models


Wenlong Huang


Chen Wang


Ruohan Zhang


Yunzhu Li


Jiajun Wu

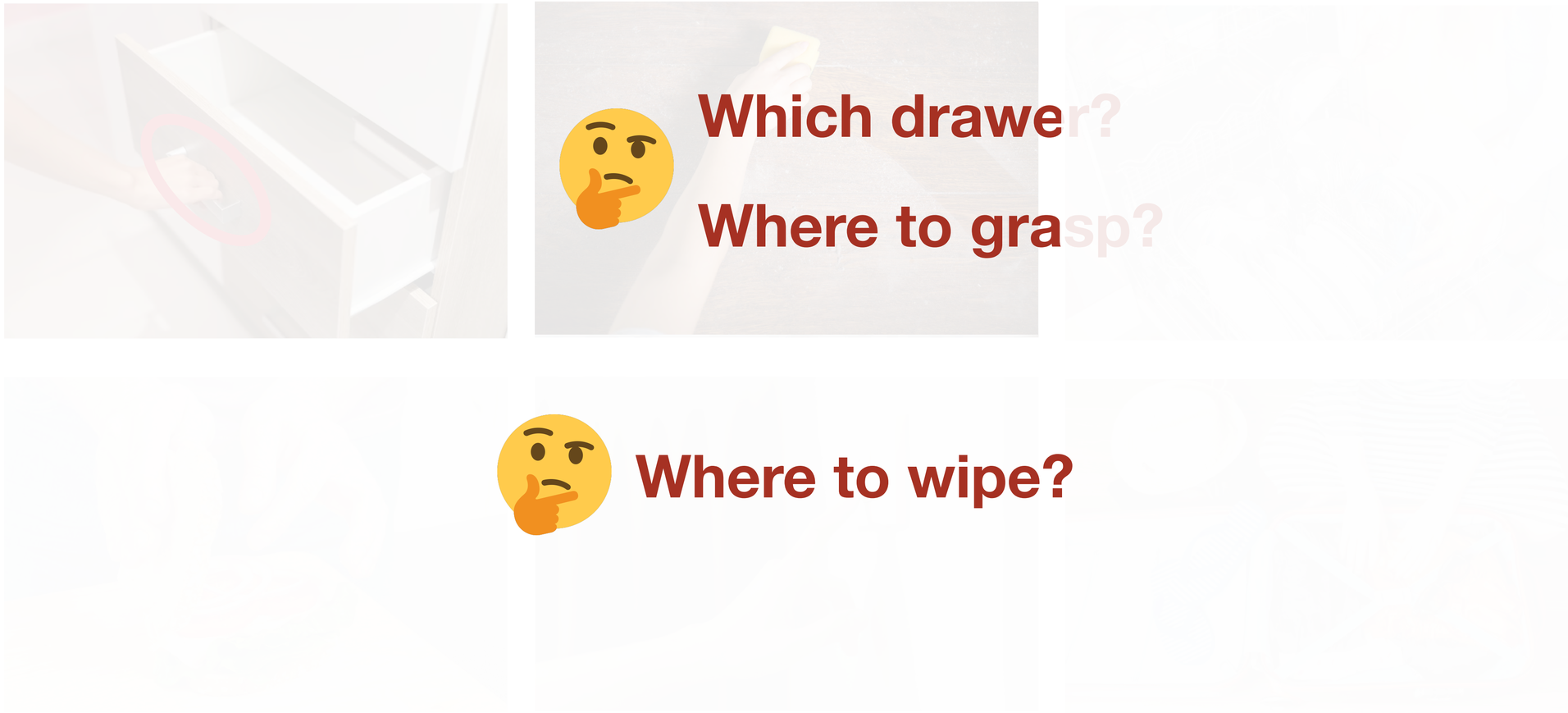

Li Fei-Fei

Stanford Artificial Intelligence

STANFORD VISION & LEARNING

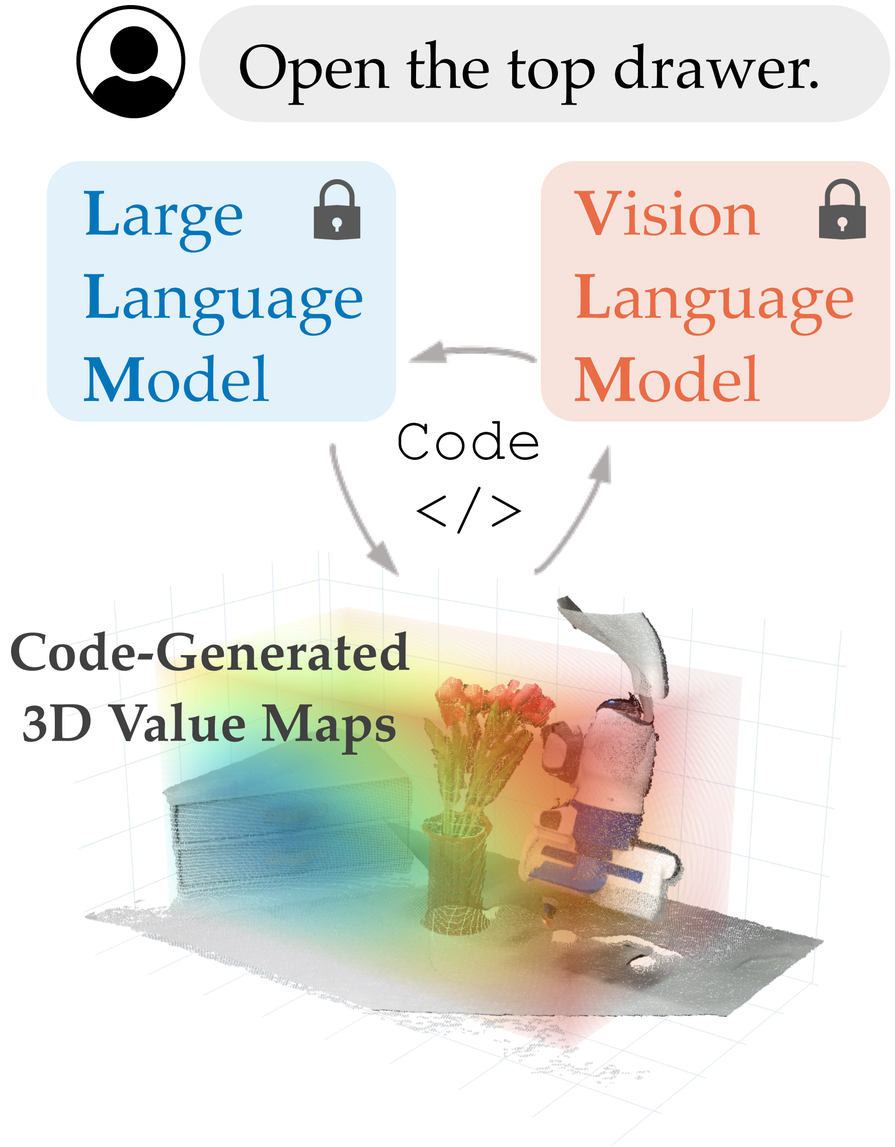
UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

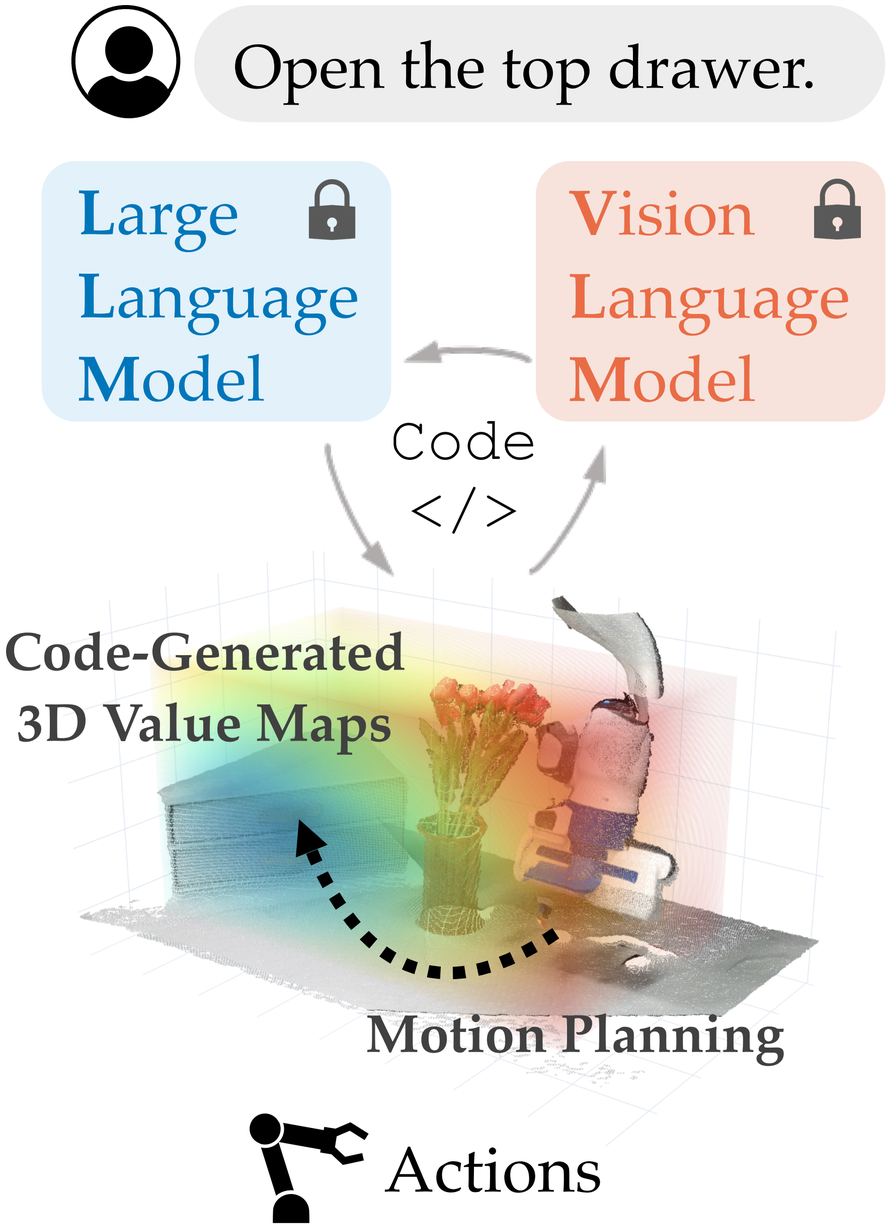
Spatial Task Representation for Manipulation



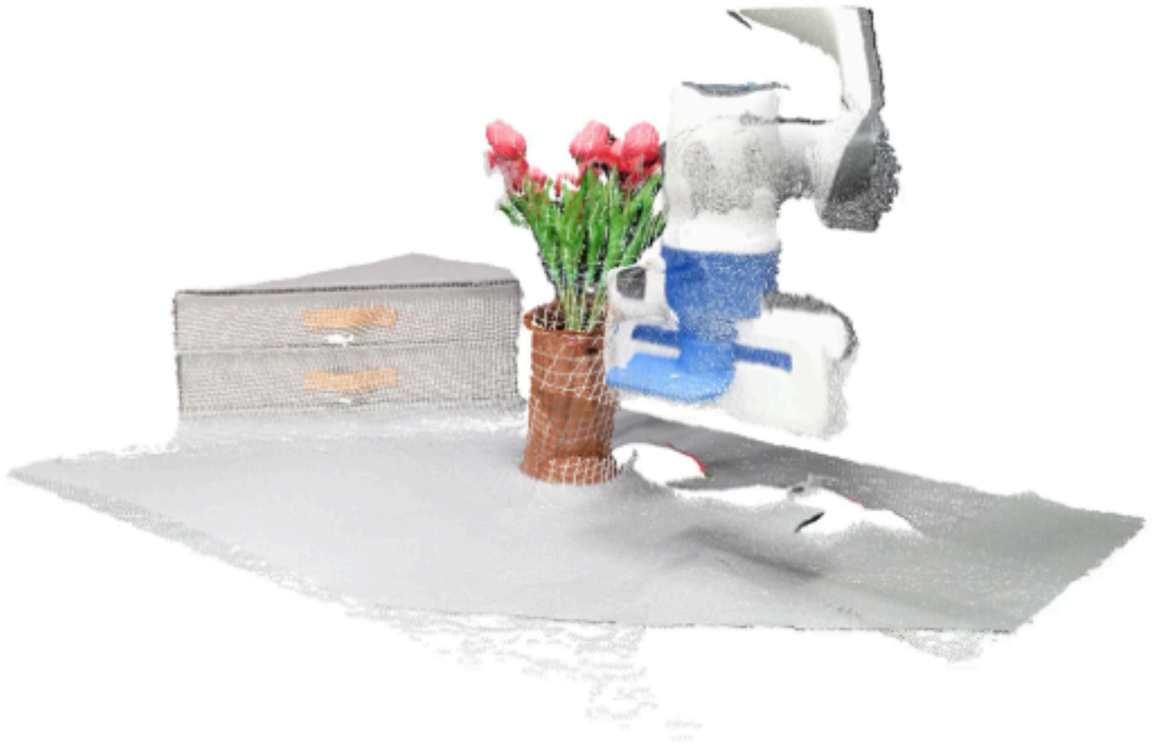
Can we extract this from existing foundation models?

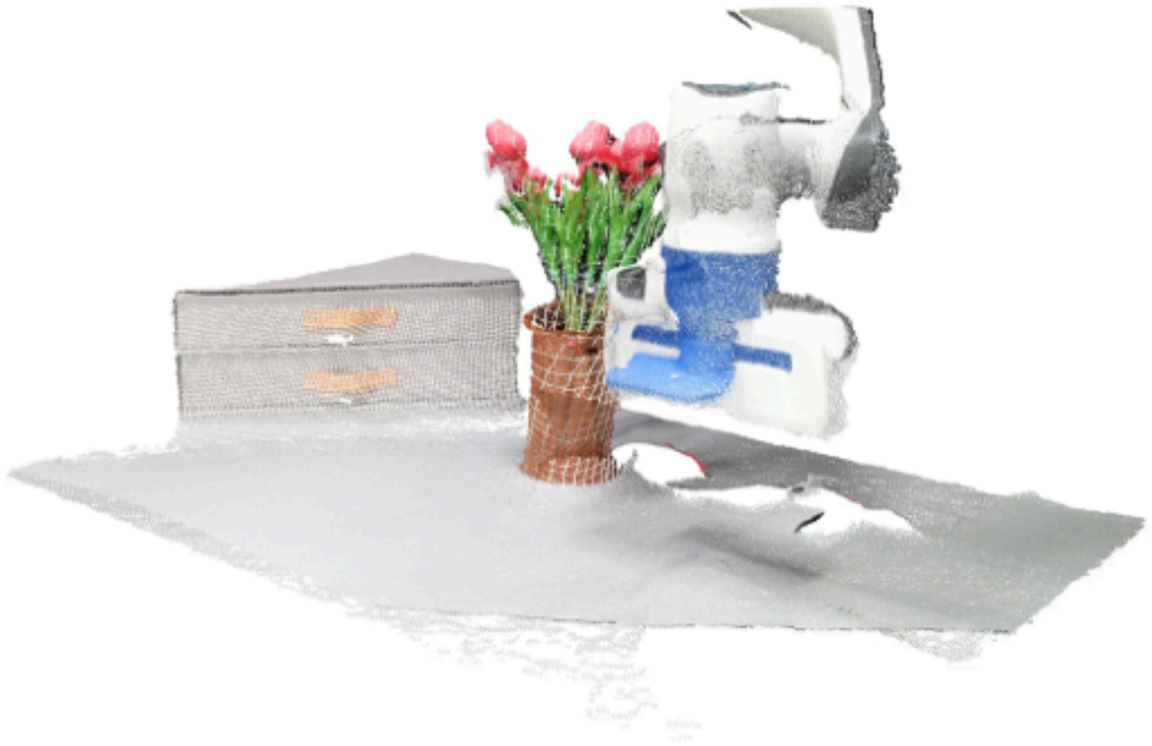
Open the top drawer.


Open the top drawer.
Large Language Model
Vision Language Model
Code
</>
Code-Generated 3D Value Maps


Open the top drawer.
Large Language Model
Vision Language Model
Code
</>
Code-Generated 3D Value Maps
Motion Planning
Actions

Open the top drawer.





Open the top drawer.

LLM Output

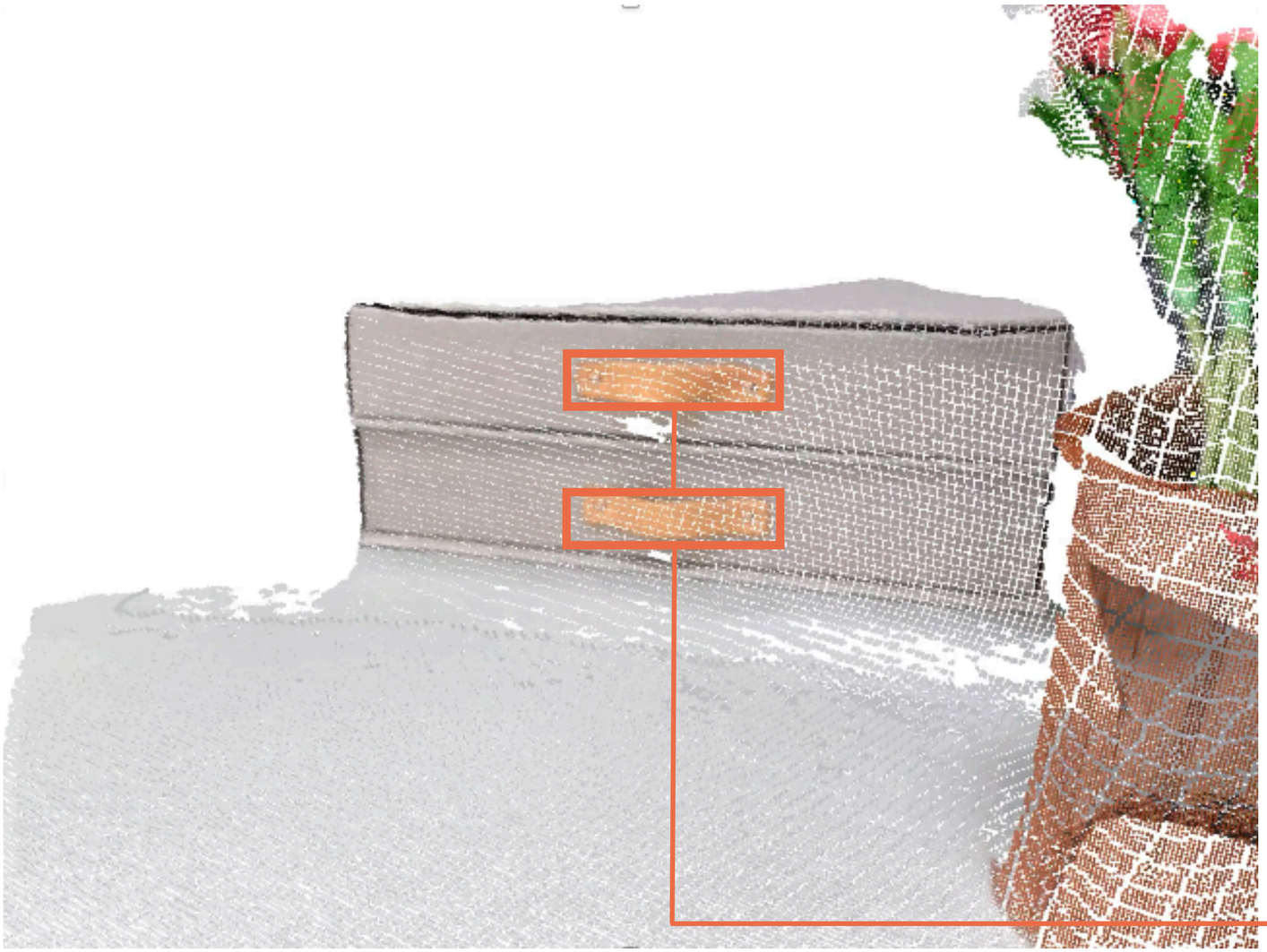
```
msize = (100,100,100)
map = np.zeros(msize)
```

Open the top drawer.

LLM Output

```
msize = (100,100,100)
map = np.zeros(msize)
handles = detect('handle')
```

Vision Language Model



Open the top drawer.

LLM Output

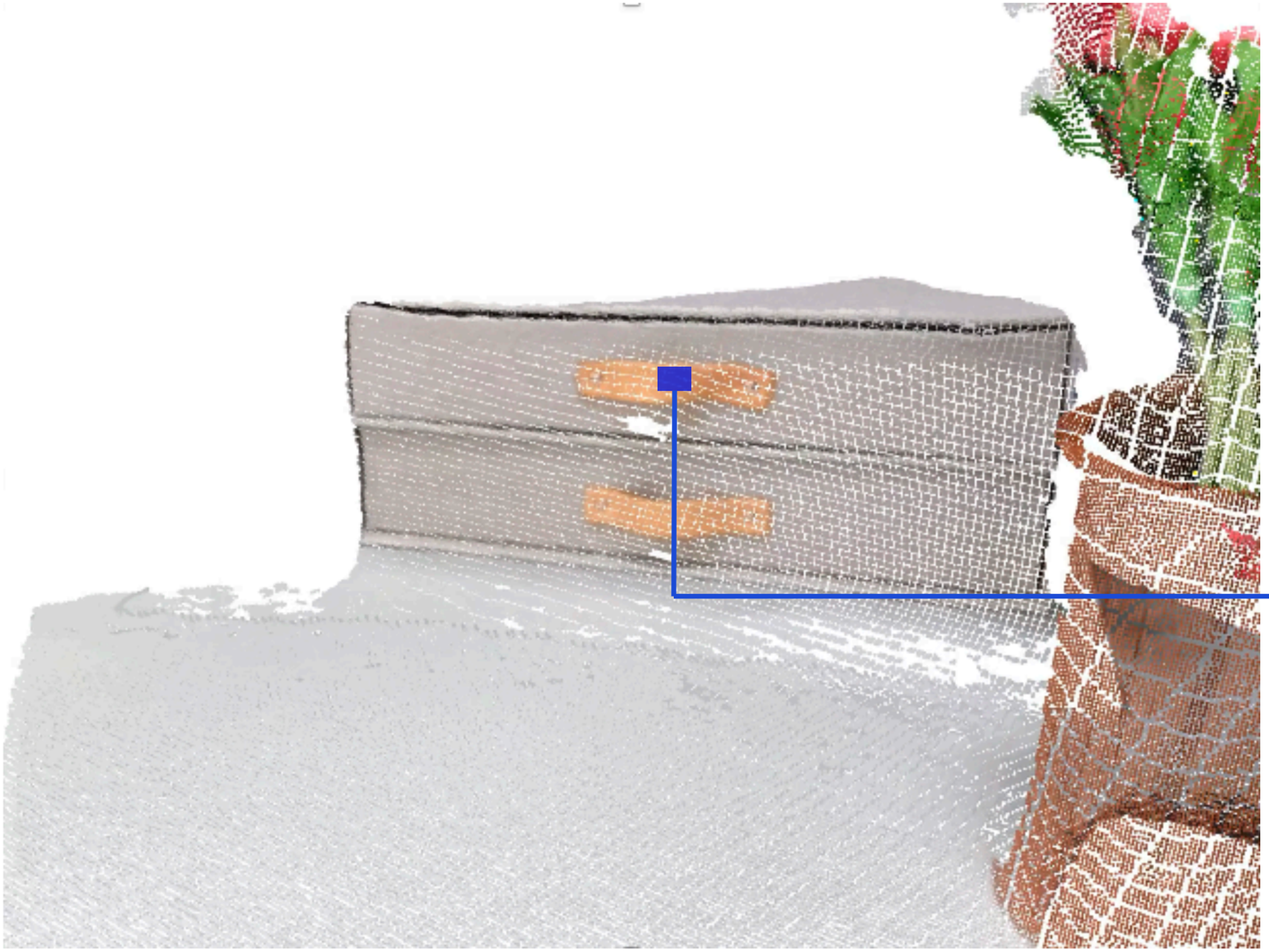
```
msize = (100,100,100)
map = np.zeros(msize)
handles = detect('handle')
```

Vision Language Model

Open the top drawer.

LLM Output

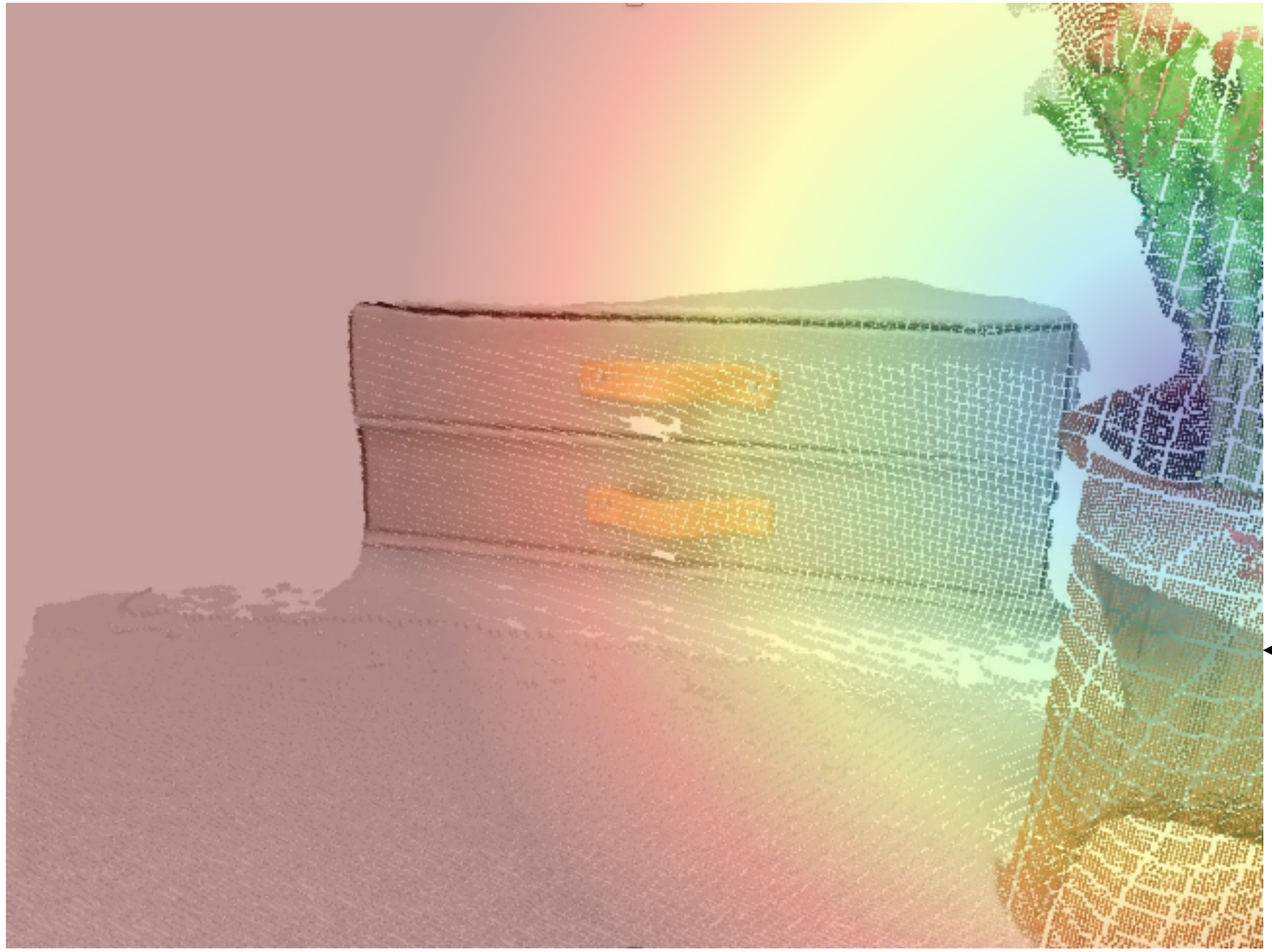
```
msize = (100,100,100)
map = np.zeros(msize)
handles = detect('handle')
k = lambda x: x.pos[2]
handles.sort(key=k)
top_handle = handles[-1]
```



Open the top drawer.

LLM Output

```
msize = (100,100,100)
map = np.zeros(msize)
handles = detect('handle')
k = lambda x: x.pos[2]
handles.sort(key=k)
top_handle = handles[-1]
x,y,z = top_handle.pos
map[x,y,z] = 1
```



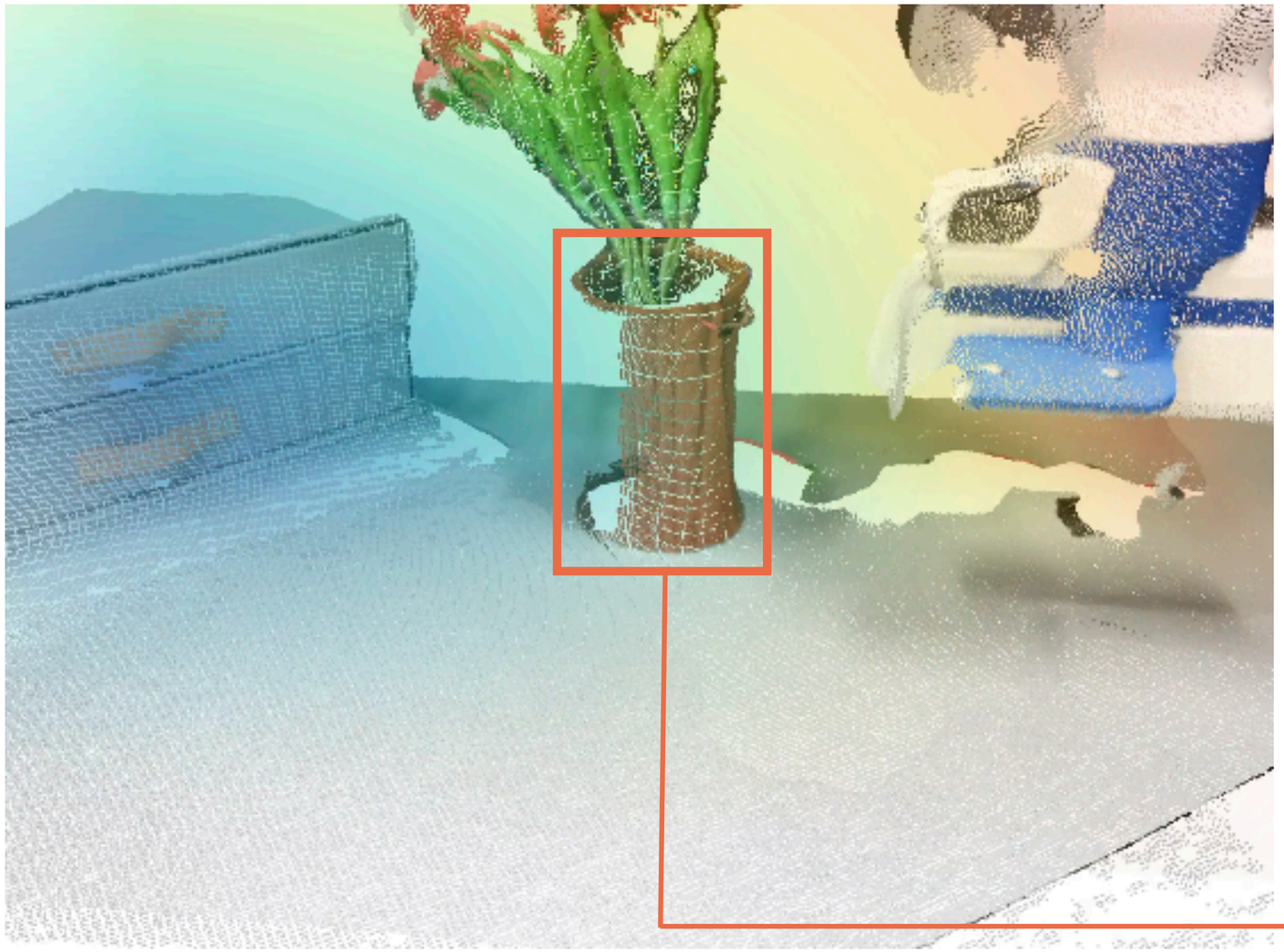
Open the top drawer.

LLM Output

```
msize = (100,100,100)
map = np.zeros(msize)
handles = detect('handle')
k = lambda x: x.pos[2]
handles.sort(key=k)
top_handle = handles[-1]
x,y,z = top_handle.pos
map[x,y,z] = 1
map = smooth(map)
```

Also watch out for that vase.

LLM Output

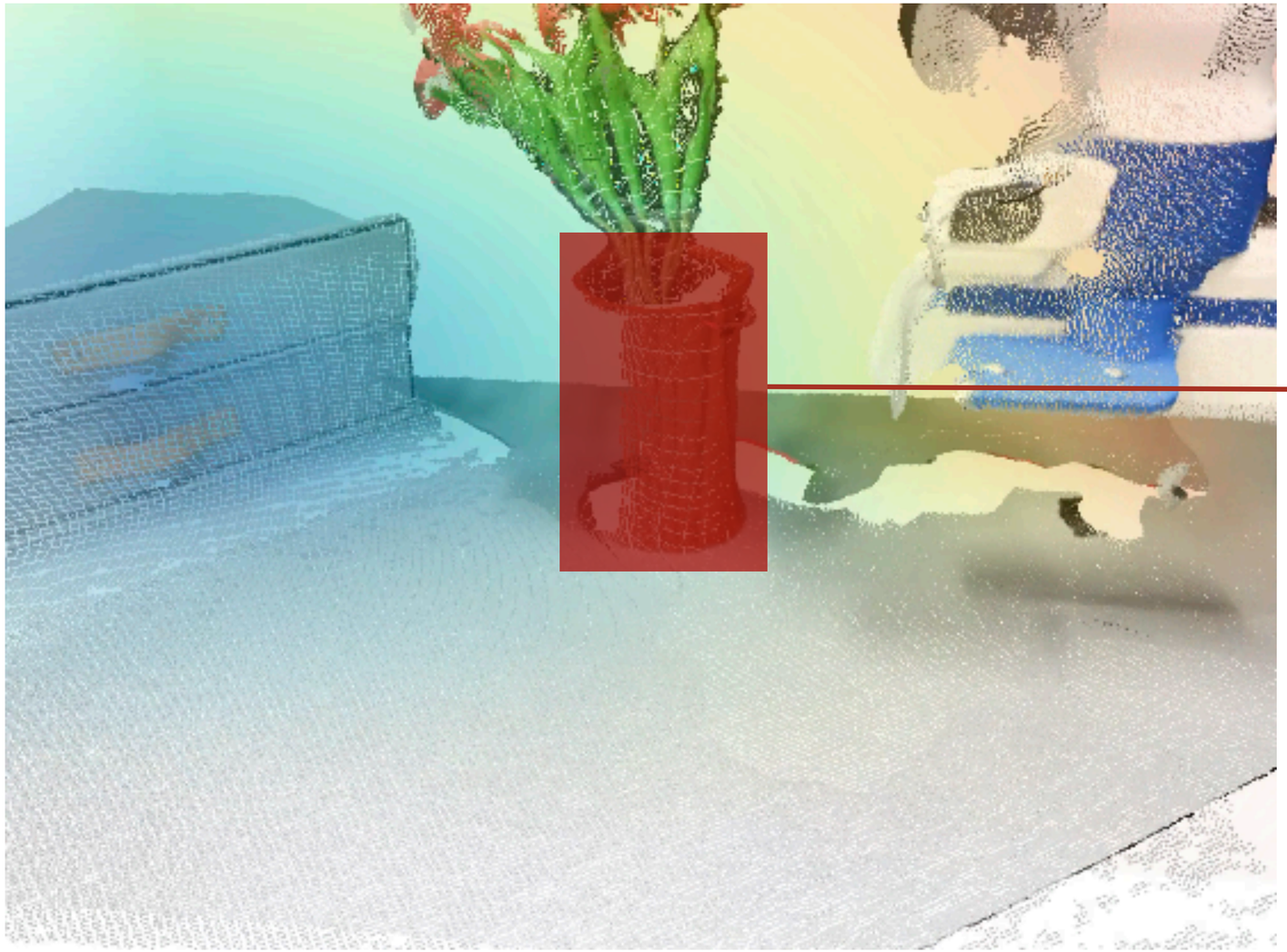


Also watch out for that vase.

LLM Output

```
vases = detect('vase')
```

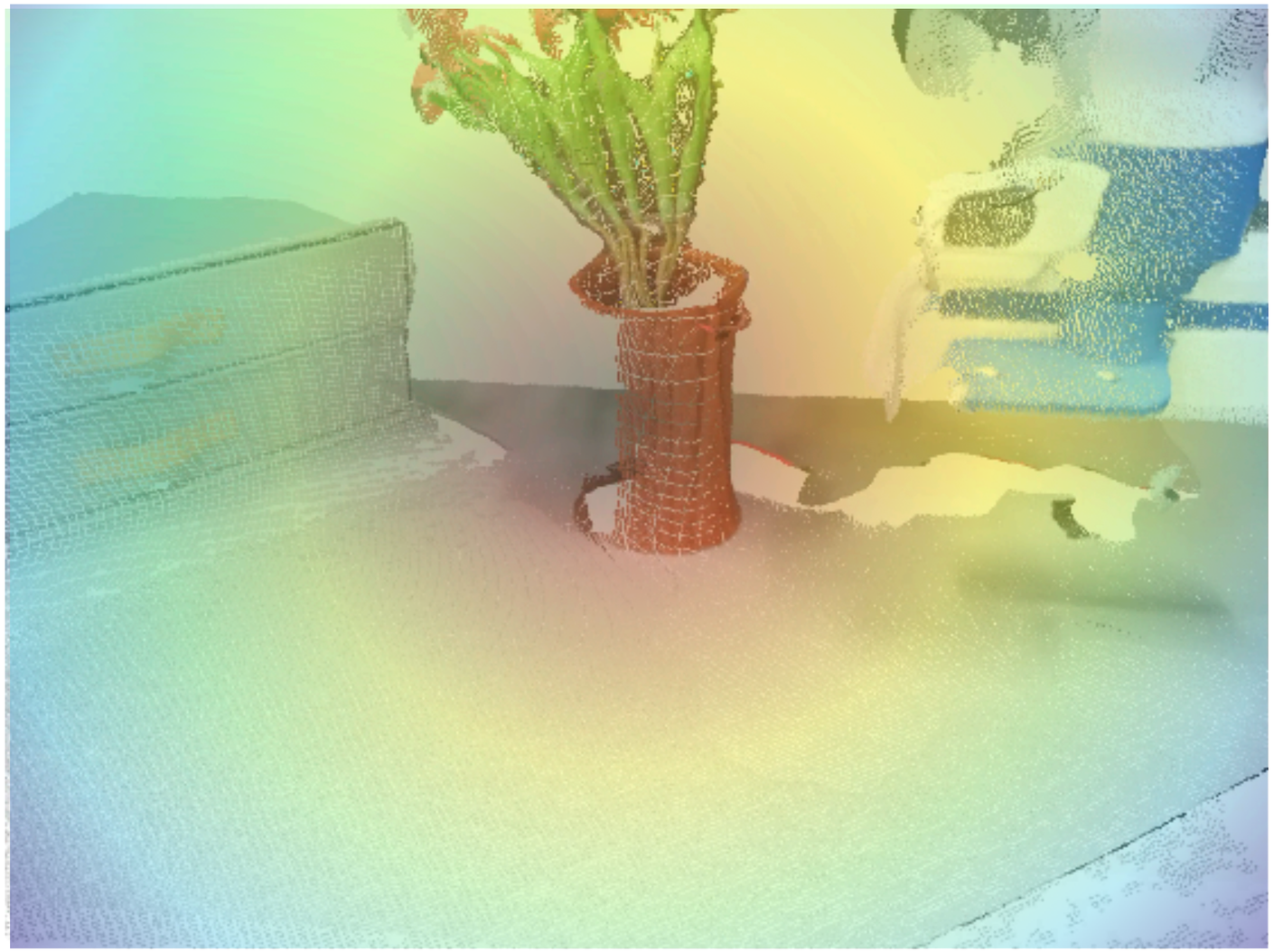
Vision Language Model



Also watch out for that vase.

LLM Output

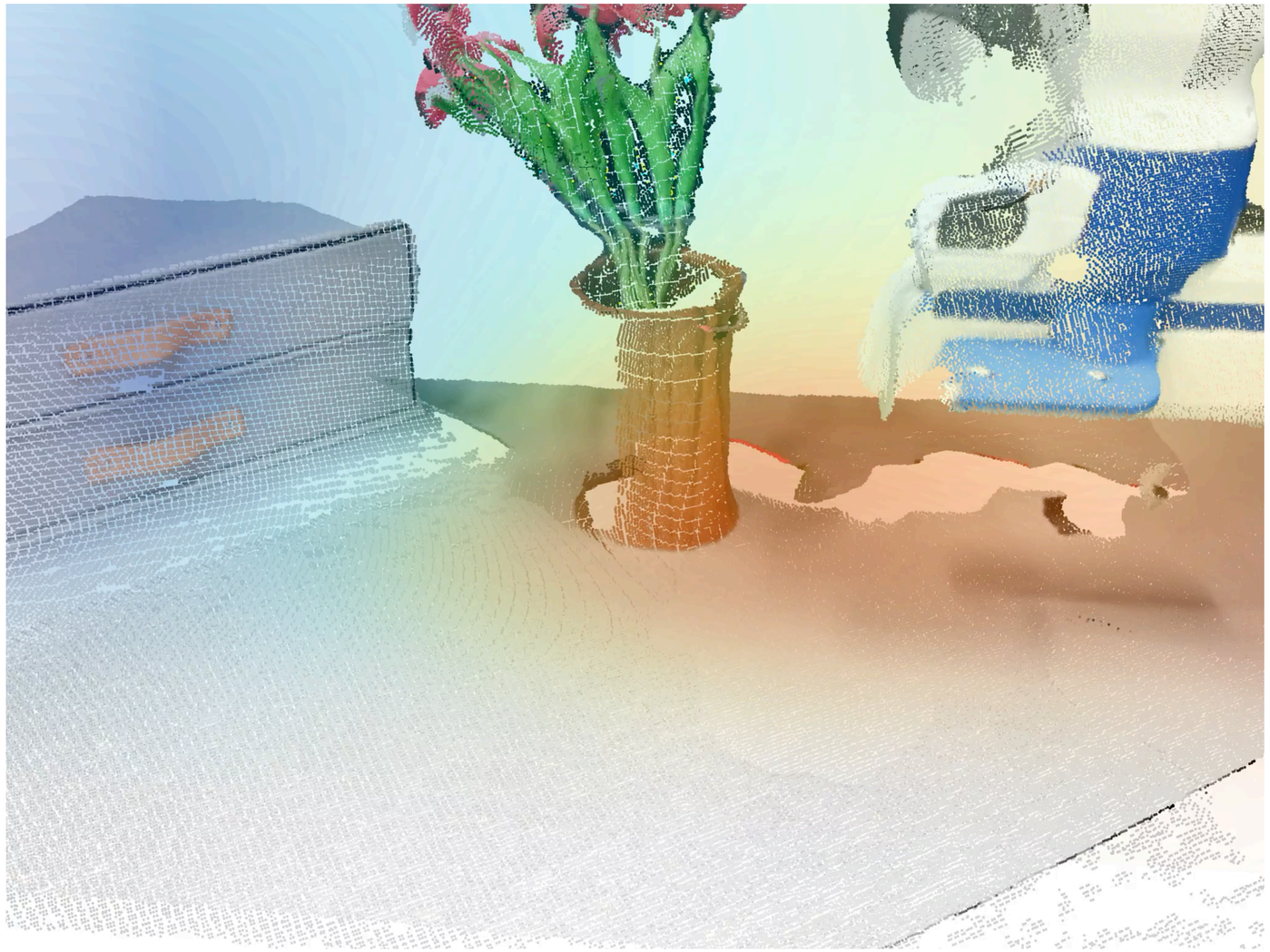
```
vases = detect('vase')
vase = vases[0]
xyz = vase.occupancy_grid
map[xyz] = -1
```

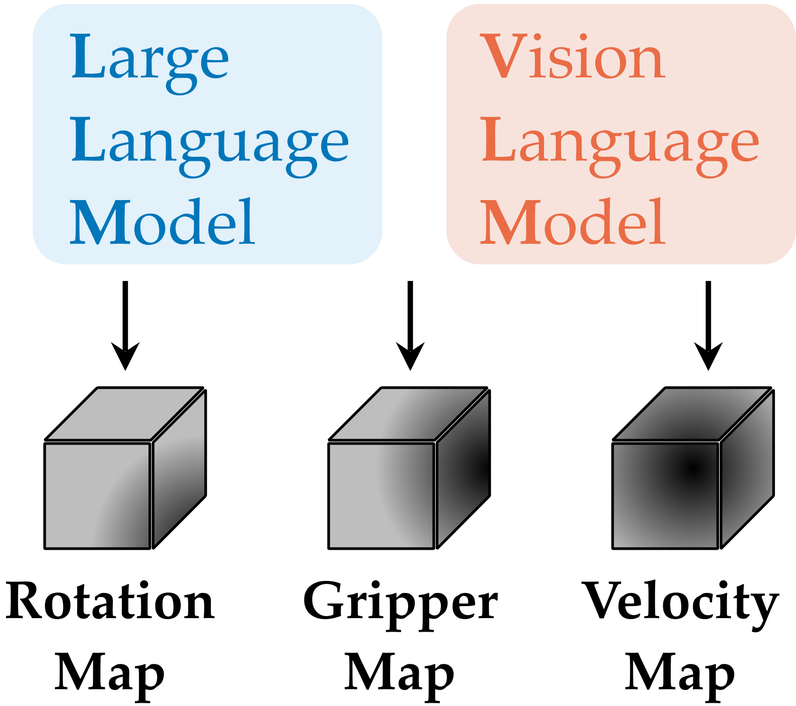


Also watch out for that vase.

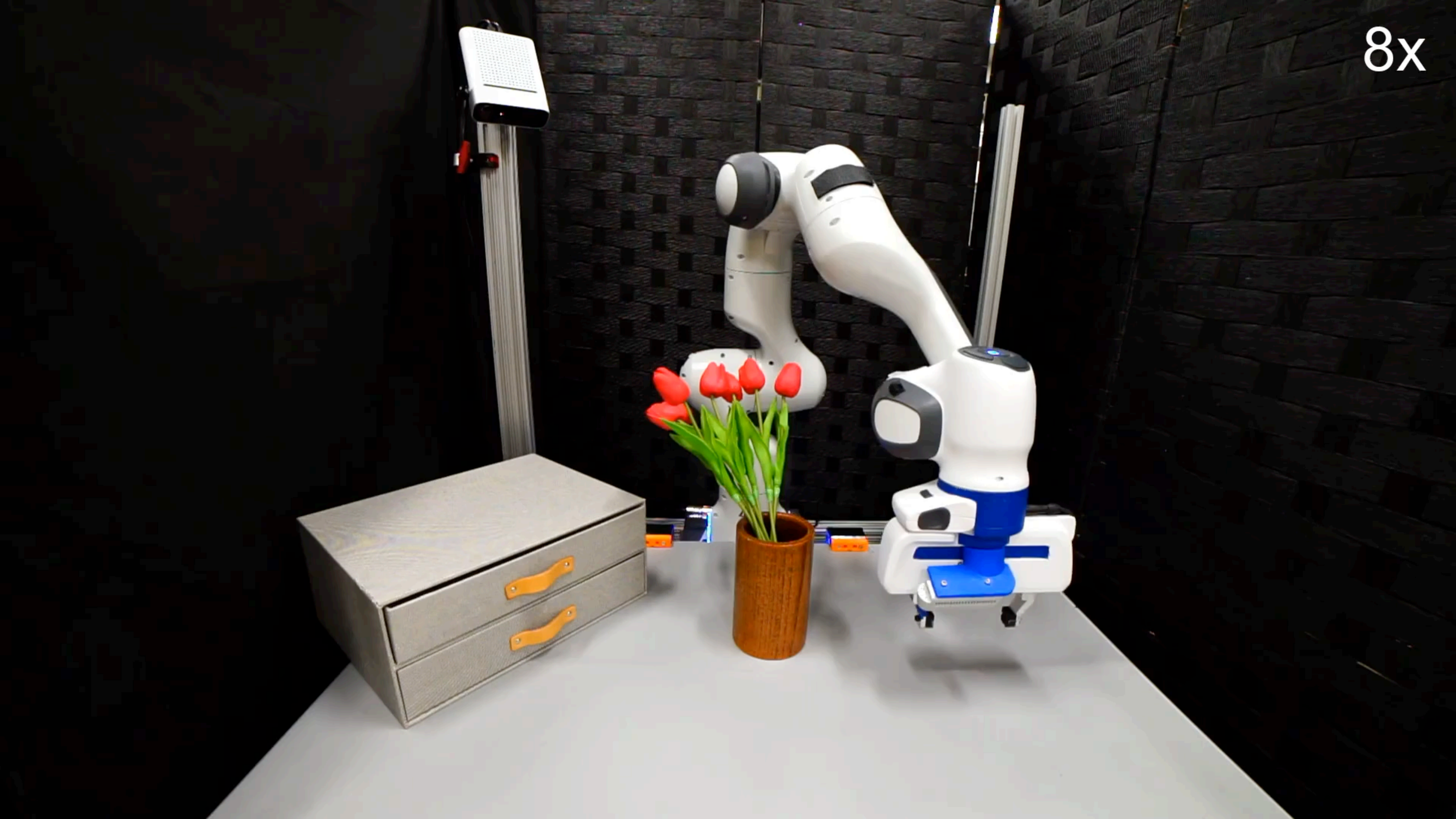
LLM Output

```
vases = detect('vase')
vase = vases[0]
xyz = vase.occupancy_grid
map[xyz] = -1
map = smooth(map)
```

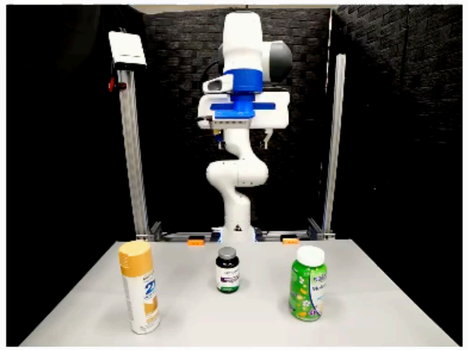


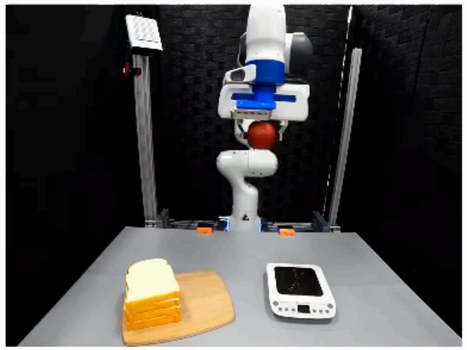

Large Language Model
Vision Language Model
Rotation Map
Gripper Map
Velocity Map

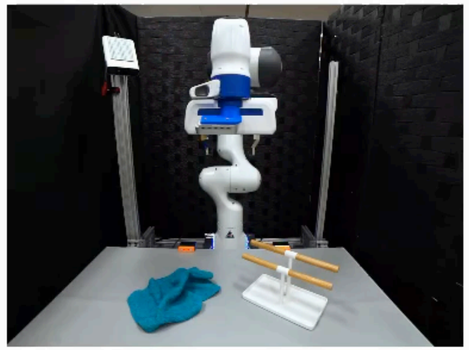
Motion Planner

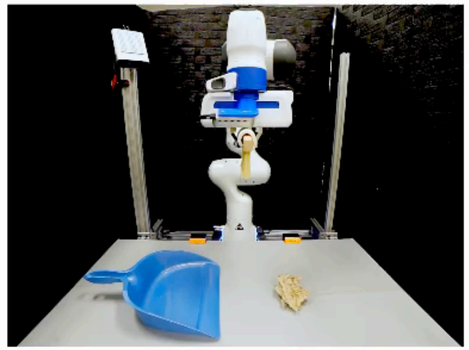

8x

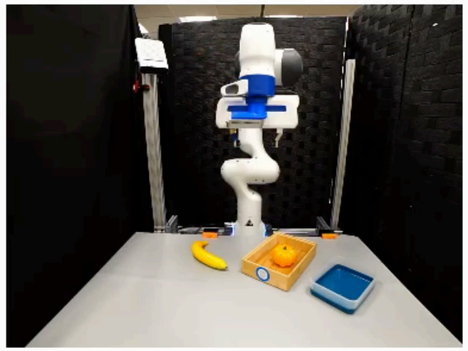
☑ no robot data required ☑ over 20 manipulation tasks

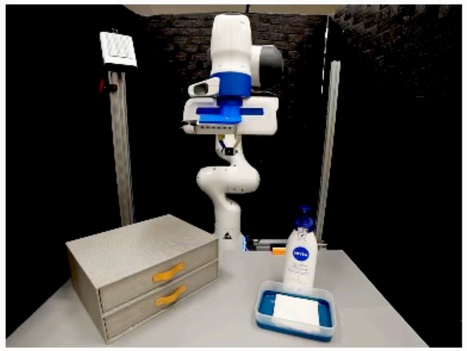

"Turn open vitamin bottle"

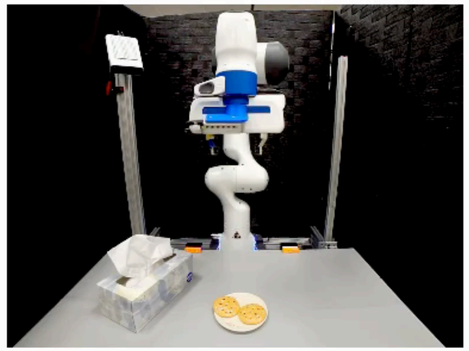

"Measure weight of apple"

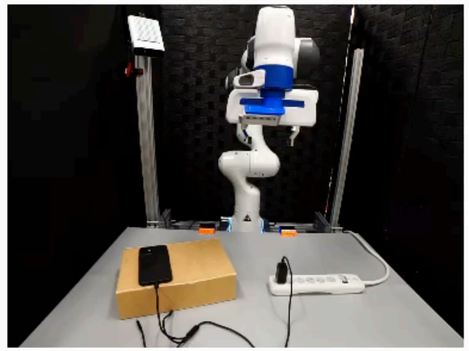

"Hang towel on rack"

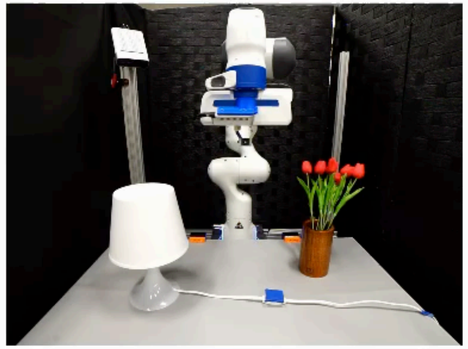

"Sweep trash into dustpan"

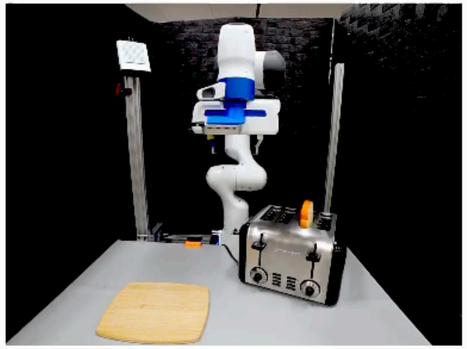

"Sort trash to blue tray"

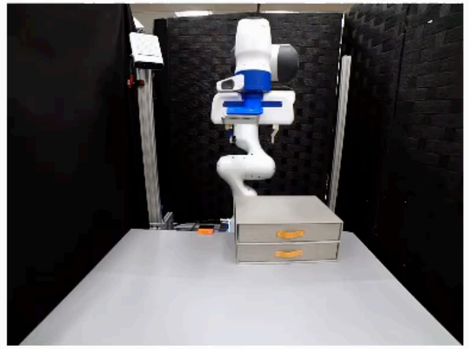

"Press down moisturizer pump"

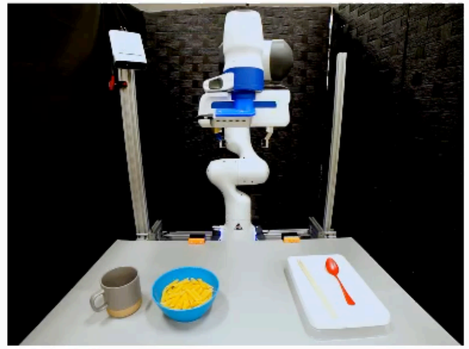

"Take out a napkin"


"Unplug charger for phone"


"Turn on lamp"


"Take out bread from toaster"


"Close top drawer"


"Set table for pasta"

Discussion

How reliable these methods that leverage off-the-shelf LLM/ VLMs for generating robot control?

Manipulate-Anything: Automating Real-World Robots using Vision-Language Models

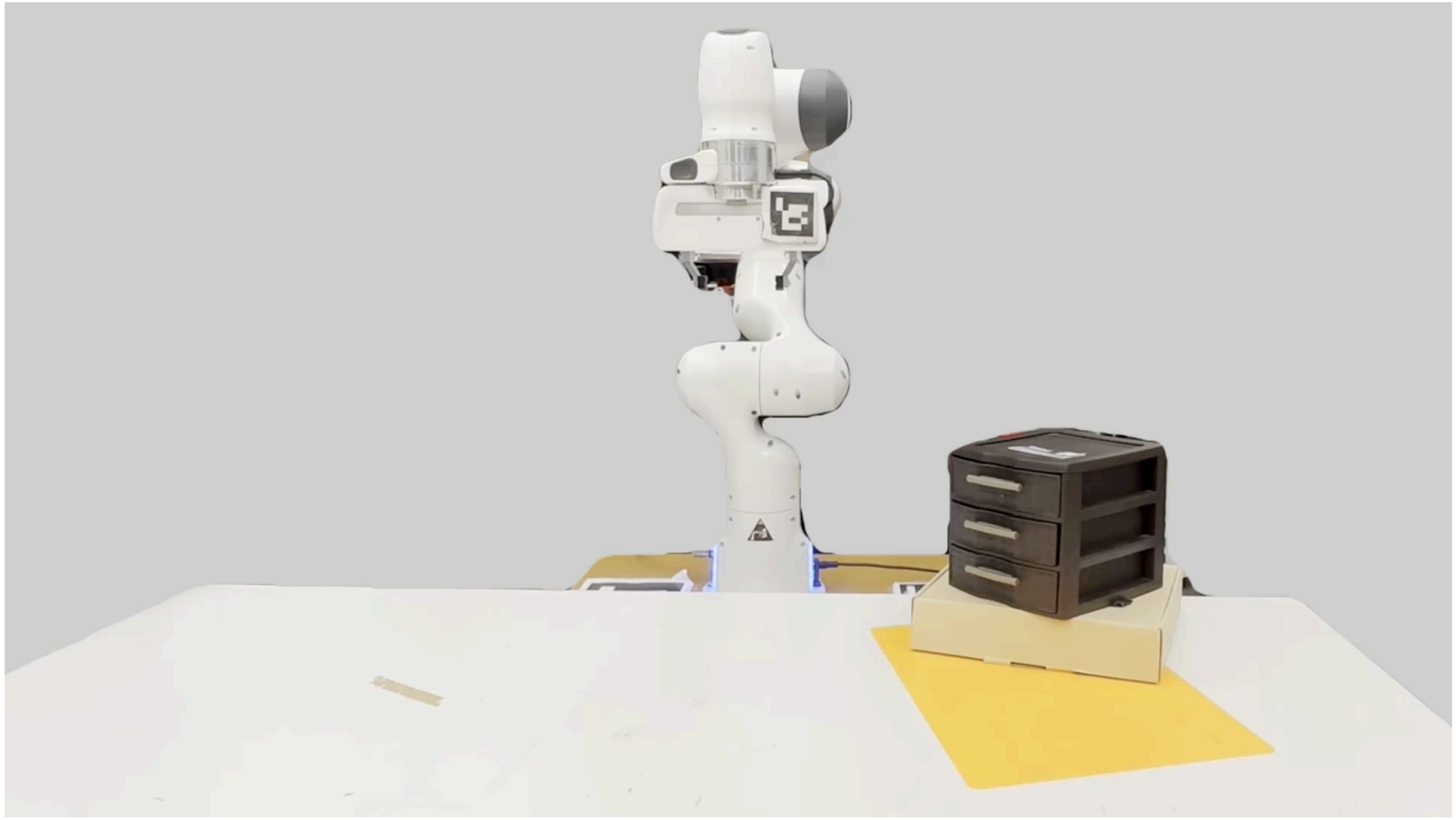
CoRL 2024

Jiafei Duan Wentao Yuan Wilbert Pumacay Yi Ru Wang Kiana Ehsani

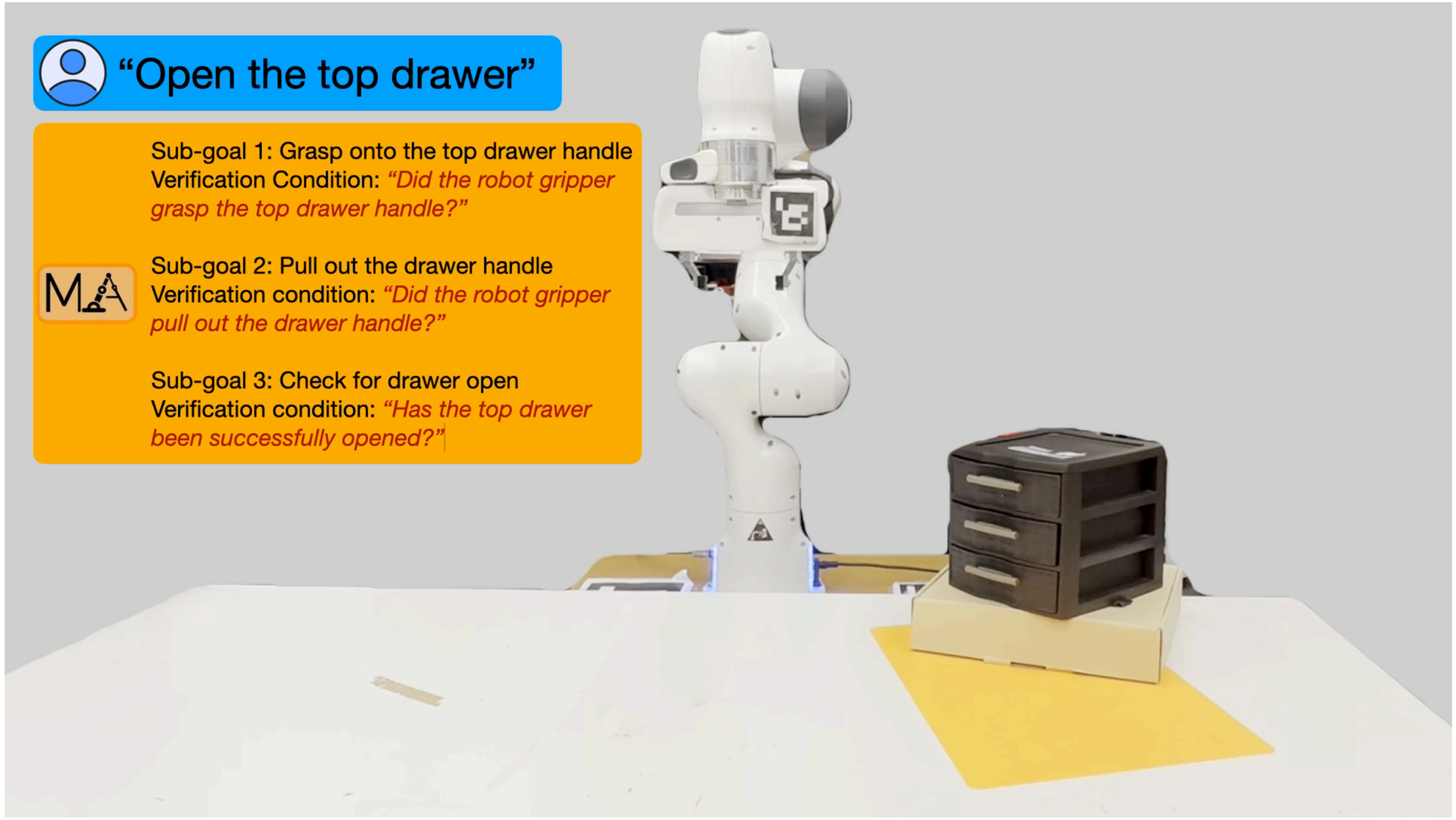
Dieter Fox Ranjay Krishna

Manipulate-Anything

What is the best way to leverage vision-large-model to generate robot data?

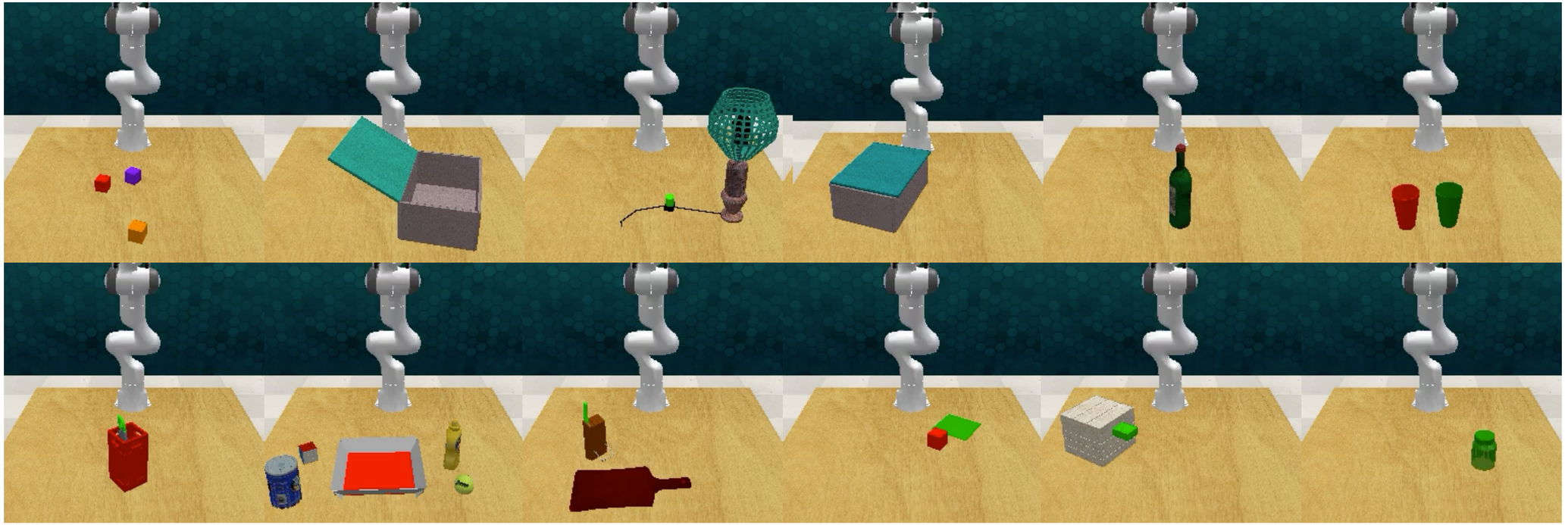


Manipulate-Anything



Manipulate-Anything

Data generation quality of Manipulate-Anything

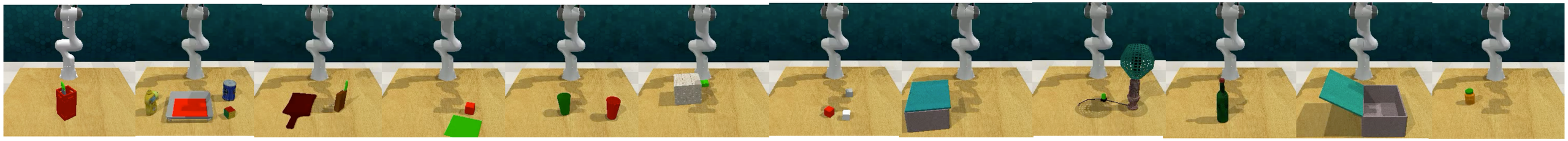


Method	Put_block	Play_jenga	Open_jar	Close_box	Open_box	Pickup_cup
VoxPoser [3]	70.7±2.31	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	26.7±14.00
CAP [4]	84.00±16.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	14.67±4.62
MA (Ours)	**96.00**±4.00	**77.33**±6.11	**80.00**±4.00	**33.33**±12.86	**29.00**±10.07	**82.67**±14.04

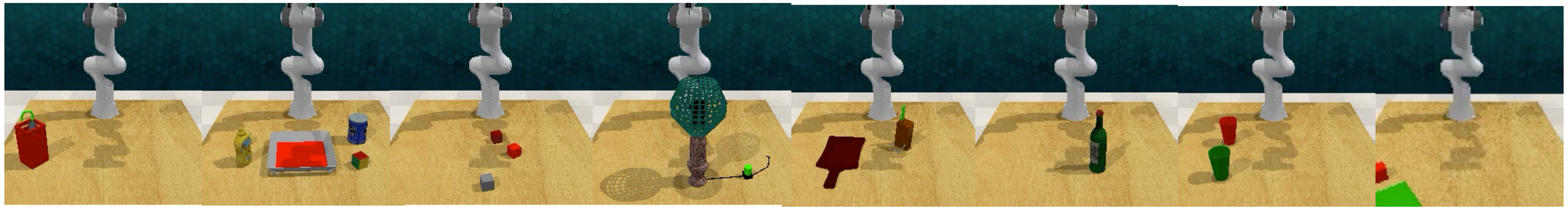
Method	Take_umbrella	Sort_mustard	Open_wine	Lamp_on	Put_knife	Pick_&_lift
VoxPoser[3]	33.33±8.33	**96.0**±6.93	8.00±4.00	57.3±12.22	**92.00**±4.00	96.00±0.00
CAP[4]	4.00±4.00	0.00±0.00	0.00±0.00	64.00±6.93	14.67±8.33	**100.00**±0.00
MA (Ours)	**61.33**±20.13	64.00±6.93	**42.00**±4.00	**69.33**±6.11	52.00±10.58	84.00±6.93

Data generation quality of Manipulate-Anything

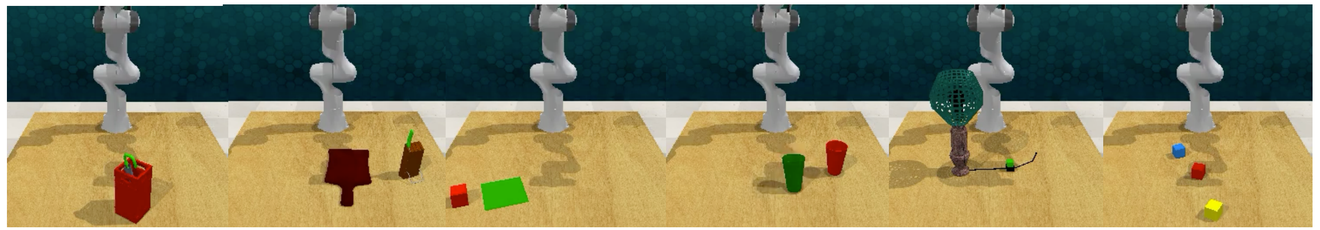
Manipulate-Anything (Zero-shot)



VoxPoser (Zero-shot)

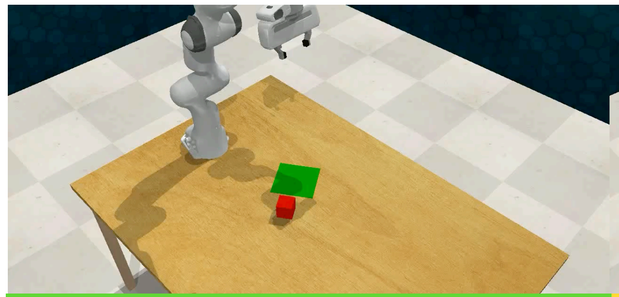
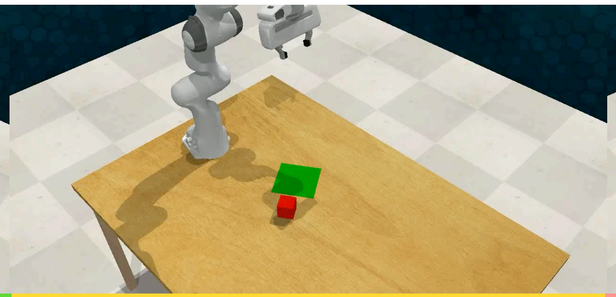
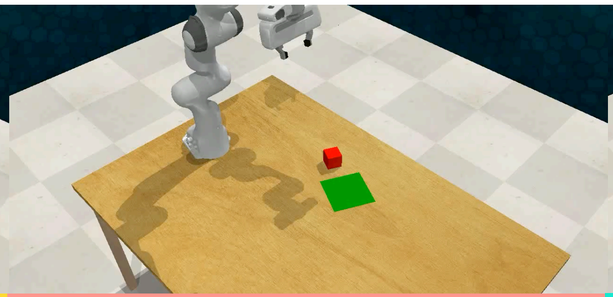
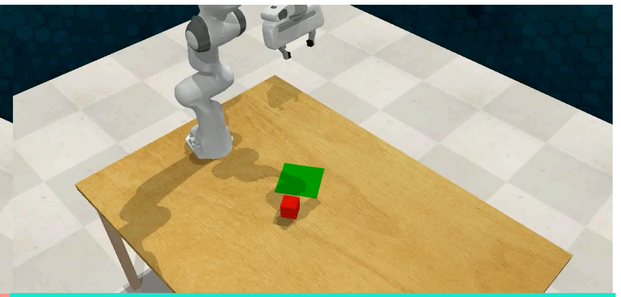


Code-As-Policies (Zero-shot)



Manipulate-Anything is capable of generating success task trajectories across a diverse set of tasks, and outperform the next SoTA methods in 9/12 of the simulation tasks.

Performance model trained on MA data

Manipulate-Anything | RLBench | VoxPoser | Code-As-Policies

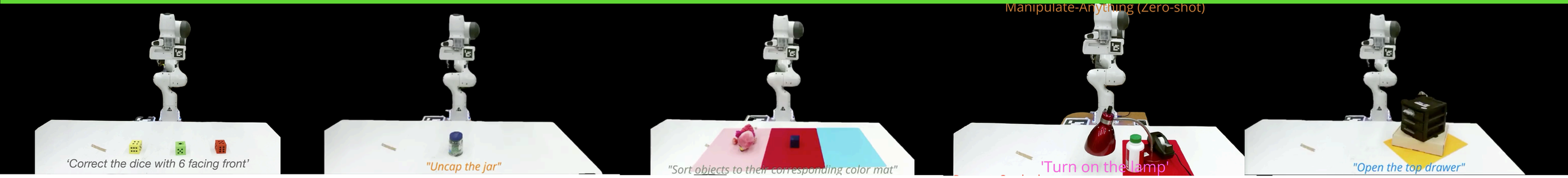
Generated data	Put_block	Play_jenga	Open_jar	Close_box	Open_box	Pickup_cup
VoxPoser[3]	2.67±2.31	-	-	-	-	4.00±4.00
CAP[4]	6.67±2.31	-	-	-	-	14.67±12.86
MA (Ours)	**85.33**±10.07	**81.33**±2.31	21.33±10.07	42.67±8.33	**30.67**±11.55	54.00±12.49
RLBench[33]	20.00±18.33	**81.33**±9.24	**58.67**±45.49	**68.00**±24.98	14.67±6.11	**54.67**±23.09

Generated data	Take_umbrella	Sort_mustard	Open_wine	Lamp_on	Put_knife	Pick_&_lift
VoxPoser[3]	4.00±4.00	0.00±0.00	1.33±2.31	5.33±4.62	1.33±2.31	5.67±1.64
CAP[4]	13.33±10.06	-	-	8.00±16.00	9.33±6.11	46.67±2.31
MA (Ours)	**84.00**±6.93	**53.33**±6.11	**86.67**±6.11	**89.33**±6.11	8.00±4.00	33.33±2.31
RLBench[33]	58.67±50.80	**53.33**±34.02	**86.67**±12.86	84.00±13.86	**30.67**±10.07	**62.67**±9.24

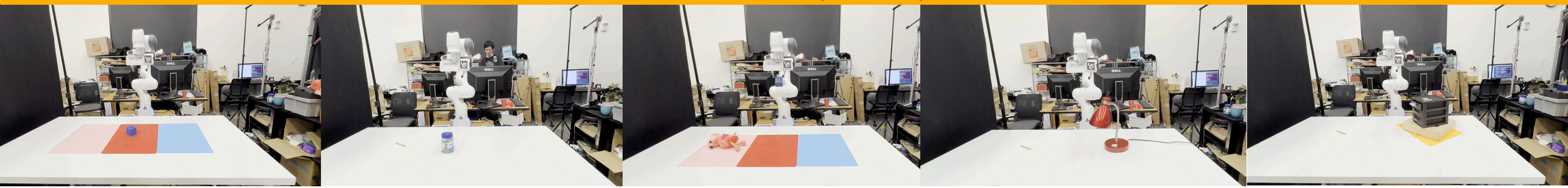
Training with MA data leads to comparable results as with using human annotated data, and we see more robust model performances when trained with MA data.

Performance model trained on MA data

Manipulate-Anything (Zero-shot)



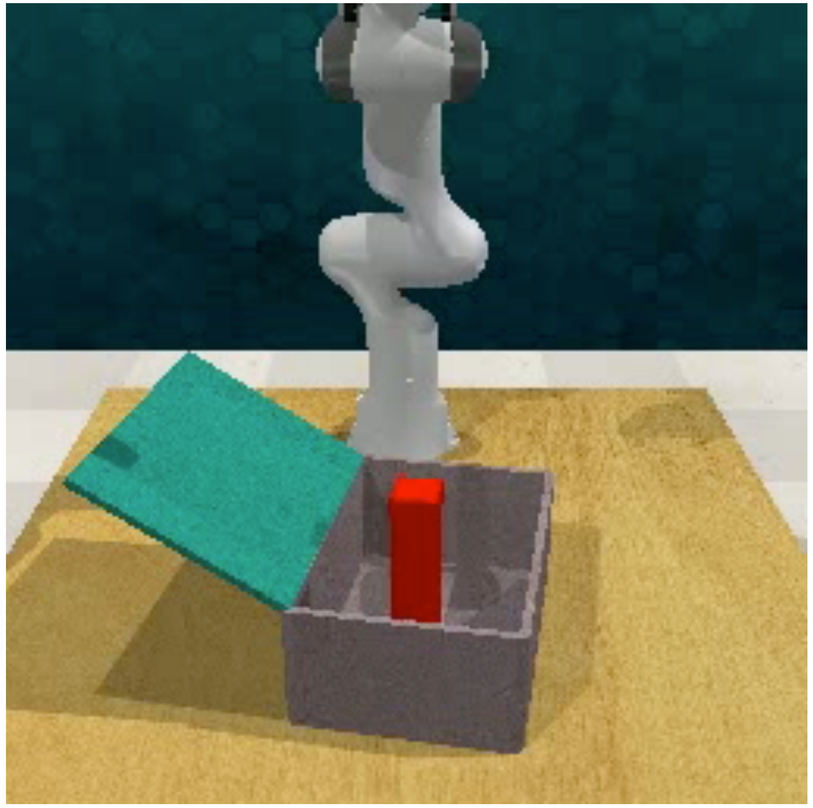
Code-As-Policies (Zero-shot)



	Open_drawer	Sort_object	On_lamp	Open_jar	Correct_dice
CAP (0-shot)	0.00 ± 0.00	13.33 ± 5.77	0.00 ± 0.00	6.67 ± 5.77	6.67 ± 5.77
MA (0-shot)	**36.67**±5.77	**60.00**±10.00	**26.67**±11.55	**40.00**±10.00	**53.33**±5.77
PerAct (MA data)	50.00 ± 0.00	33.33 ± 5.77	46.67 ± 5.77	56.67 ± 5.77	60.00 ± 0.00
PerAct (Human data)	**53.33**±11.55	**36.67**±5.77	**60.00**±0.00	**76.67**±5.77	**80.00**±10.00

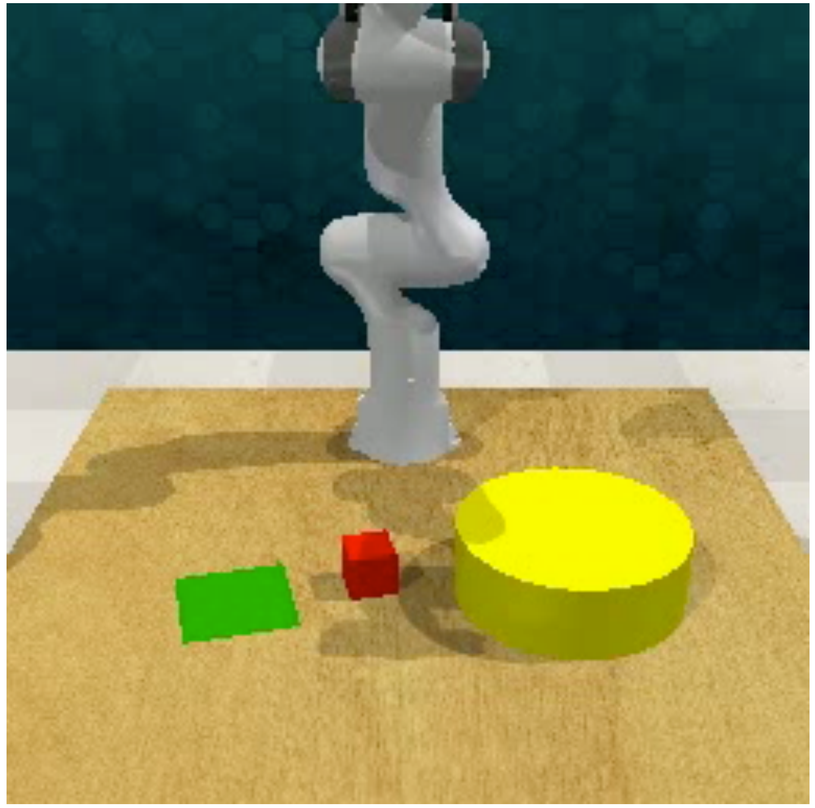
We observed similar trends in real-world results as to simulation.

Additional experiments- Exploratory task generation



Proposed Task 1: Put the red block flat down into the box

Proposed Task 2: Close the box



Proposed Task 1: Place the red block onto the green patch

Proposed Task 2: Hide the red block

Additional experiments- Robustness in generation

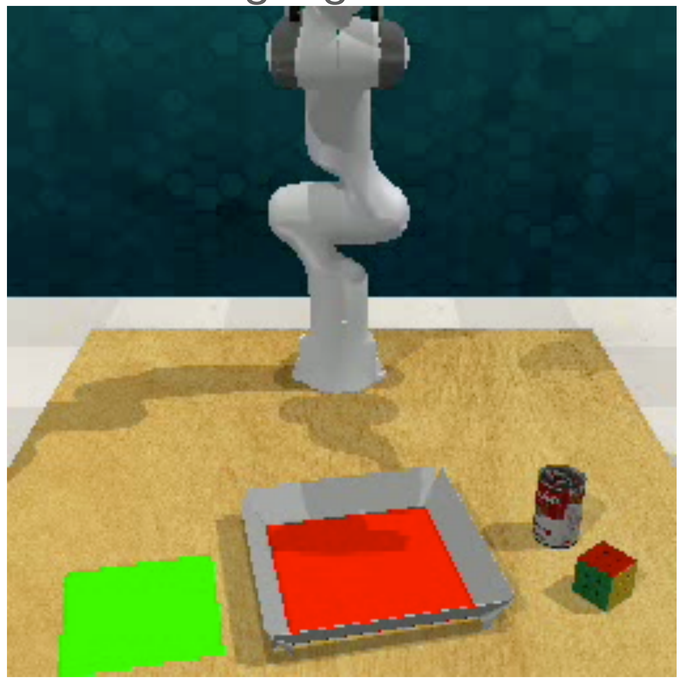
Original Task

‘Place the soup can onto the red patch’

Object-specific variations

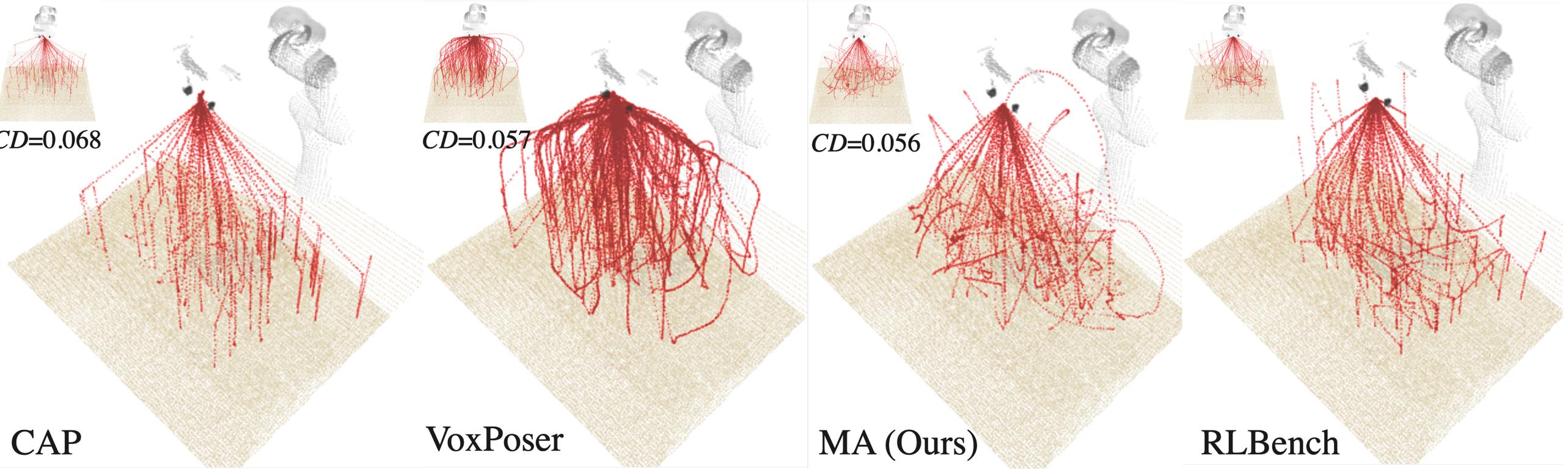
‘Pick up the container by the patterned sides’

Language variation



‘Place the container onto green patch’

Interesting insights



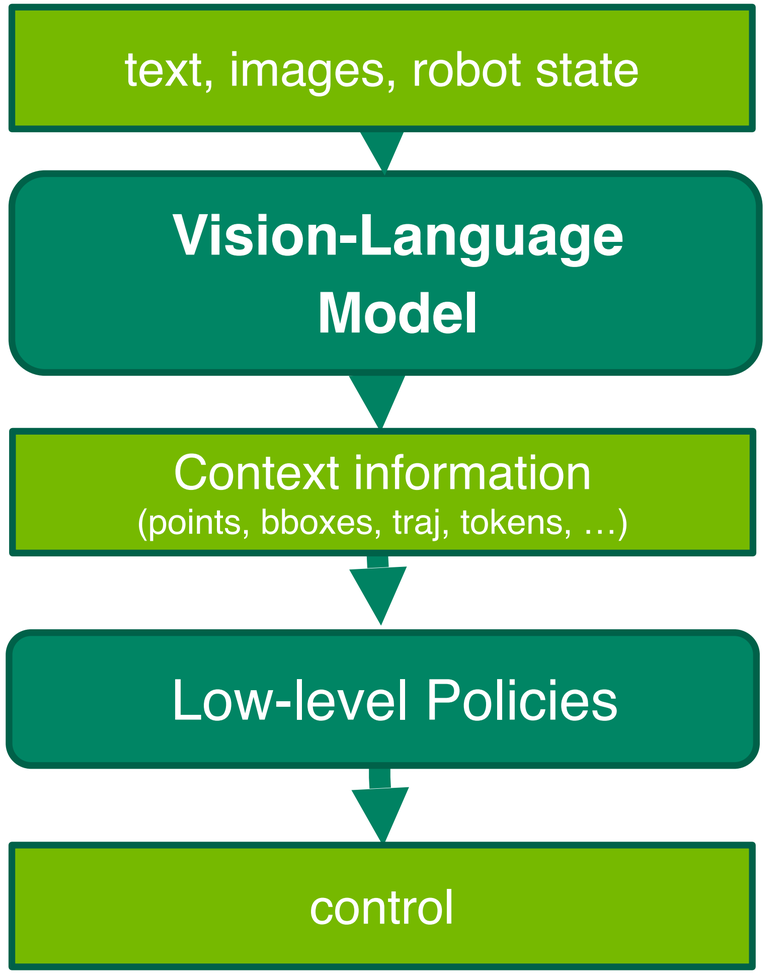
Action Distribution

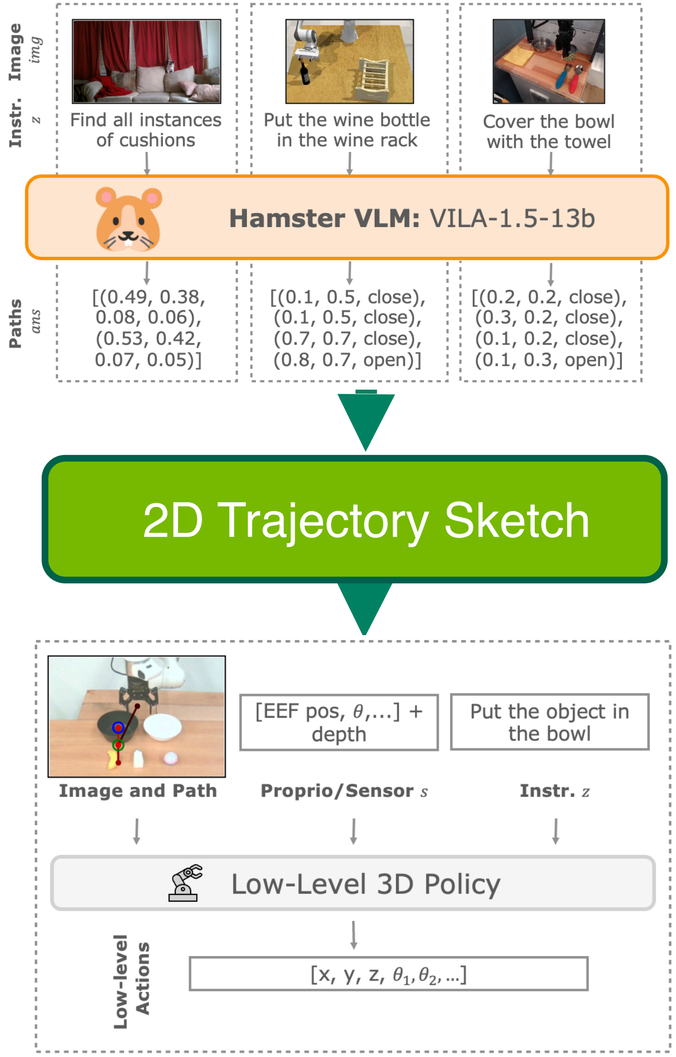
The formulation of Manipulate-Anything for zero-shot data generator give more human-like trajectories.

HAMSTER

Hierarchical Action Models for Open-World Robot Manipulation

HiRobot, GeminiRobotics, GR00T-N1, HAMSTER, …





- VILA-1.5-13b open model
- Pre-trained on internet-scale data for open-world visual reasoning
- Fine-tuned on sim+real robot tasks

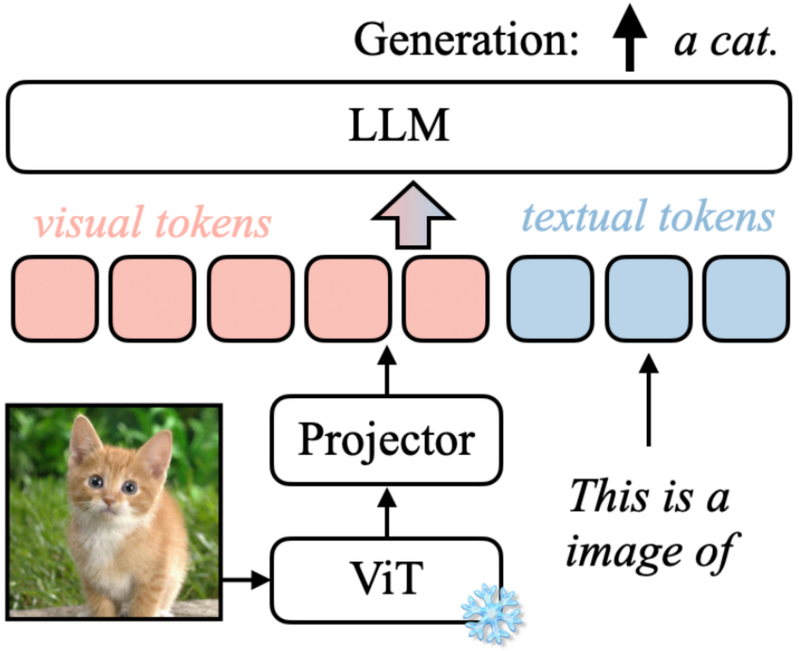
[Lin-Yin-Ping-Molchanov-Shoeybi-Han: CVPR-24]

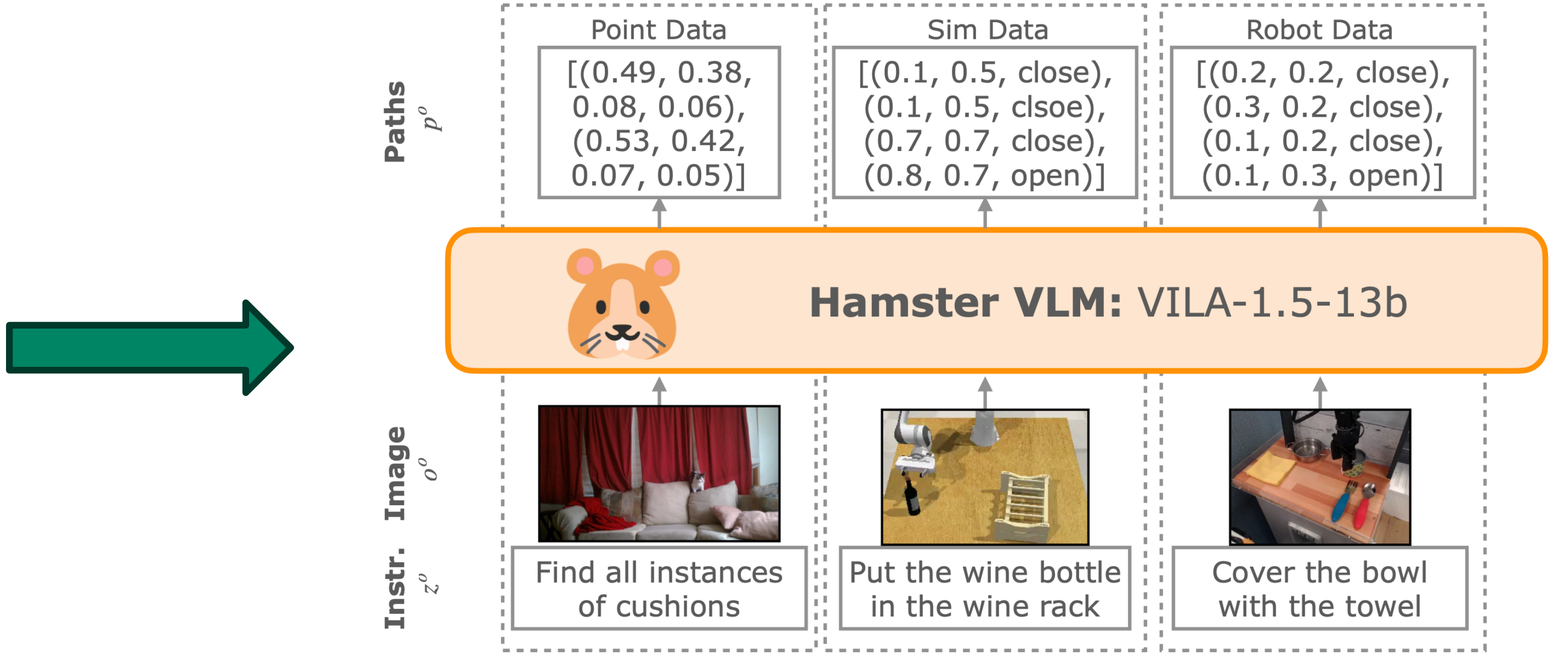
- 3DDA or RVT-2 motion policy
- Trained on less, real robot demos

3DDA [Ke-Gkanatsios-Fragkiadaki: CoRL-24]
RVT-2[Goyal-Blukis-Xu-Guo-Chao-Fox: CoRL-24]

NVIDIA.
PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

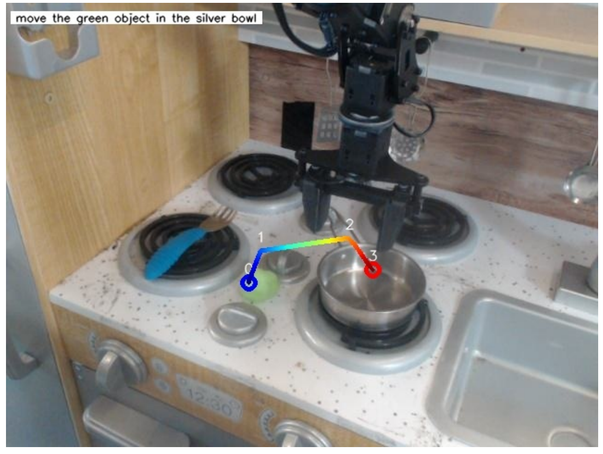
Fine-Tuning VILA



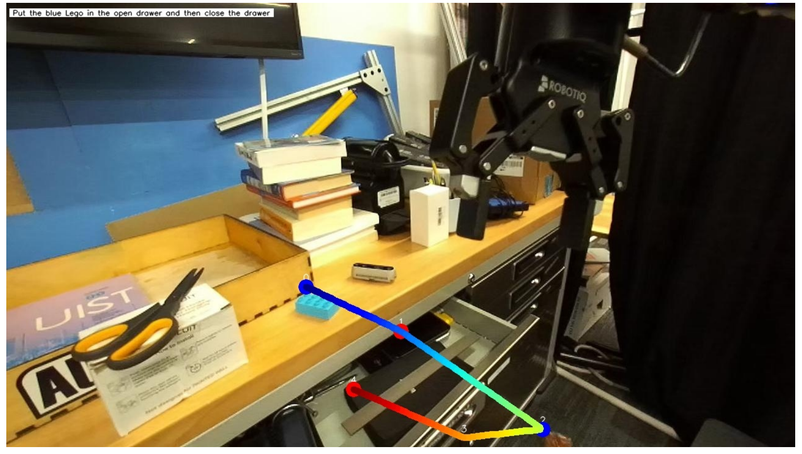


Fine-Tuning VILA

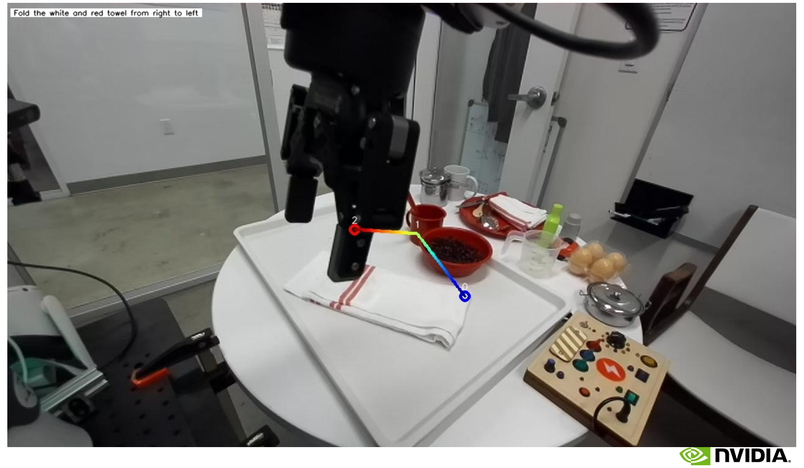
Move the green object in the silver bowl



Put the blue Lego in the open drawer and then close the drawer



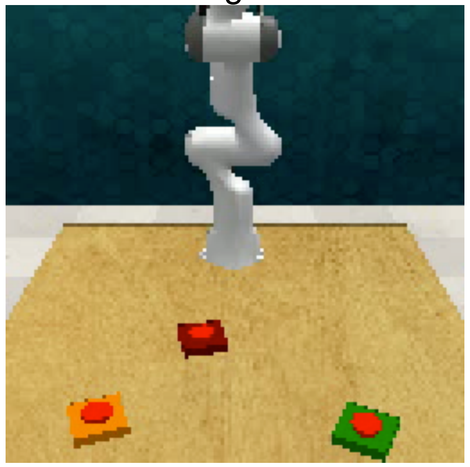
Fold the white and red towel from right to left

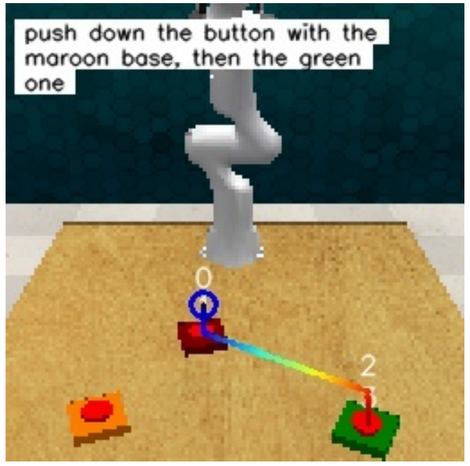


Fine-Tuning VILA

Screw in the rose light bulb

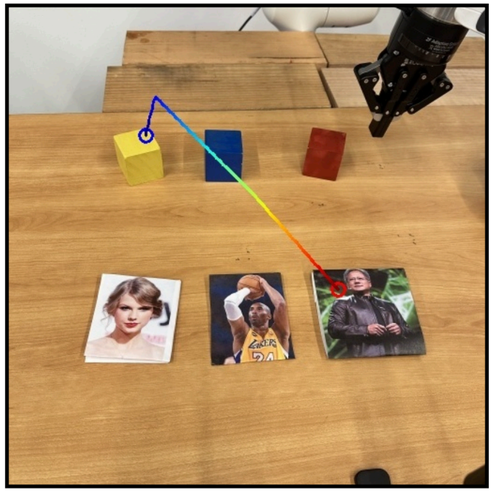
Push down the button with maroon base, then the green one



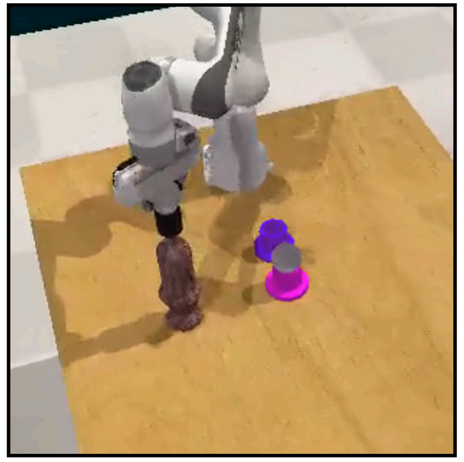


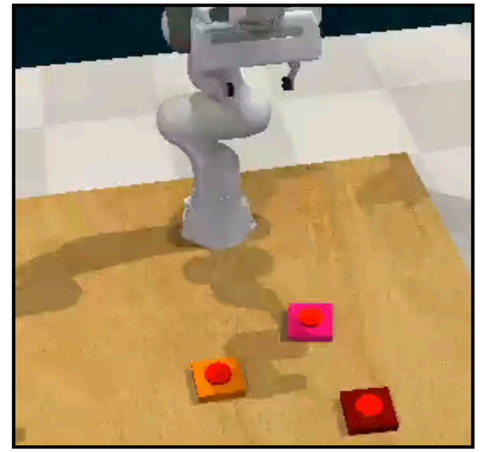
Slide the bottom drawer open

Fine-Tuned VILA: Evaluation Examples

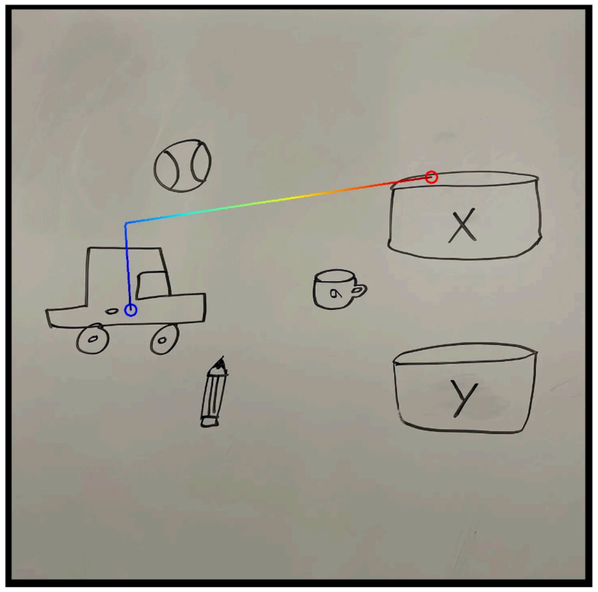


Move the block to Jensen Huang

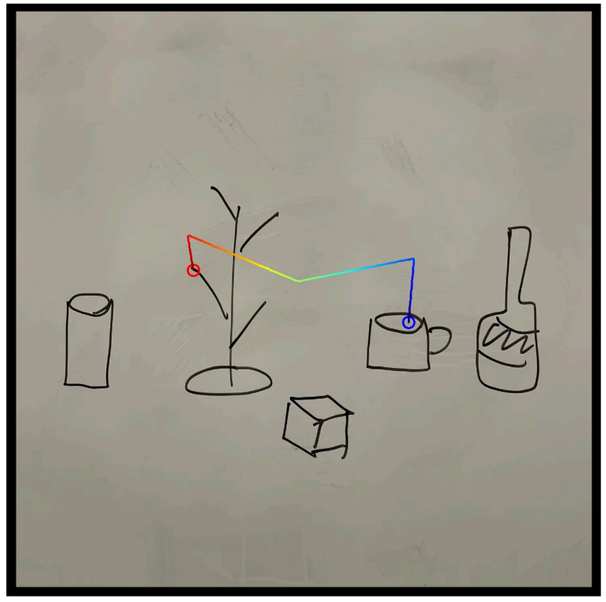




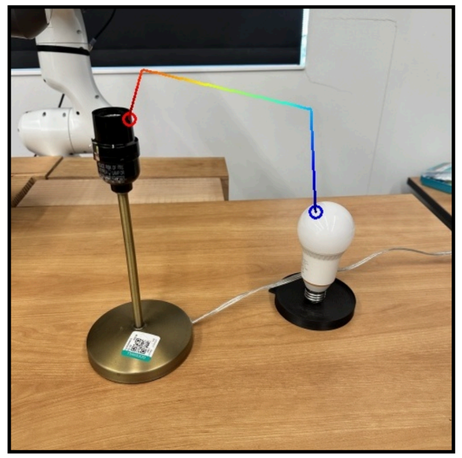
RLBench examples



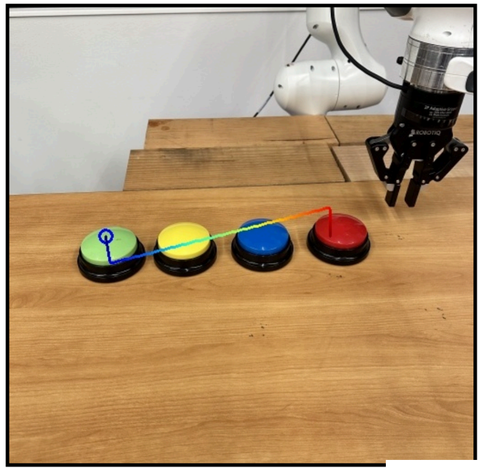
Move the toy car to the bowl with x



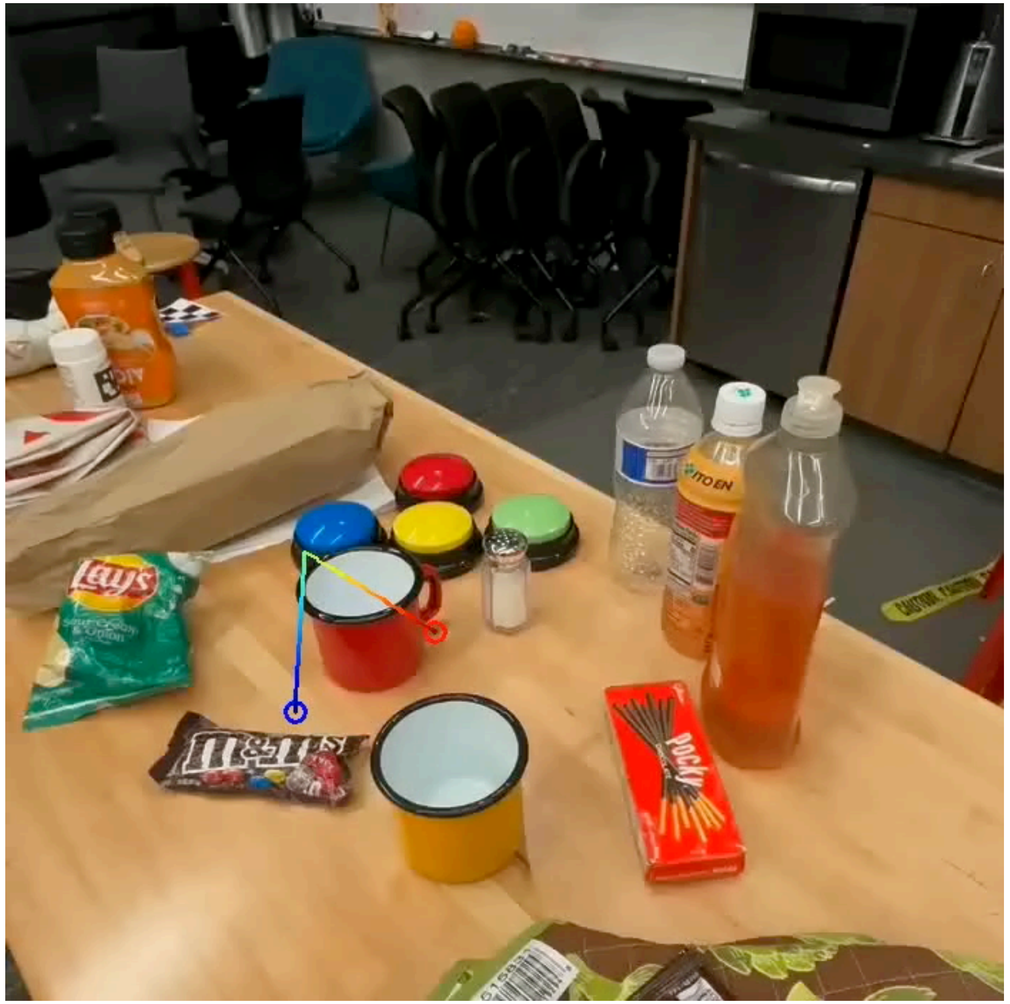
Place the cup on the cup holder



Screw the light bulb in the lamp



Push the button with the color of cucumber, then press the button with color of fire

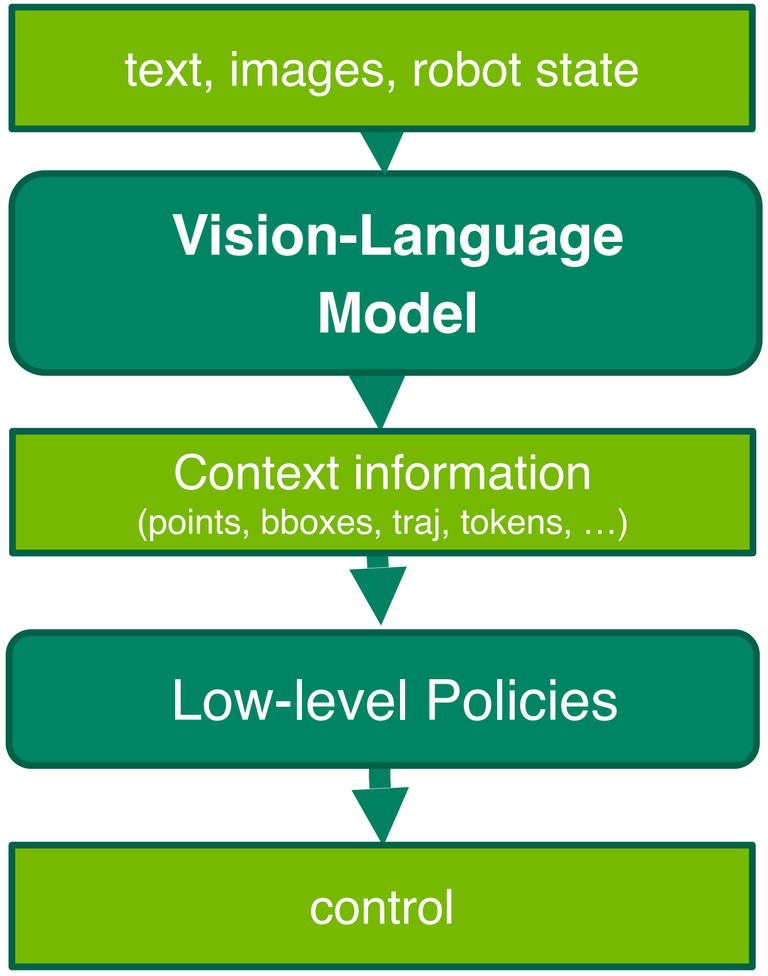


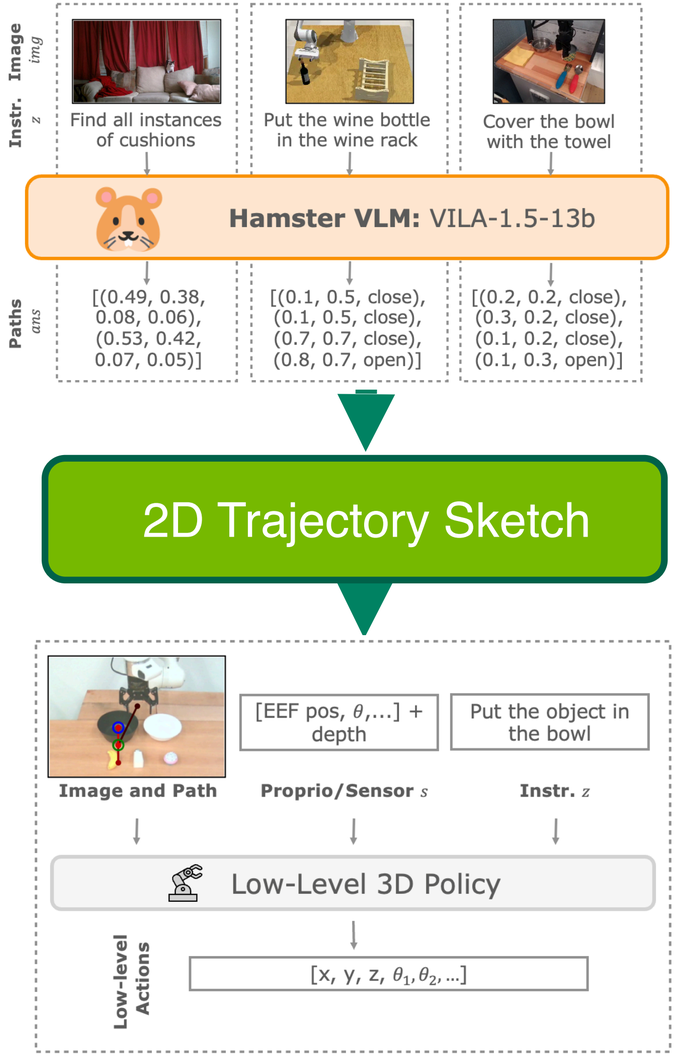
Pick up the M&M chocolate and put it in the yellow mug

HAMSTER

Hierarchical Action Models for Open-World Robot Manipulation

HiRobot, GeminiRobotics, GR00T-N1, HAMSTER, …





- VILA-1.5-13b open model
- Pre-trained on internet-scale data for open-world visual reasoning
- Fine-tuned on sim+real robot tasks

[Lin-Yin-Ping-Molchanov-Shoeybi-Han: CVPR-24]

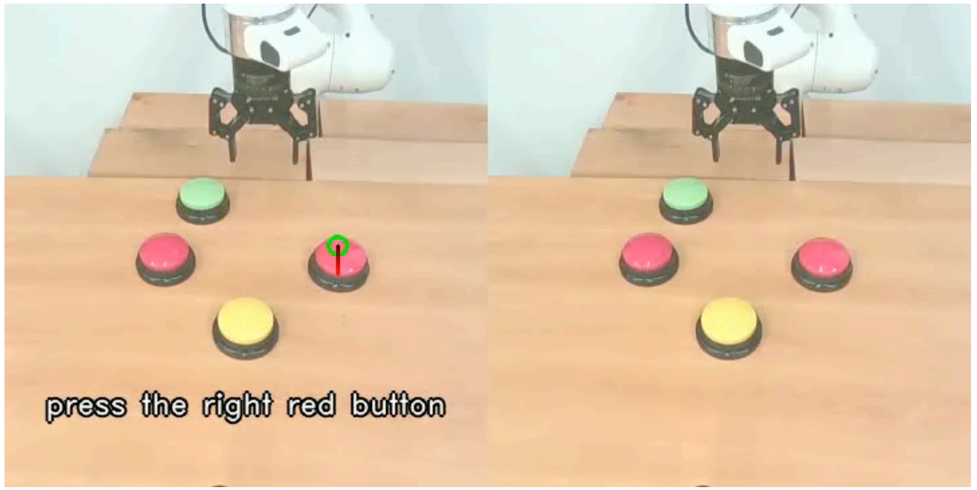
- 3DDA or RVT-2 motion policy
- Trained on less, real robot demos

3DDA [Ke-Gkanatsios-Fragkiadaki: CoRL-24]
RVT-2[Goyal-Blukis-Xu-Guo-Chao-Fox: CoRL-24]

NVIDIA.
PAUL G. ALLEN SCHOOL OF COMPUTER SCIENCE & ENGINEERING

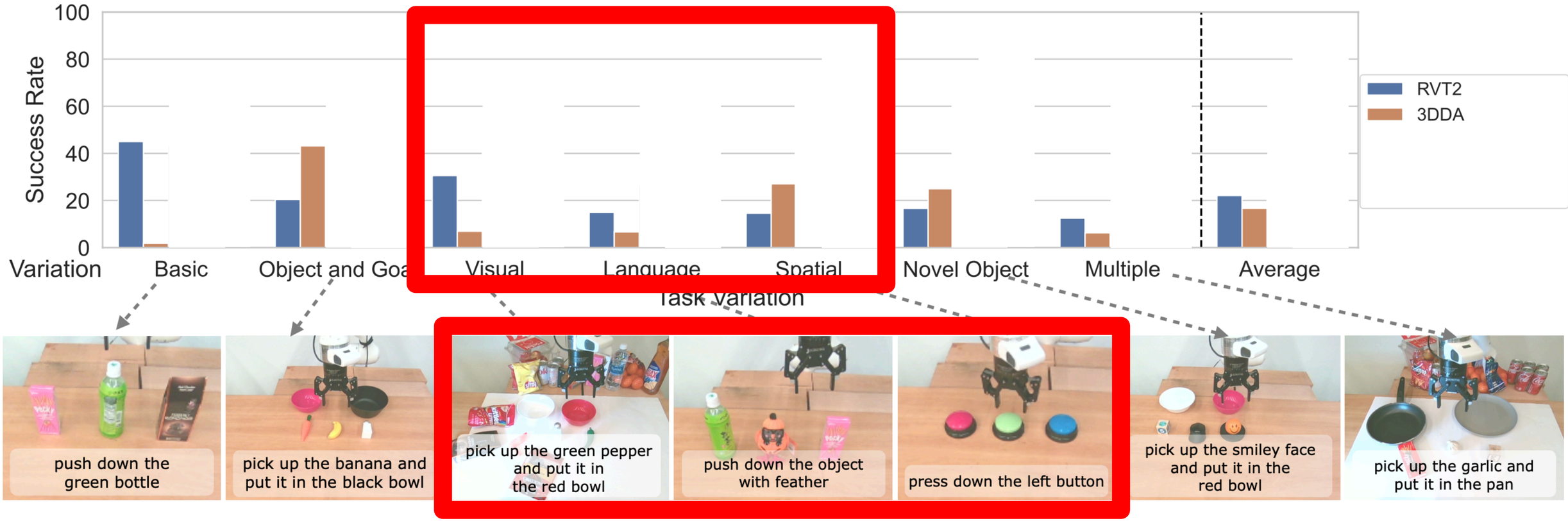
RVT2 [Goyal-Blukis-Xu-Guo-Chao-F: RSS-2024]

3DDA [Ke-Gkanatsios-Fragkiadaki: CoRL-2024]

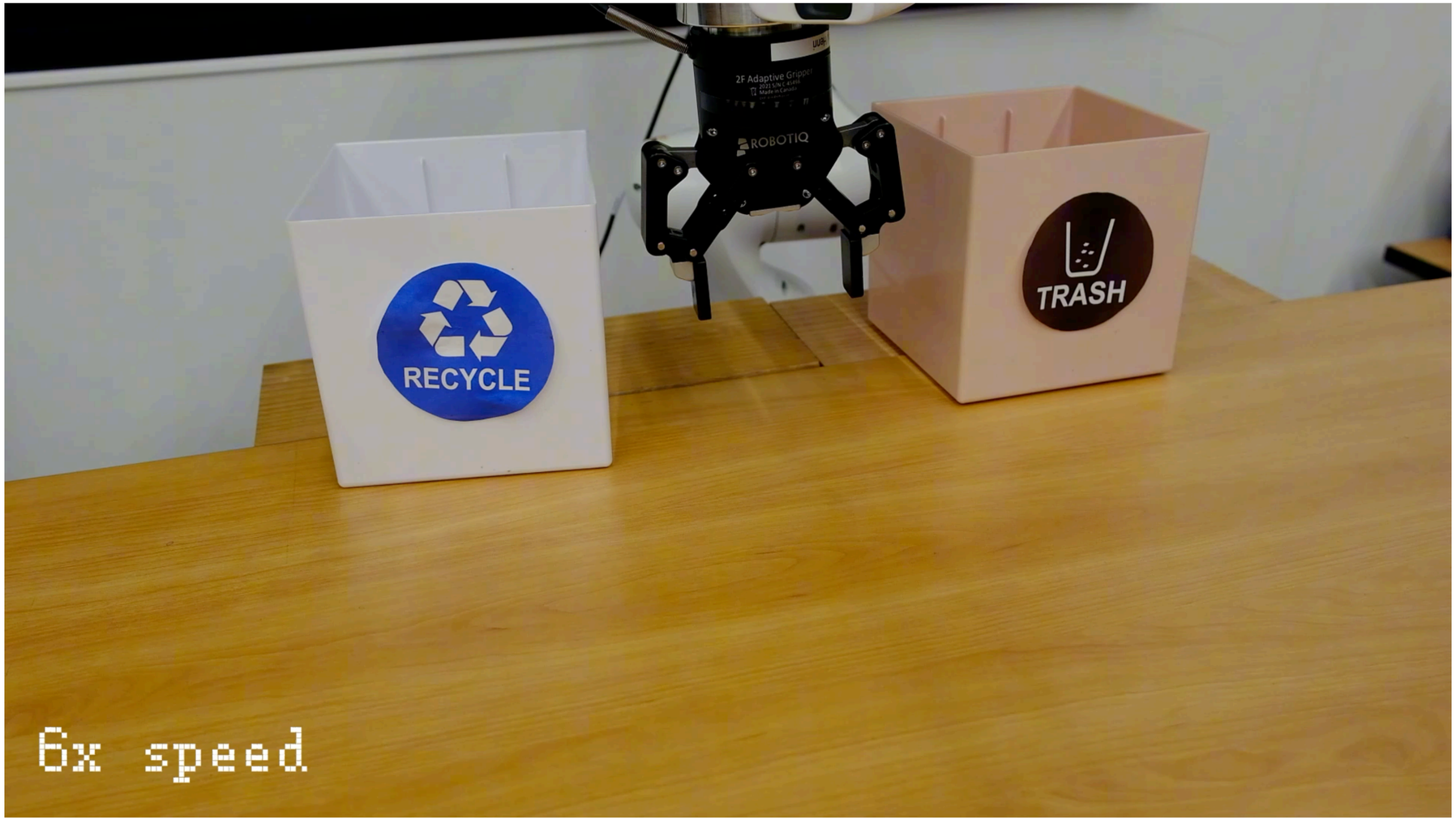


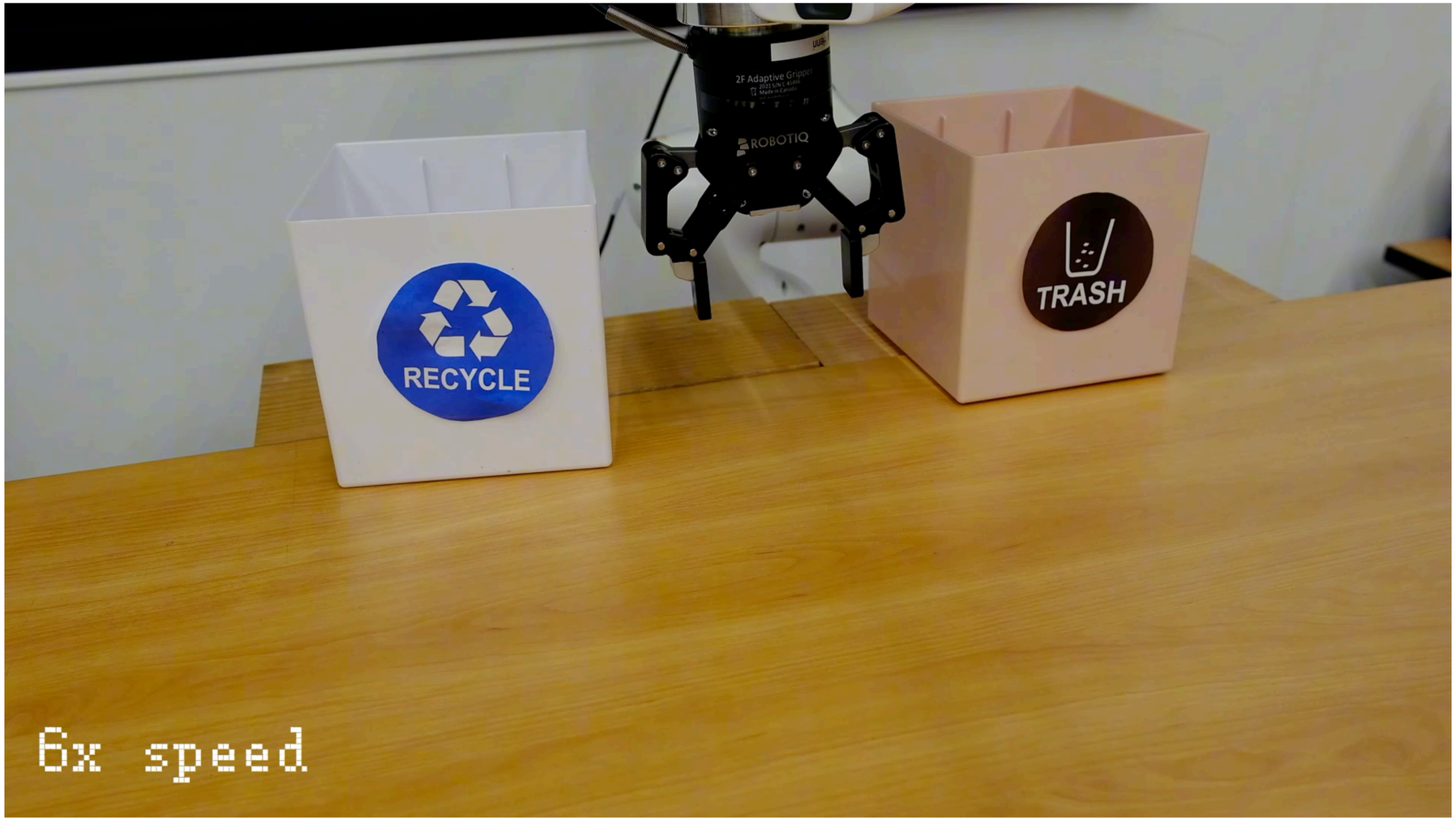
Changes between train and test scenes and tasks

- object arrangements
- visual appearance of scenes
- different names to refer to objects
- spatial language to refer to objects

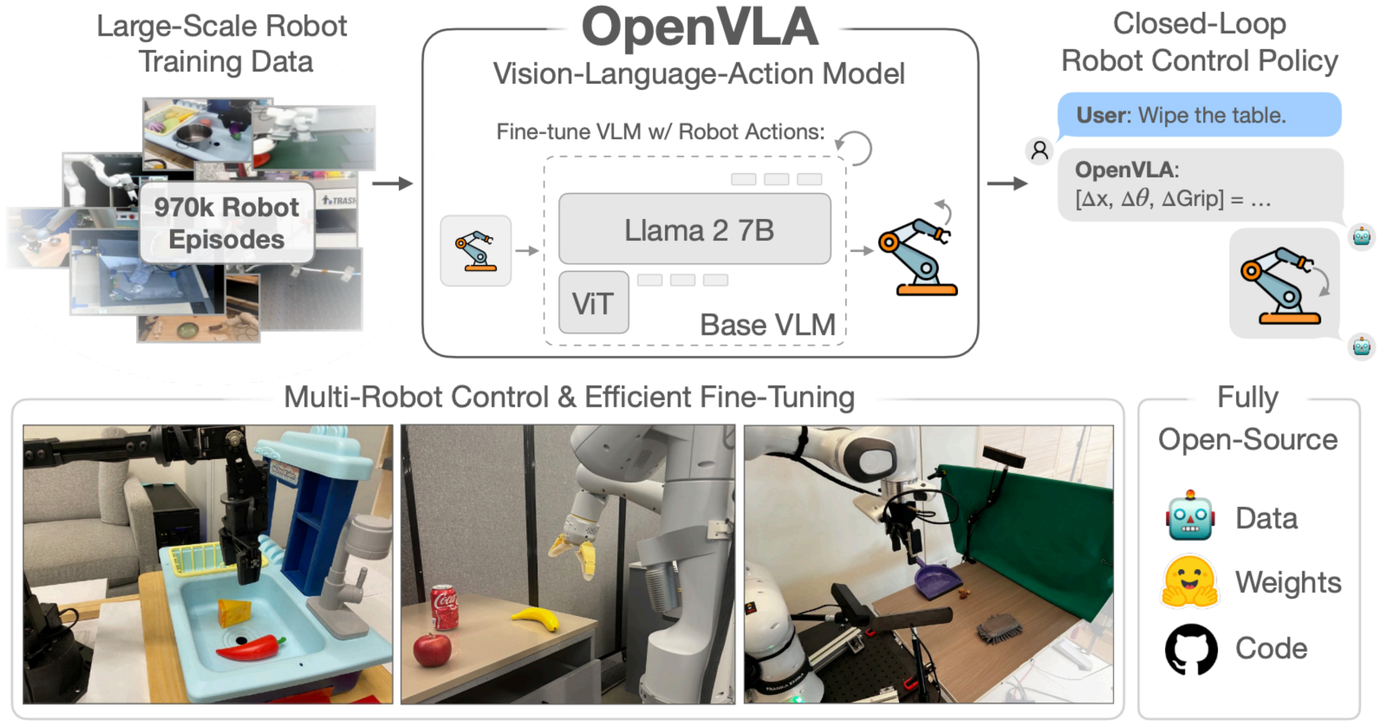

100
80
60
40
20
0
Success Rate
RVT2
3DDA
Variation
Basic
Object and Goa
Visual
Language
Spatial
Novel Object
Multiple
Average
Task Variation
push down the green bottle
pick up the banana and put it in the black bowl
pick up the green pepper and put it in the red bowl
push down the object with feather
press down the left button
pick up the smiley face and put it in the red bowl
pick up the garlic and put it in the pan


unfold the towel
把笑脸放到黑色杯子里
(put the smiley face in the black mug)
open the drawer
pick up the juice box and
put it in the top bin
have the coke on
Jensen Huang
press the blue button
wipe the table
close the drawer
fanta can should be in the
recycle bin


2F Adaptive Gripper
ROBOTIQ
RECYCLE
TRASH
6x speed


2F Adaptive Gripper
ROBOTIQ
RECYCLE
TRASH
6x speed

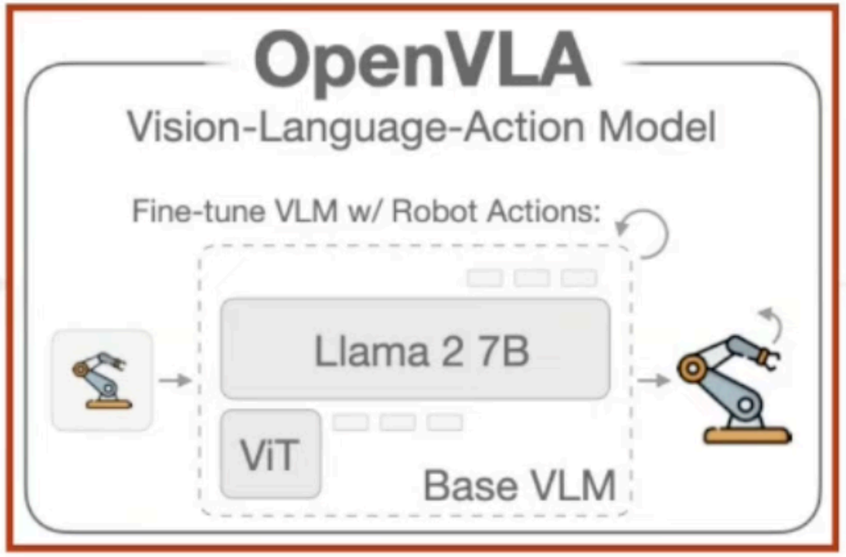
OpenVLA: An Open Source Vision-Language-Action Model



Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, Chelsea Finn

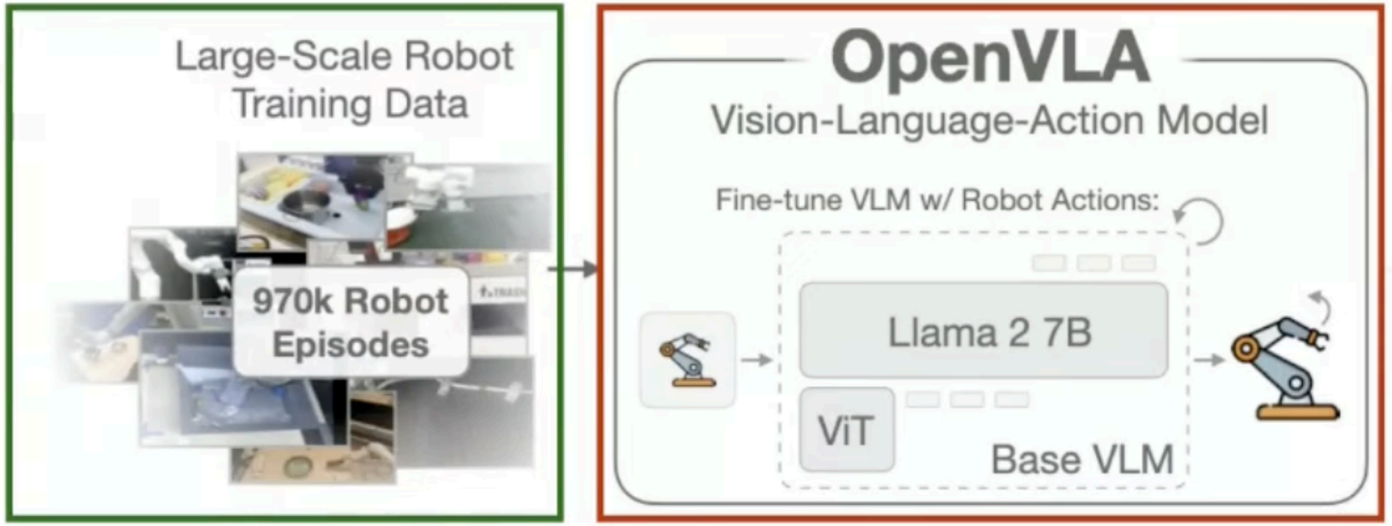
What is OpenVLA?

- 7B VLA built on pretrained "Prismatic VLM" backbone
  - LLM: Llama 2
  - Vision: DINOv2 + SigLIP



What is OpenVLA?

- 7B VLA built on pretrained "Prismatic VLM" backbone
  - LLM: Llama 2
  - Vision: DINOv2 + SigLIP

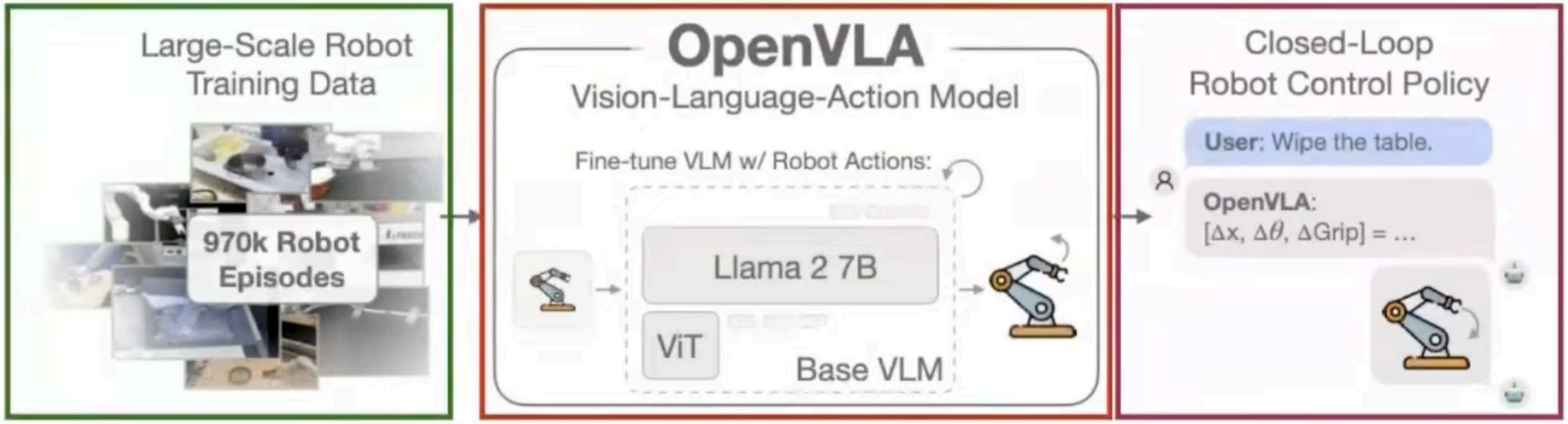


What is OpenVLA trained on?

- 970k episodes from Open X-Embodiment dataset
  - 27 real robot datasets
- Trained on 15 more datasets than RT-2-X

What is OpenVLA?

- 7B VLA built on pretrained "Prismatic VLM" backbone
  - LLM: Llama 2
  - Vision: DINOv2 + SigLIP



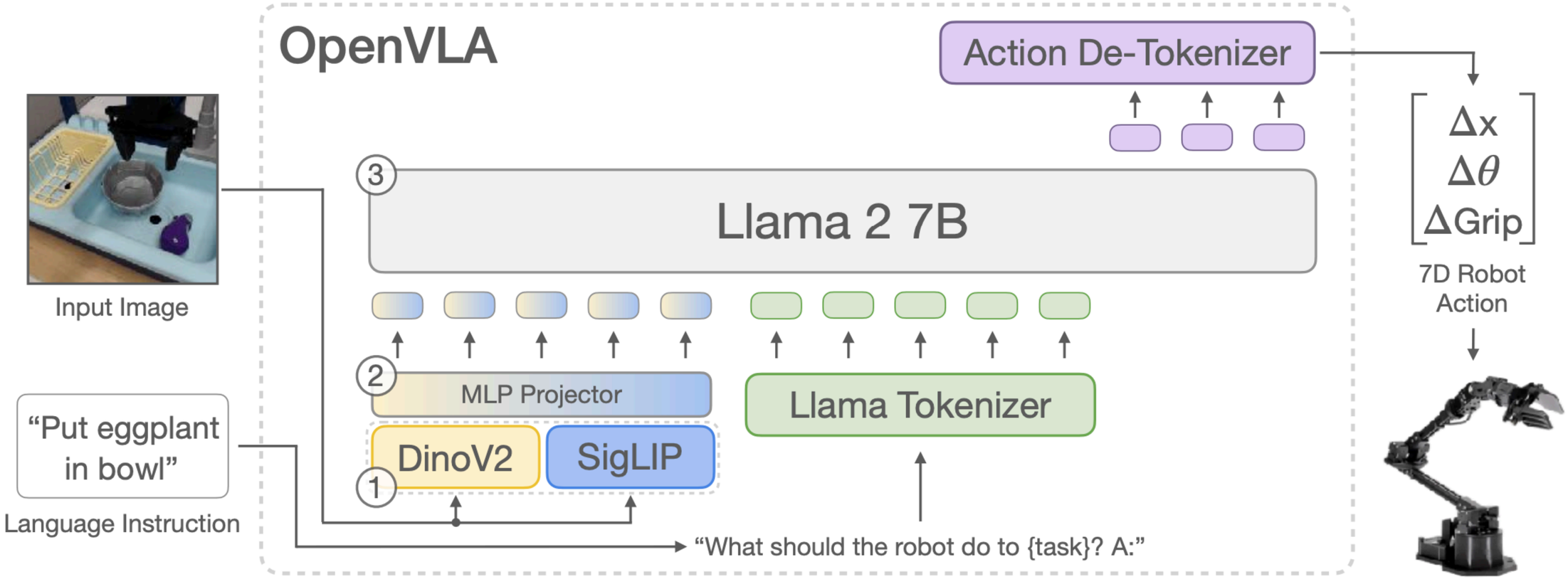
What is OpenVLA trained on?

- 970k episodes from Open X-Embodiment dataset
  - 27 real robot datasets
- Trained on 15 more datasets than RT-2-X

OpenVLA Inputs / Outputs?

- Inputs: Prompt w/ single image & language instruction
  - <Image> In: What action should the robot take to put eggplant into pot? Out:
- Outputs: Tokenized robot action

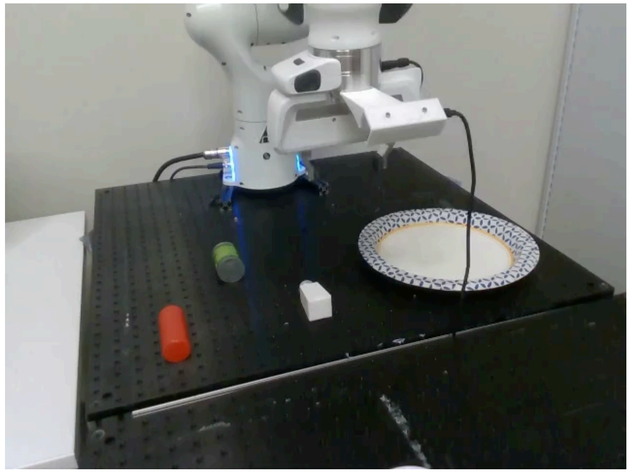
A close look at OpenVLA



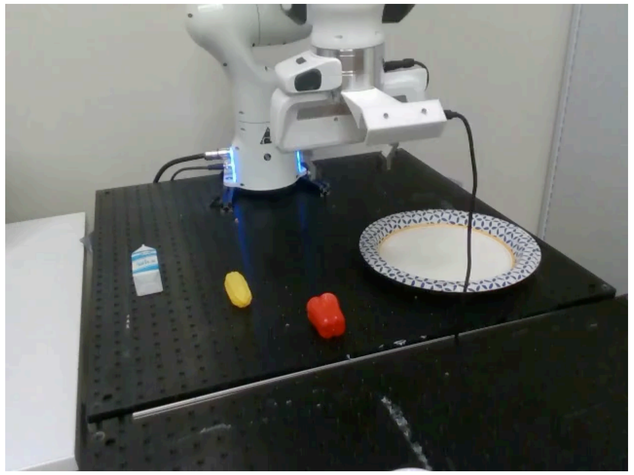
OpenVLA Robot Action Tokenization

- Robot action space: 7 dimensions
  - 6-DoF delta end-effector pose: $\Delta pos_x, \Delta pos_y, \Delta pos_z, \Delta rot_x, \Delta rot_y, \Delta rot_z$
  - 1-DoF gripper control: $\Delta gripper$ (binary: 0 = close, 1 = open)
- Each dimension is scaled to [-1, +1], then discretized into 255 uniform bins
  - $\Delta pos_x \rightarrow$ -1 [1 | 2 | . . . | 254 | 255] +1
  - $\Delta pos_y \rightarrow$ -1 [1 | 2 | . . . | 254 | 255] +1
  - ...
- Therefore, each action $\hat{a}_t$ can be represented by a string of 7 tokens
  - Example:
    - Raw action: [0.00 0.03 -0.82 0.00 -0.14 0.57 1.00]
    - Tokenized: " 128 132 14 128 110 201 255 "
- Only need 255 tokens to represent the entire action space!
- In practice: We override the 255 least frequently used tokens in vocab

OpenVLA results



Move Salt Shaker to plate



Move Corn onto plate

Pour corn to bowl

Lift upright the coke can

Put corn onto plate