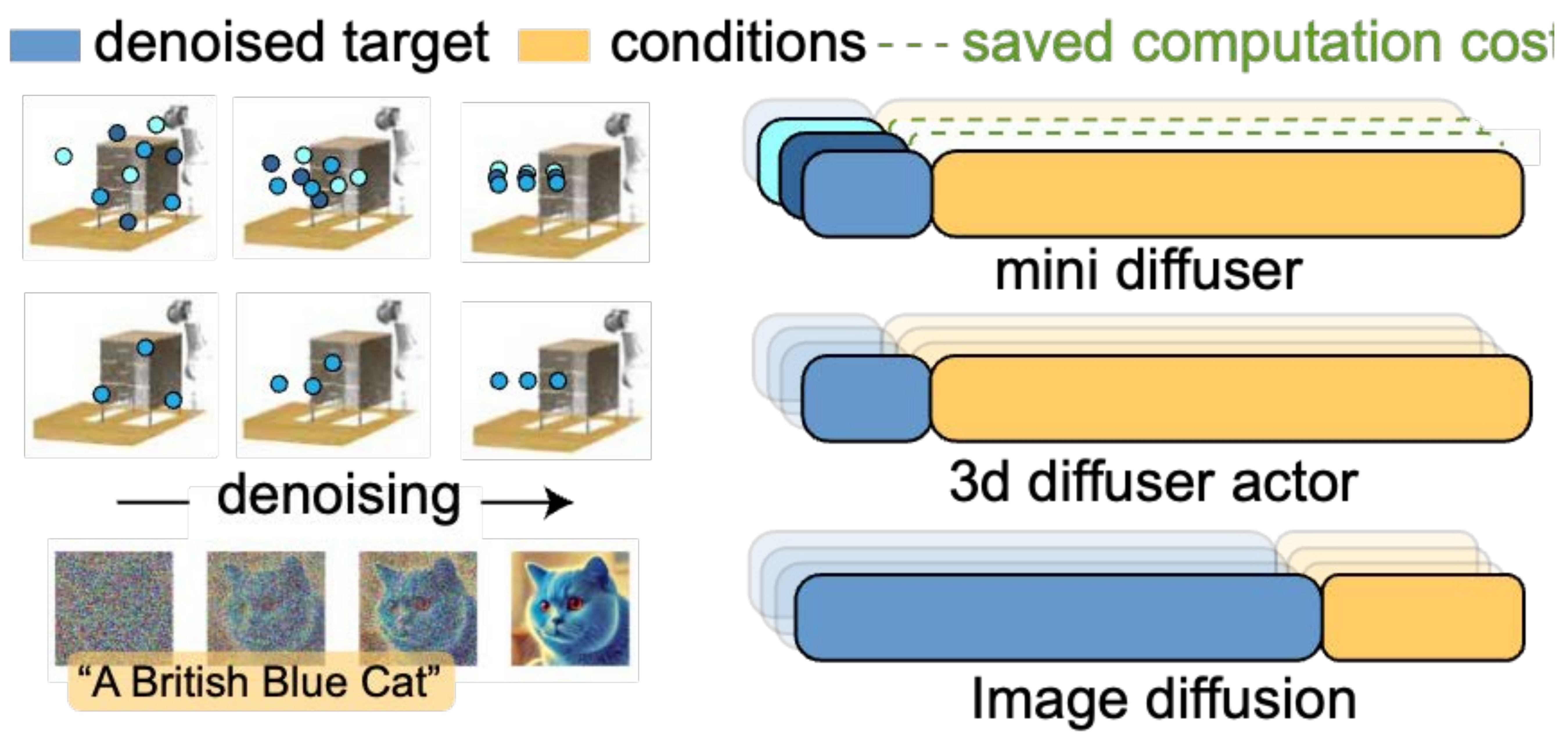
# Sample-efficient Manipulation with Equivariant Models and Fast Training

Renaud Detry

KU LEUVEN

ELLIIT – Nov 18, 2025

# Mini Diffuser: Fast Multi-task Diffusion Policy Training Using Two-level Mini-batches



*"Reduces by an order of magnitude the time and memory needed to train multi-task vision-language robotic diffusion policies."*

**Yutong Hu**, Pinhao Song, Kehan Wen, Renaud Detry

# Image vs robot diffusion: same model, different target

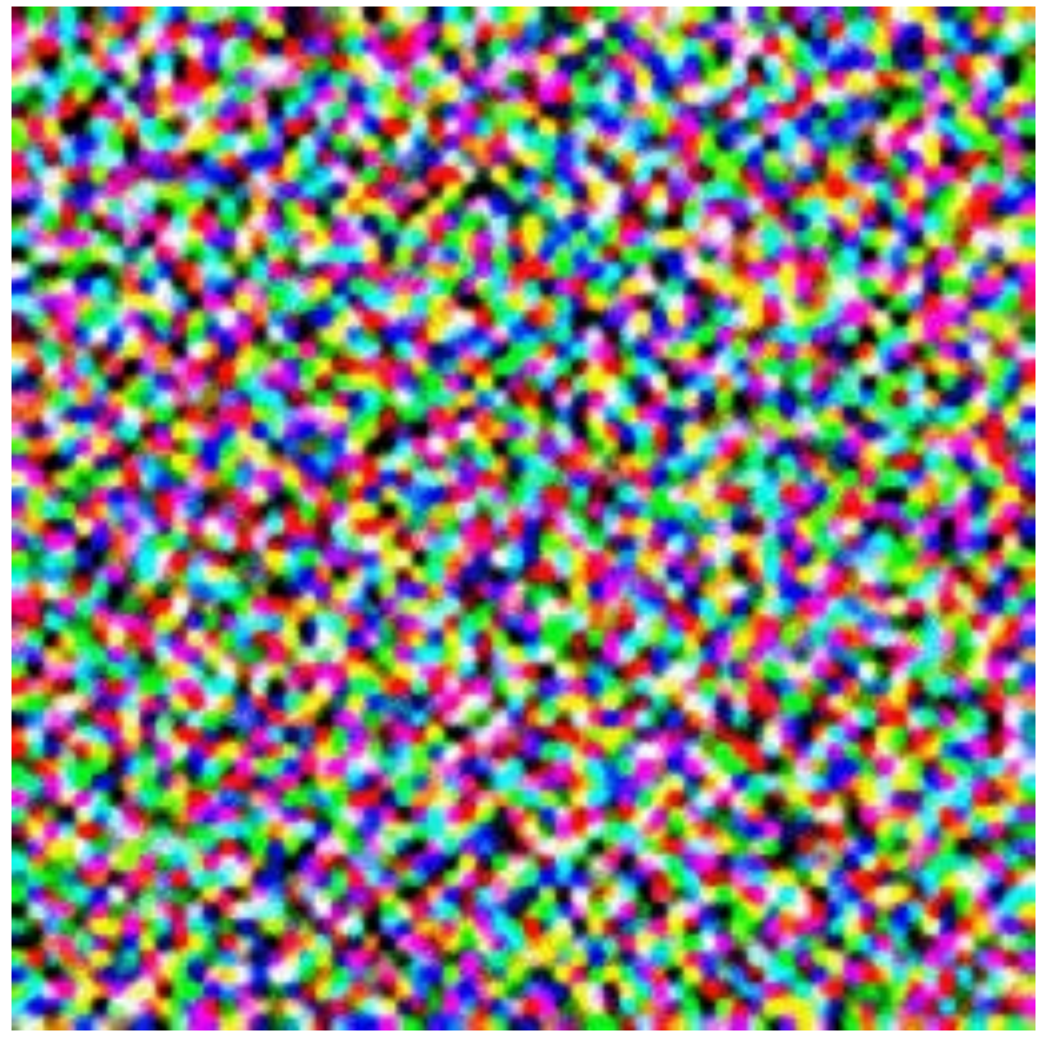
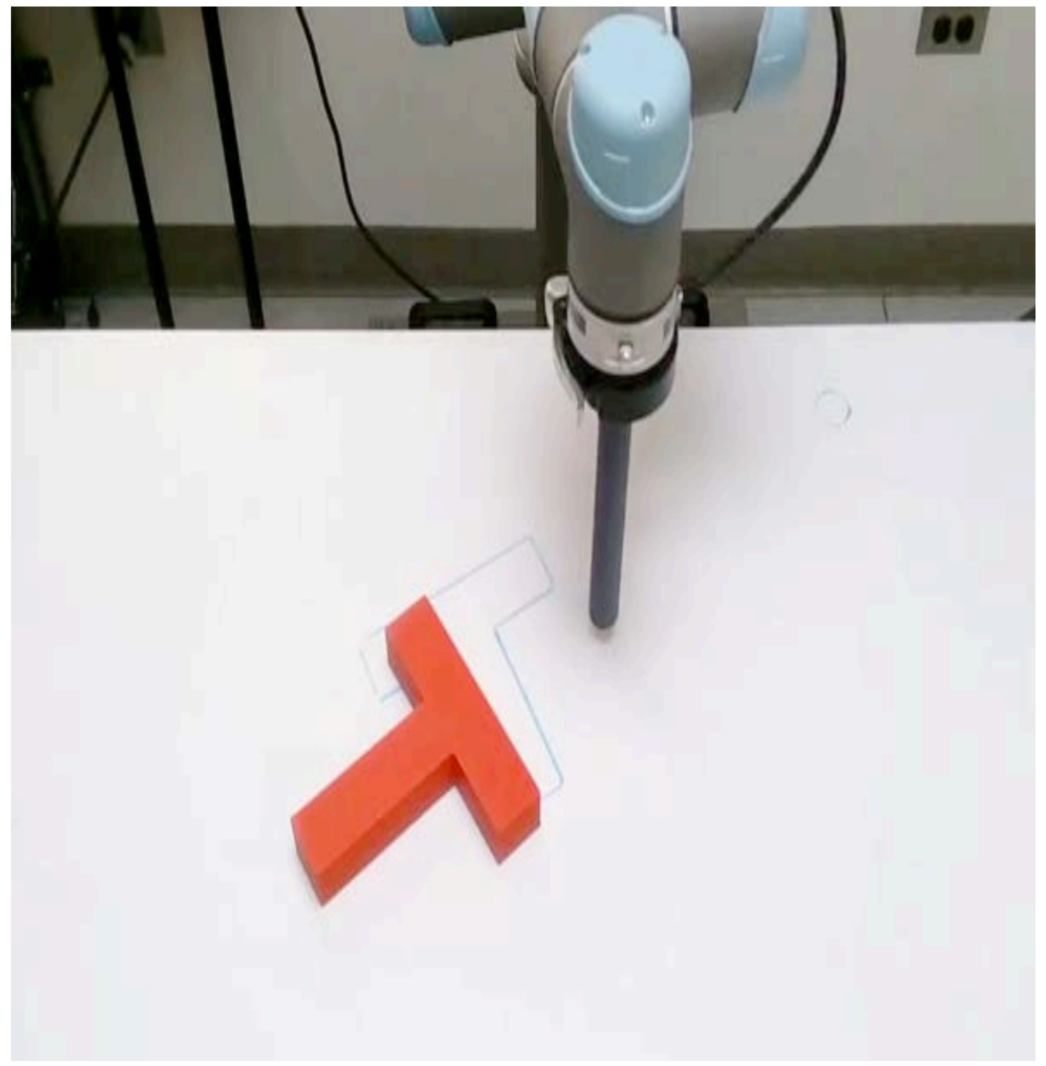


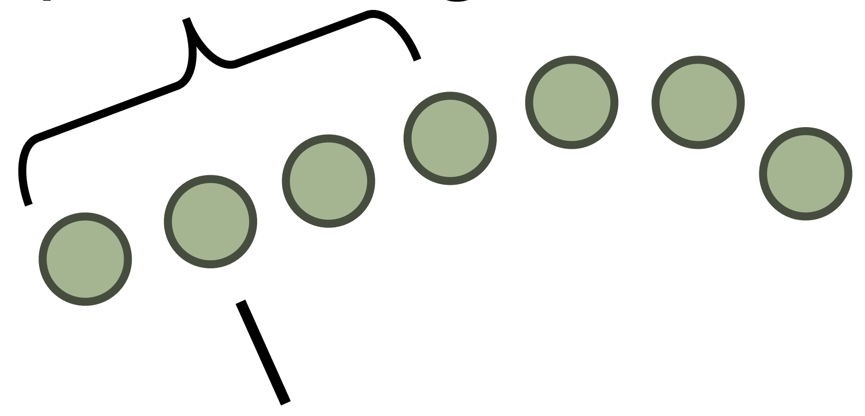
Image Diffusion [1]



Policy Diffusion [2]

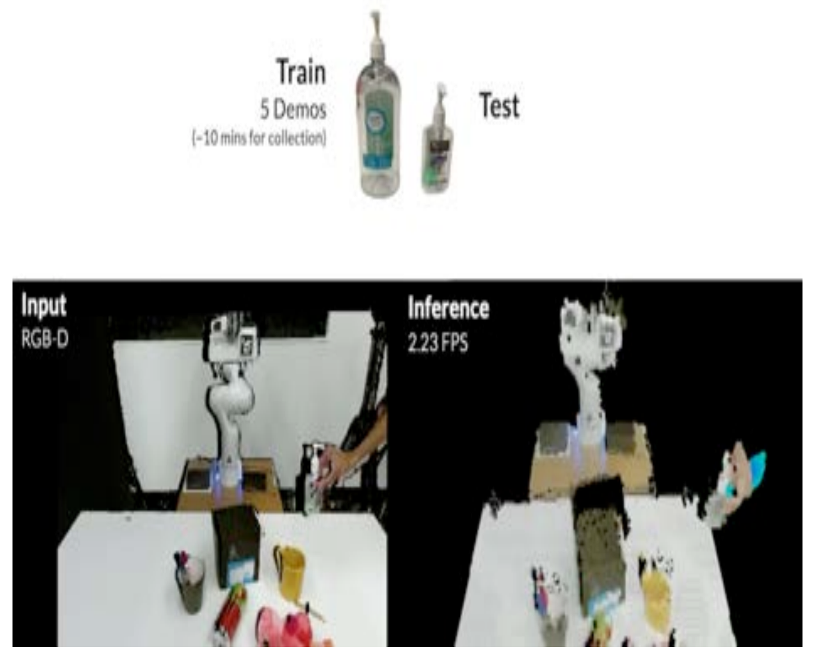
Rolling window prediction and control

Sequence length $= l$



$l = 1$, only predict next pose

Denoising process for action $\boldsymbol{a}(\boldsymbol{s})$



$\boldsymbol{a}$ given different $\boldsymbol{s}$ [3]

[1] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models."

[2] Chi, Cheng, et al. "Diffusion policy: Visuomotor policy learning via action diffusion."

[3] Shridhar, Mohit, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation."

# Image vs robot diffusion: same model, different target

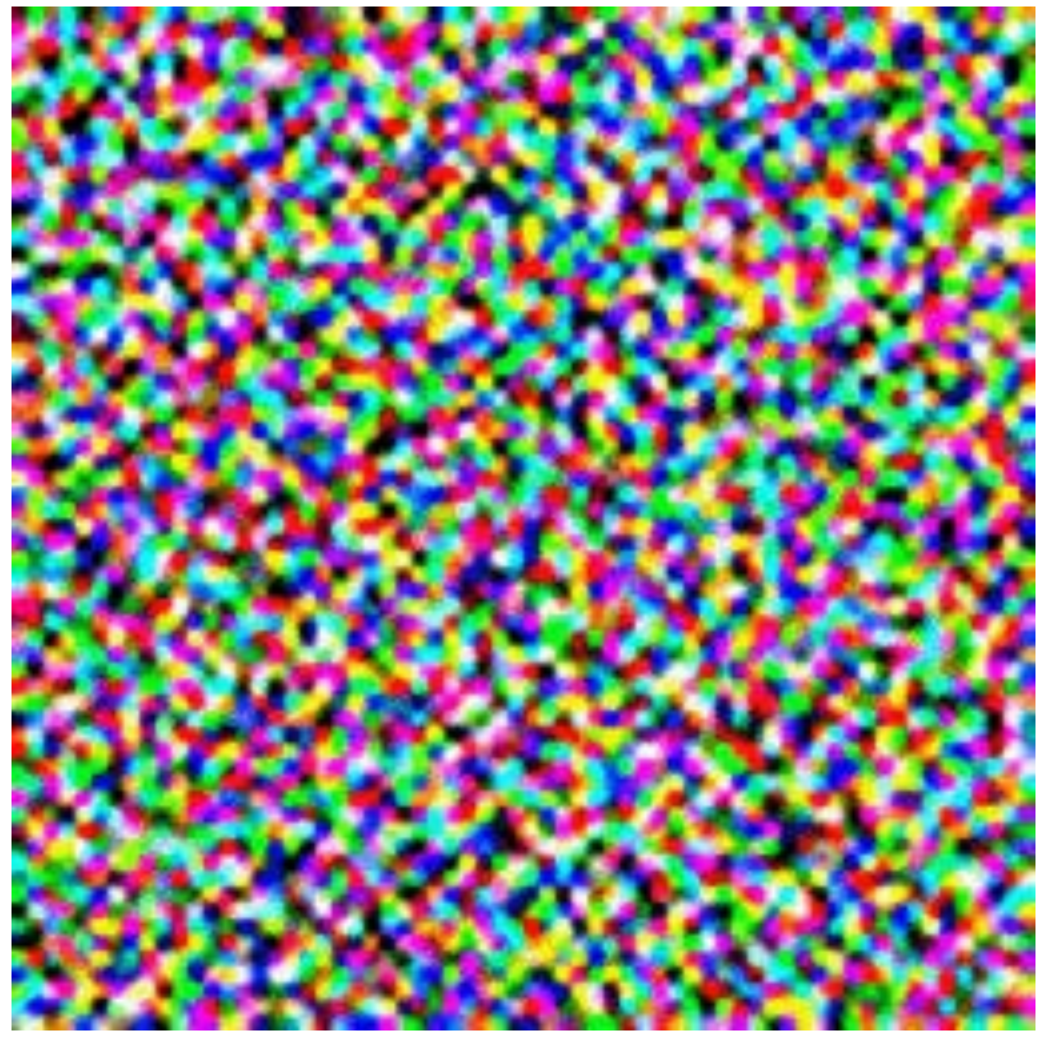
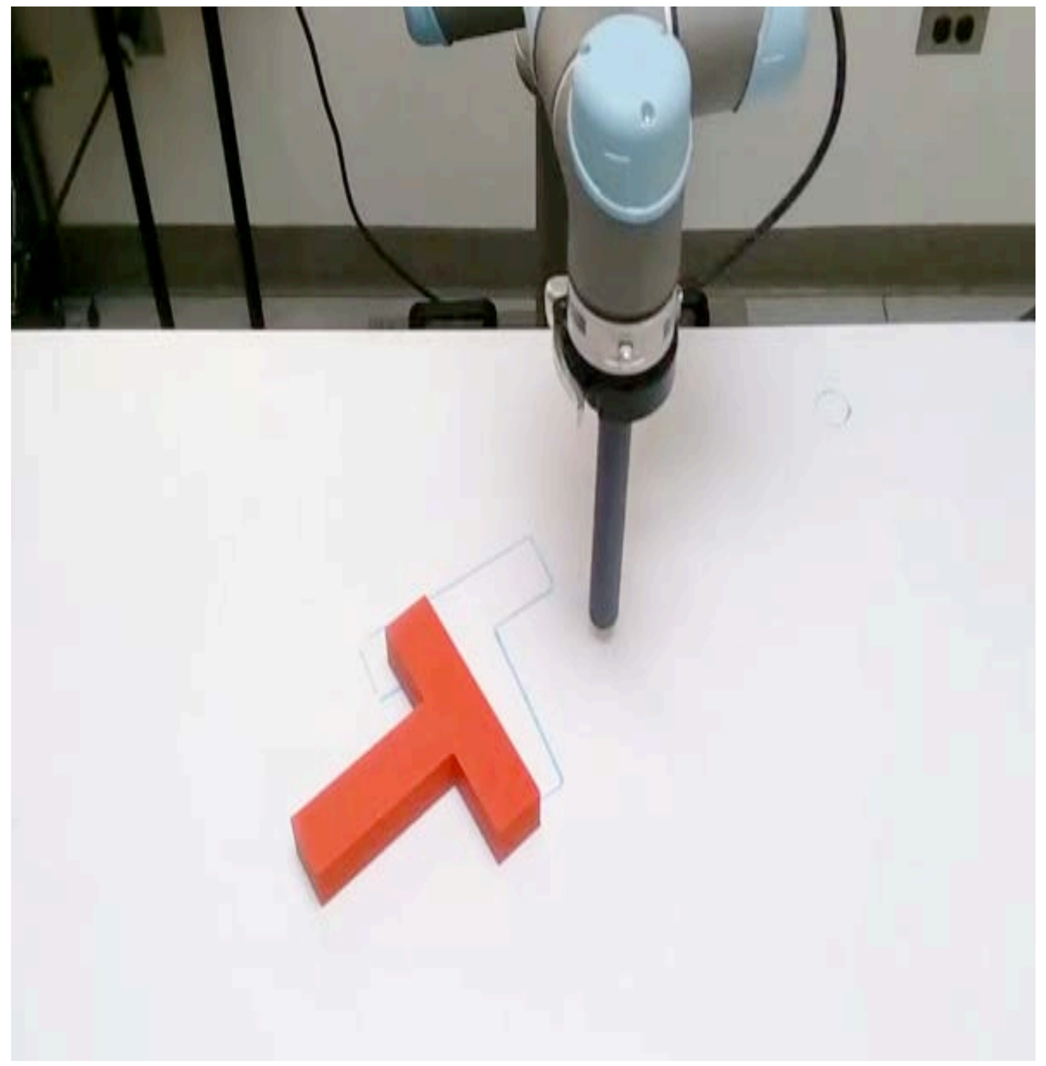
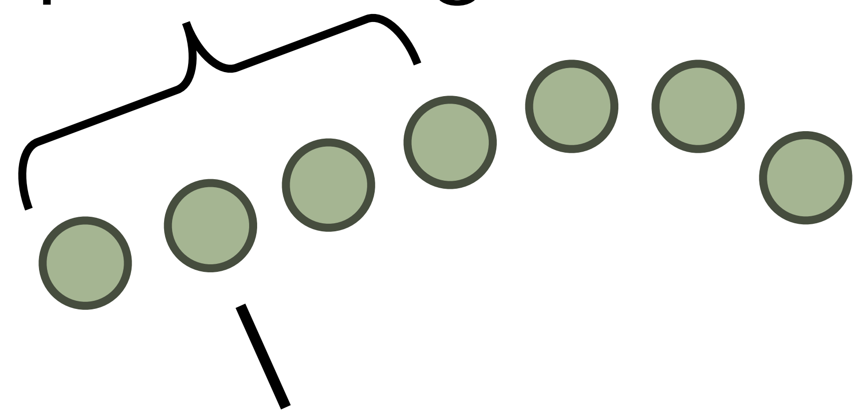


Image Diffusion [1]


Policy Diffusion [2]

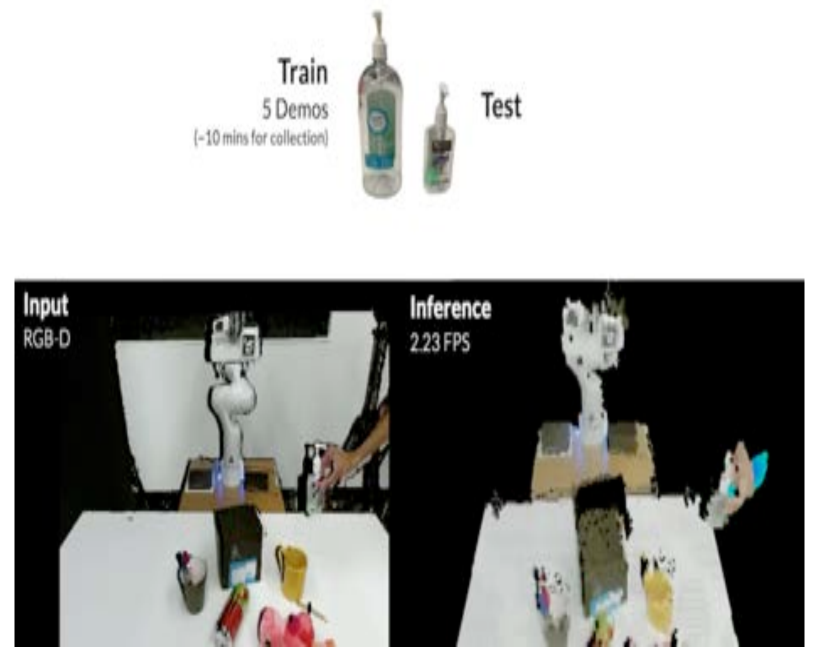
Rolling window prediction and control

Sequence length $= l$



$l = 1$, only predict next pose

Denoising process for action $\boldsymbol{a}(\boldsymbol{s})$



$\boldsymbol{a}$ given different $\boldsymbol{s}$ [3]

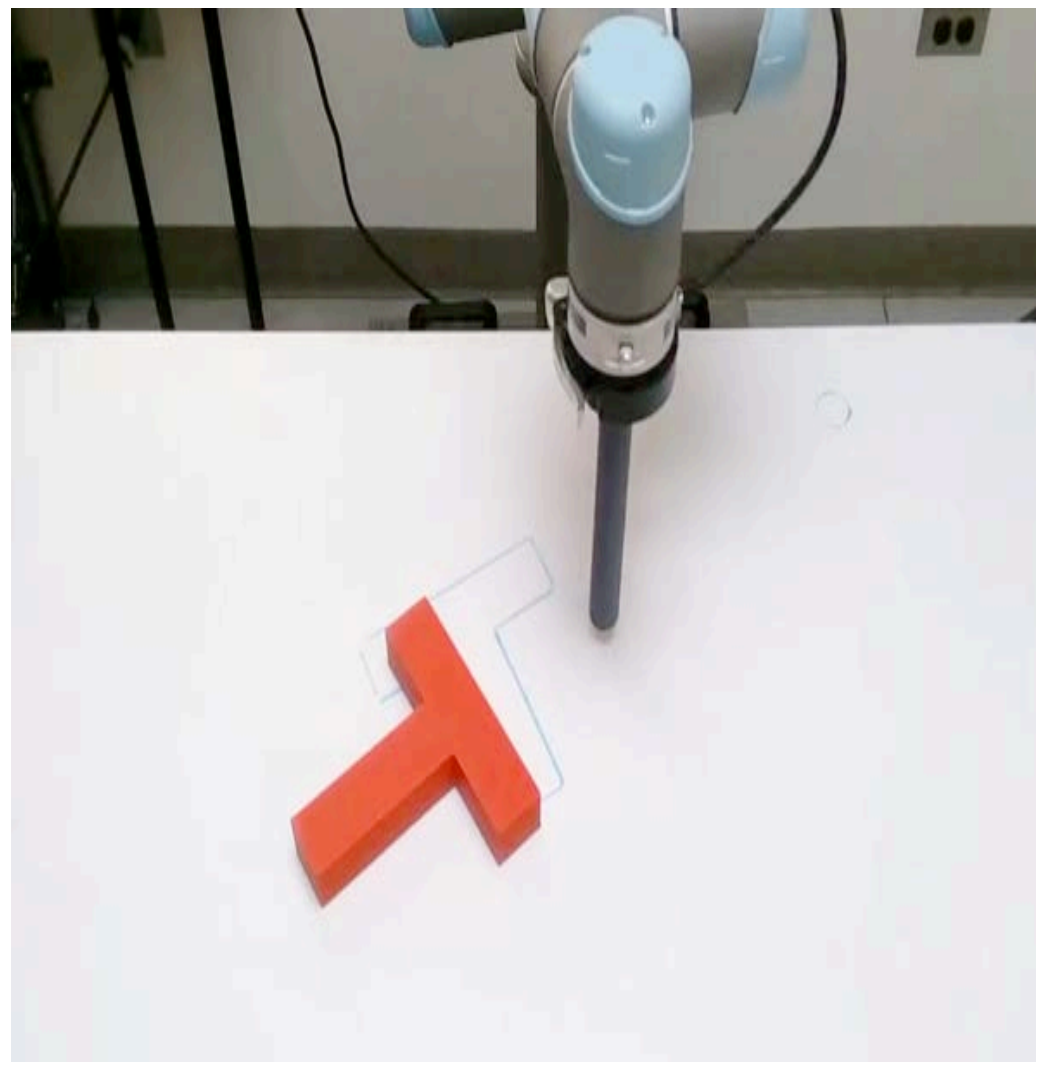
[1] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models."

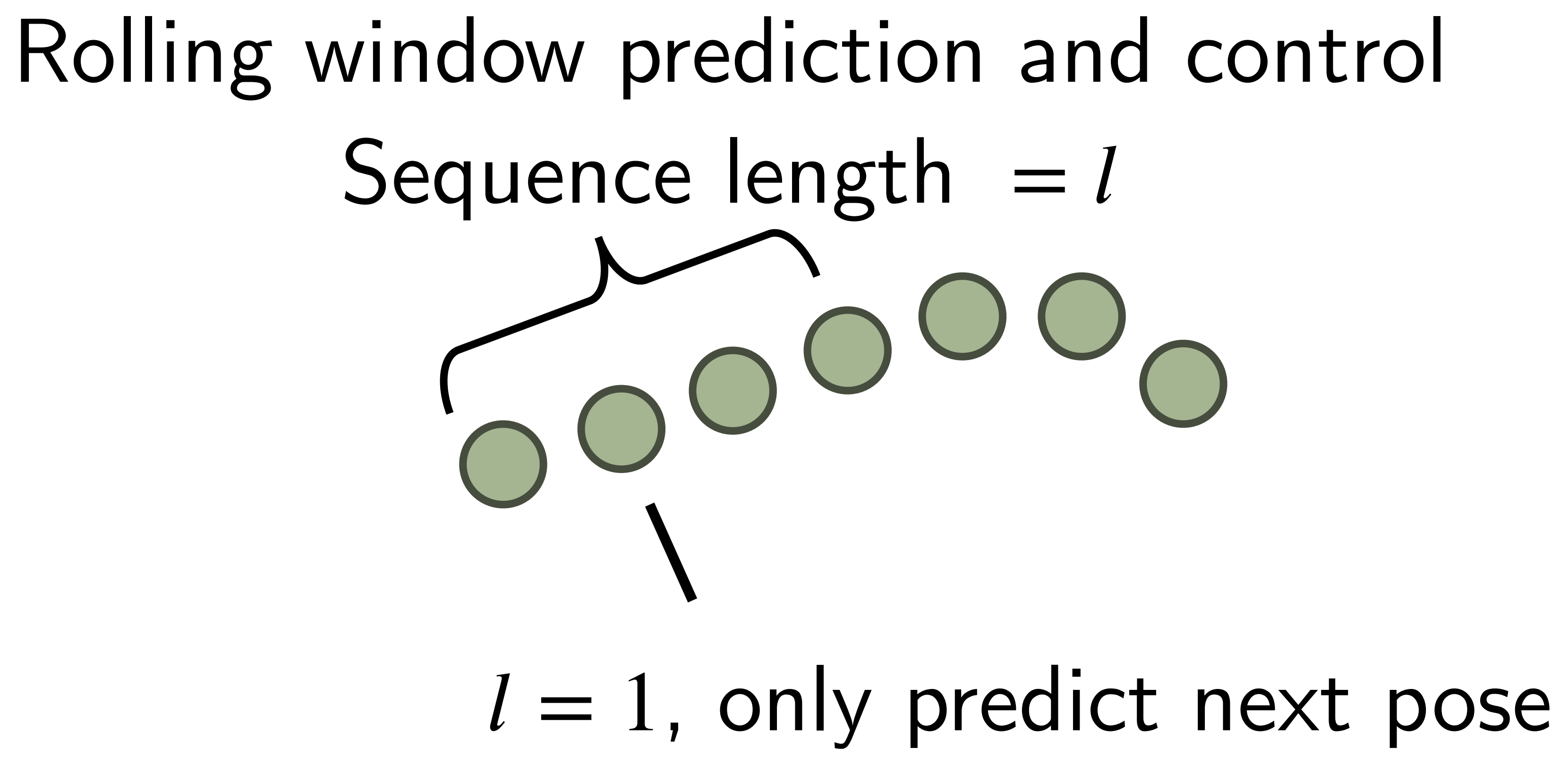
[2] Chi, Cheng, et al. "Diffusion policy: Visuomotor policy learning via action diffusion."

[3] Shridhar, Mohit, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation."

# Image vs robot diffusion: same model, different target

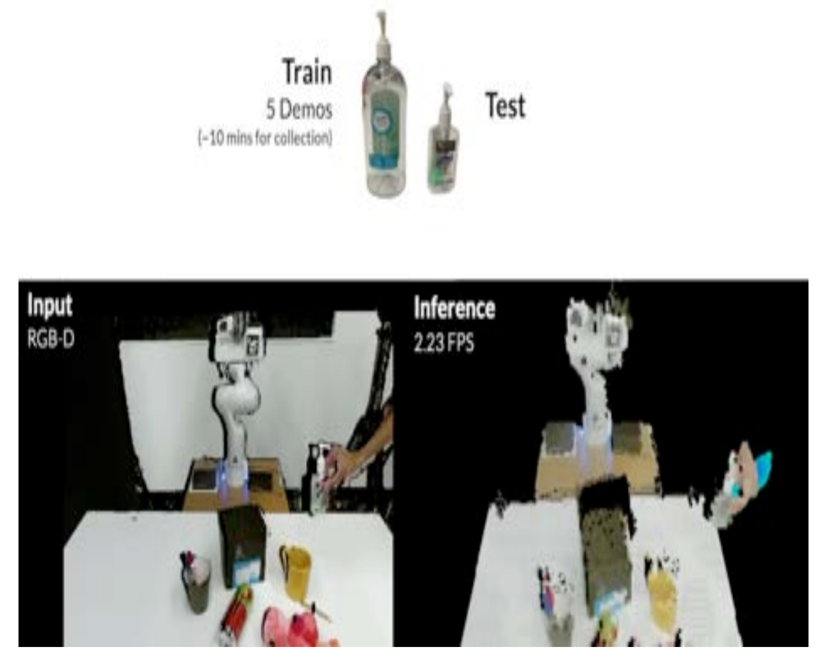
Image Diffusion [1]


Policy Diffusion [2]



Rolling window prediction and control
Sequence length $= l$
$l = 1$, only predict next pose

Denoising process for action $\boldsymbol{a}(\boldsymbol{s})$



$\boldsymbol{a}$ given different $\boldsymbol{s}$ [3]

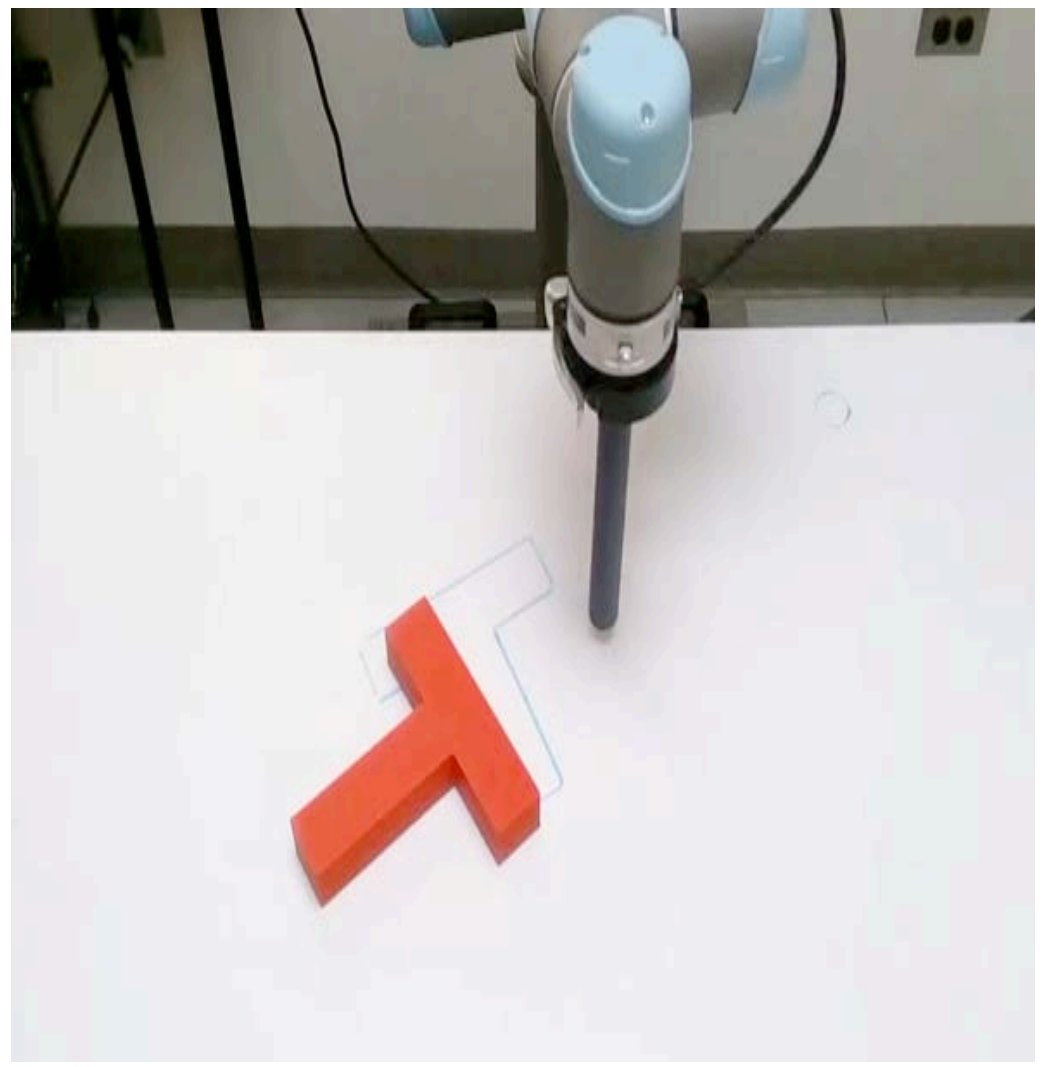
[1] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models."

[2] Chi, Cheng, et al. "Diffusion policy: Visuomotor policy learning via action diffusion."

[3] Shridhar, Mohit, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation."

# Image vs robot diffusion: same model, different target

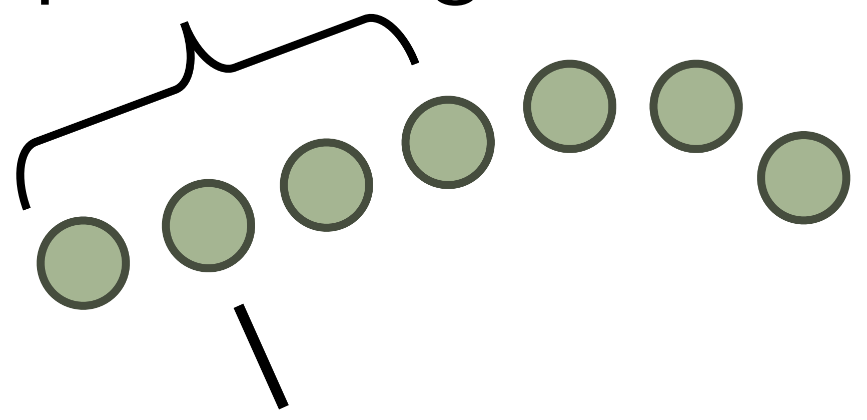
Image Diffusion [1]



Policy Diffusion [2]

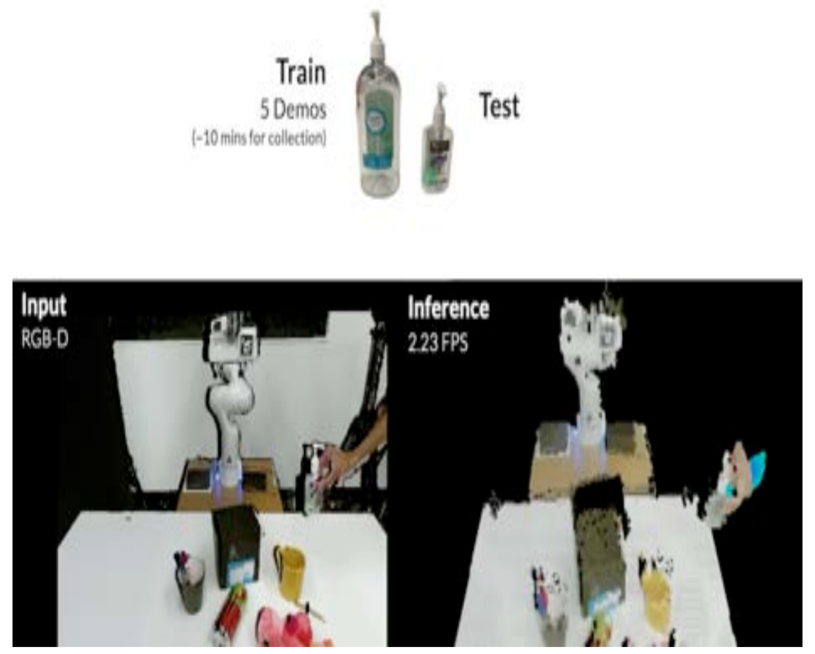
Rolling window prediction and control

Sequence length $= l$



$l = 1$, only predict next pose

Denoising process
for action $\boldsymbol{a}(\boldsymbol{s})$



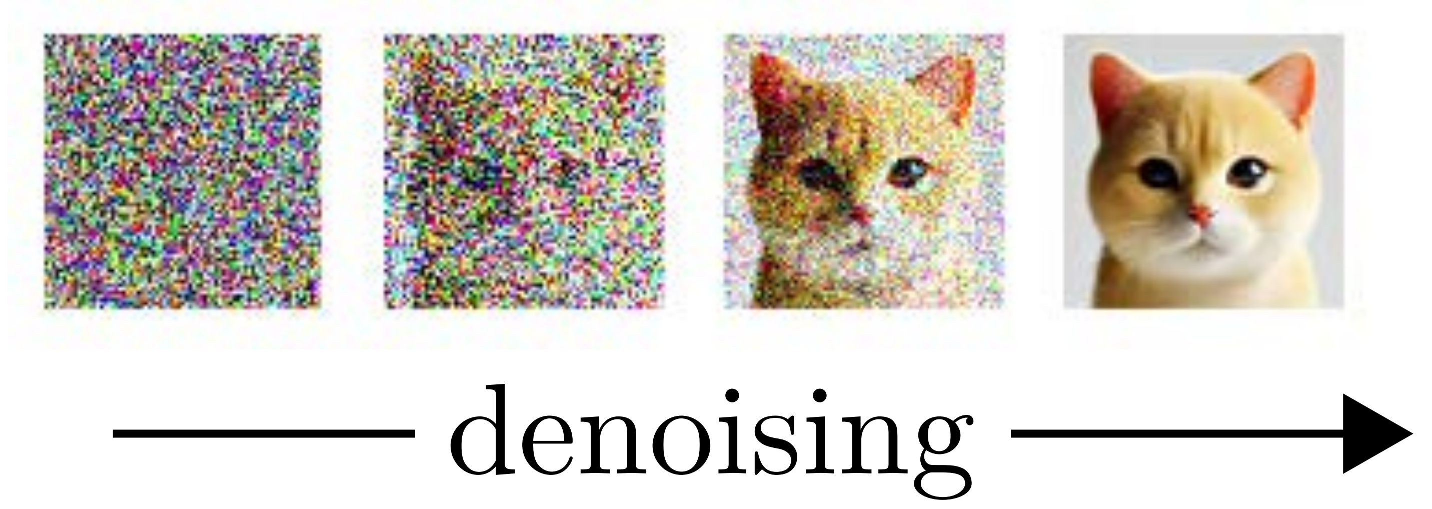
$\boldsymbol{a}$ given
different $\boldsymbol{s}$ [3]

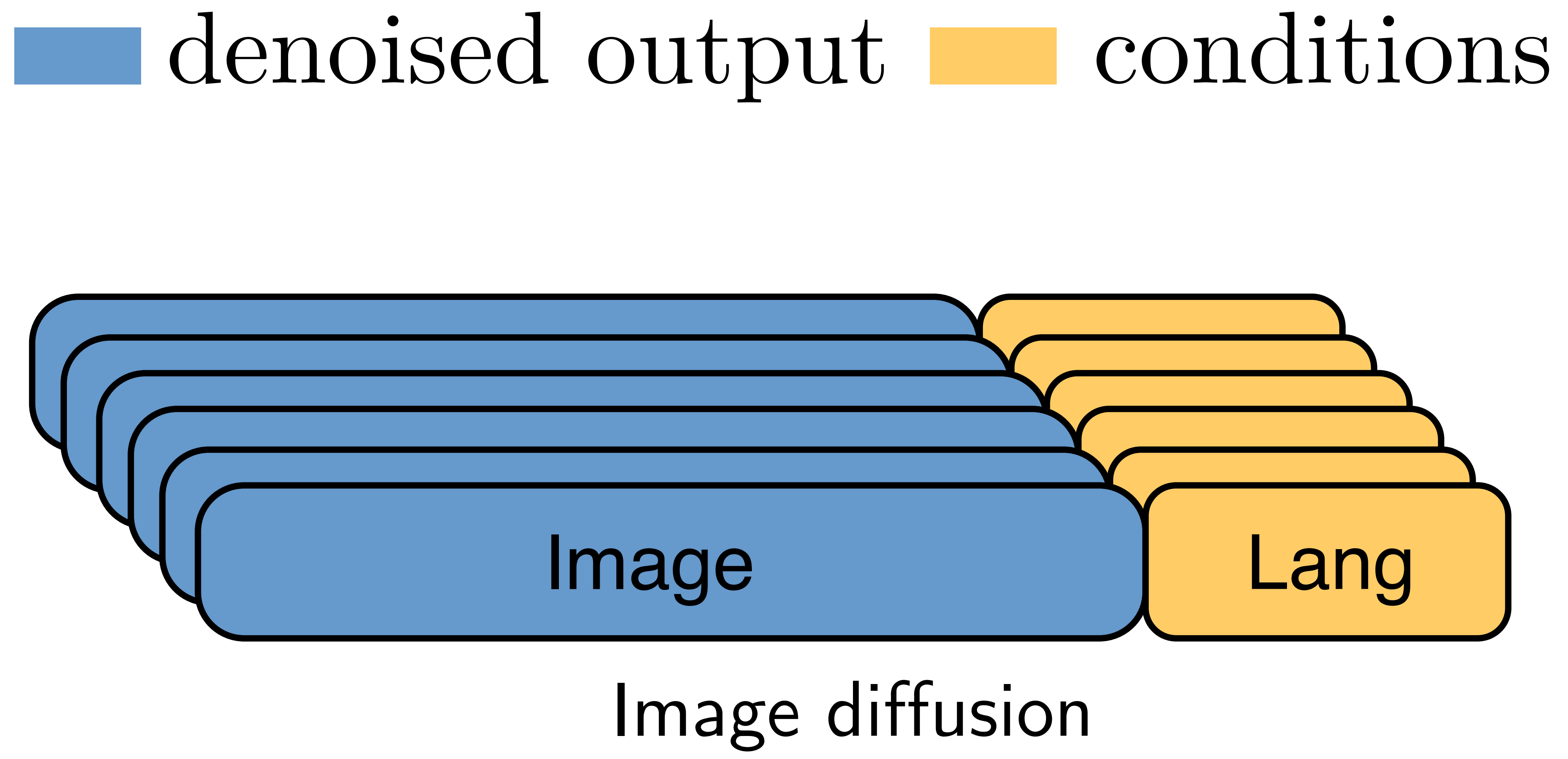
[1] Rombach, Robin, et al. "High-resolution image synthesis with latent diffusion models."

[2] Chi, Cheng, et al. "Diffusion policy: Visuomotor policy learning via action diffusion."

[3] Shridhar, Mohit, Lucas Manuelli, and Dieter Fox. "Perceiver-actor: A multi-task transformer for robotic manipulation."

# Batched Training in Diffusion Models





# Batched Training in Diffusion Models

🟦 denoised output 🟨 conditions

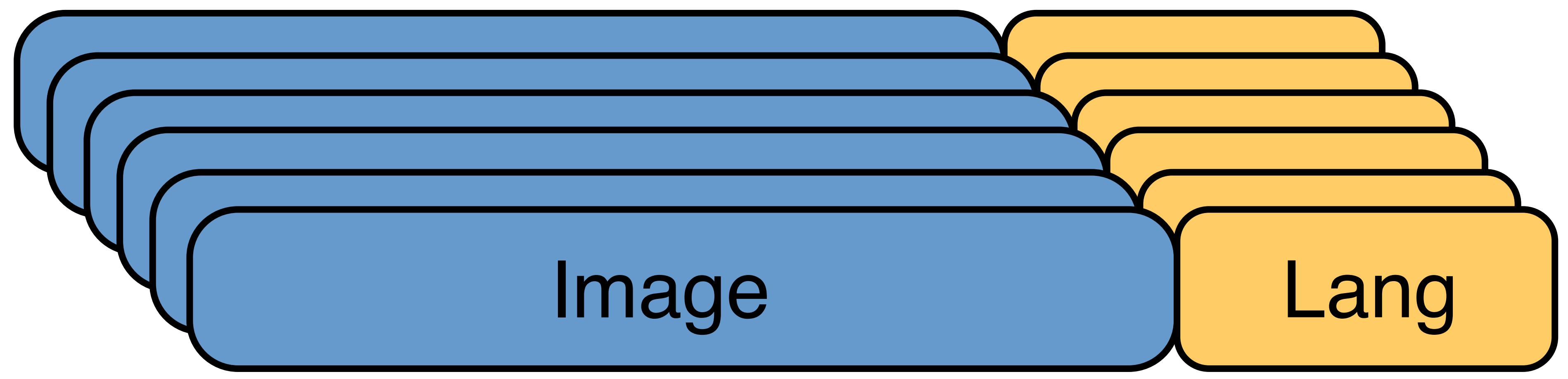
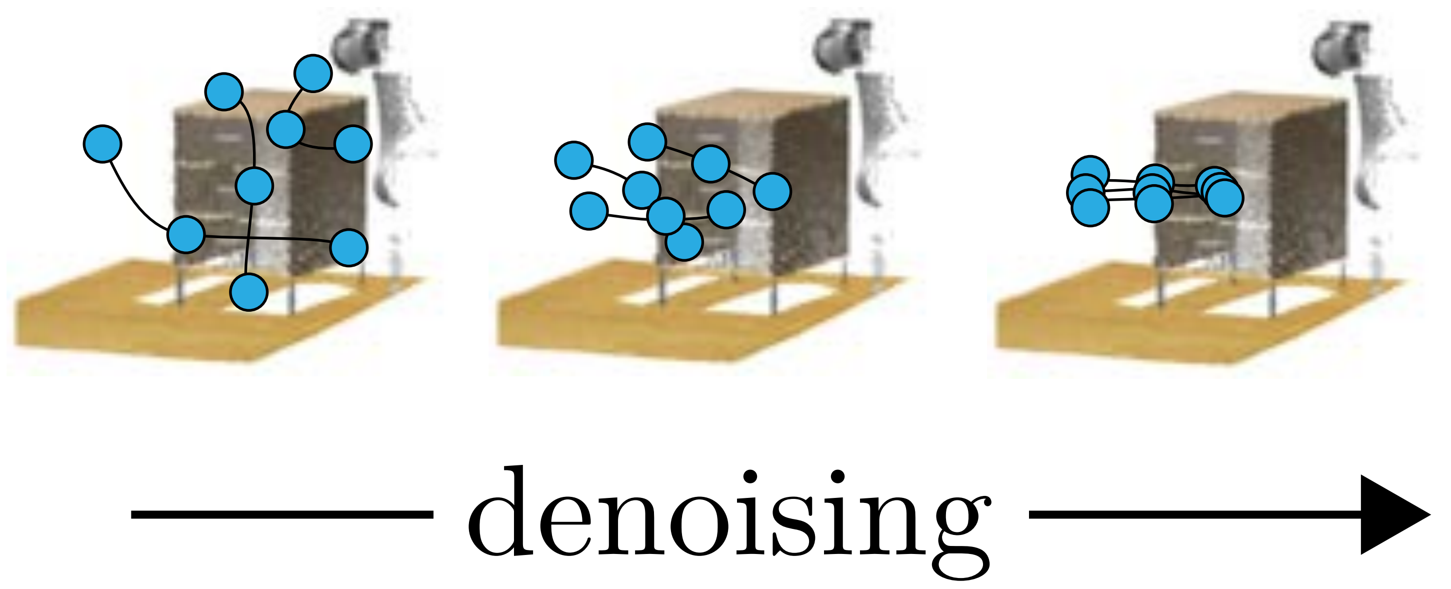
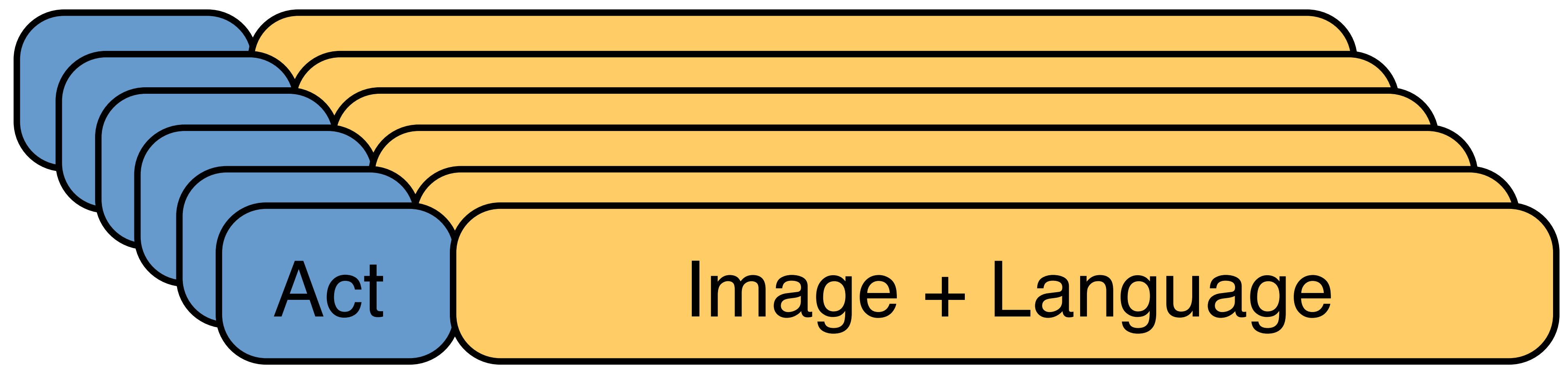


Image diffusion





3D Diffuser Actor

# Batched Training in Diffusion Models

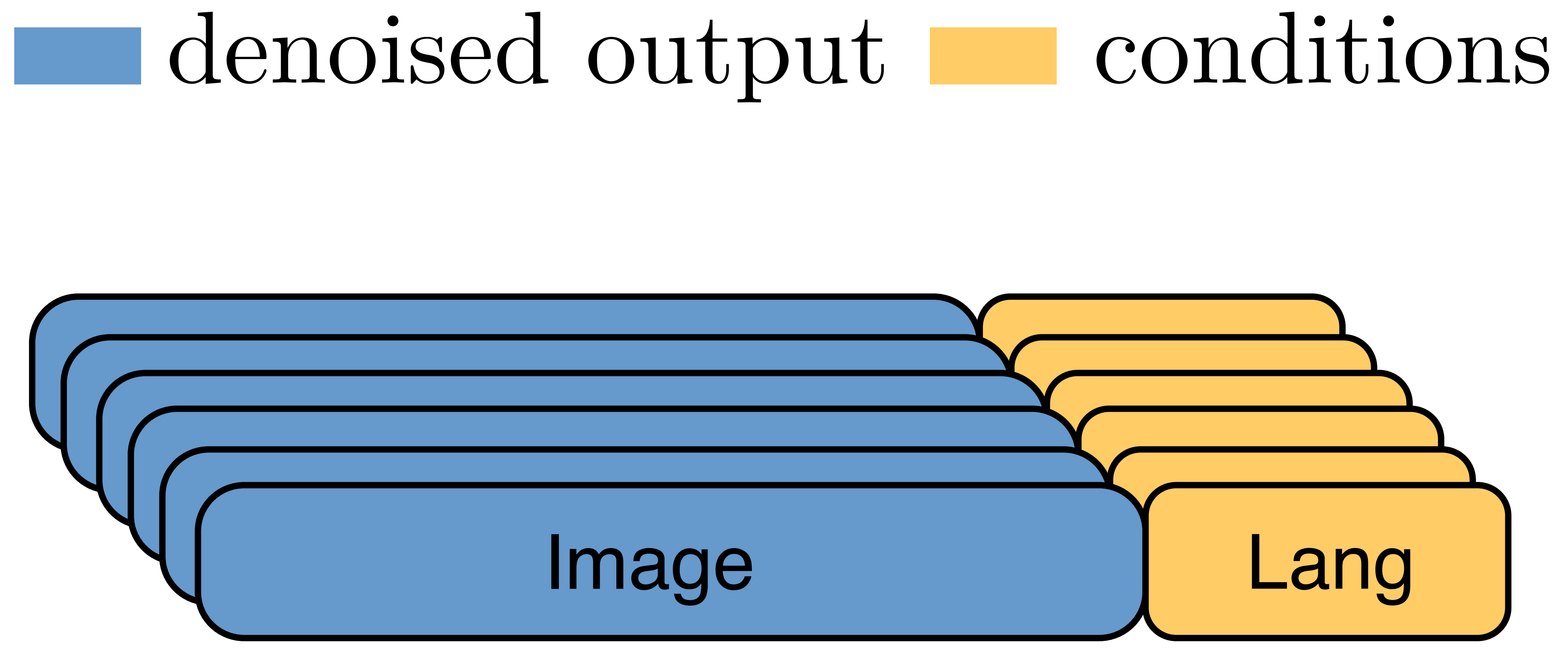
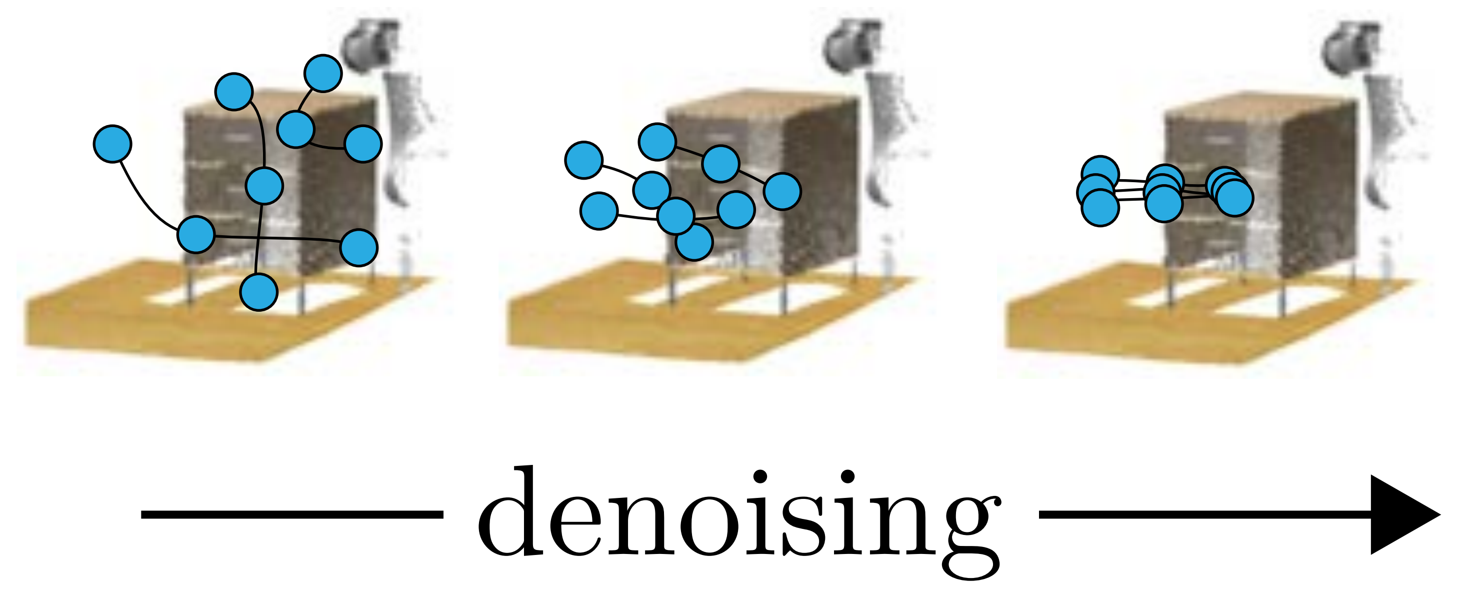
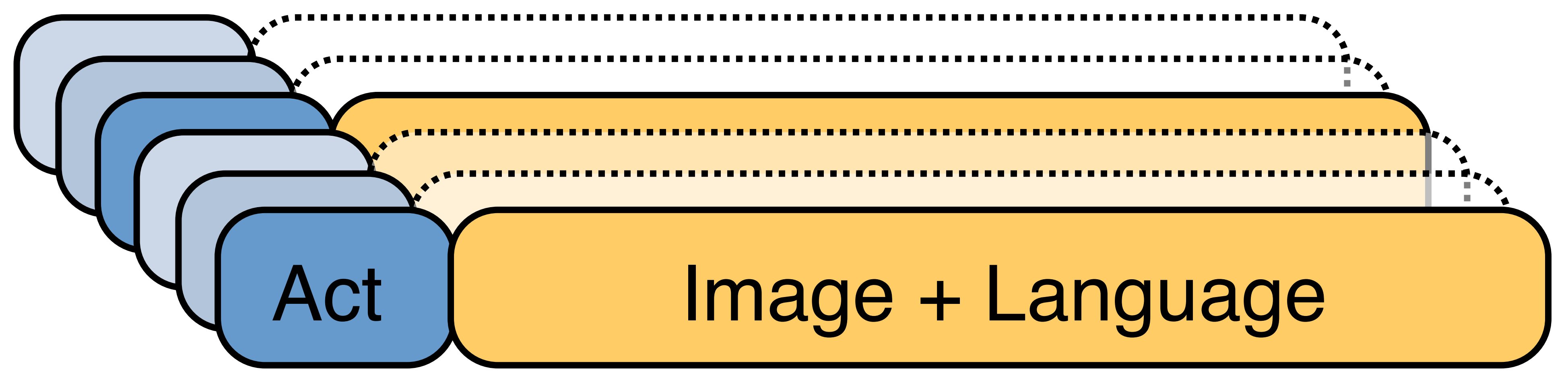


Image diffusion

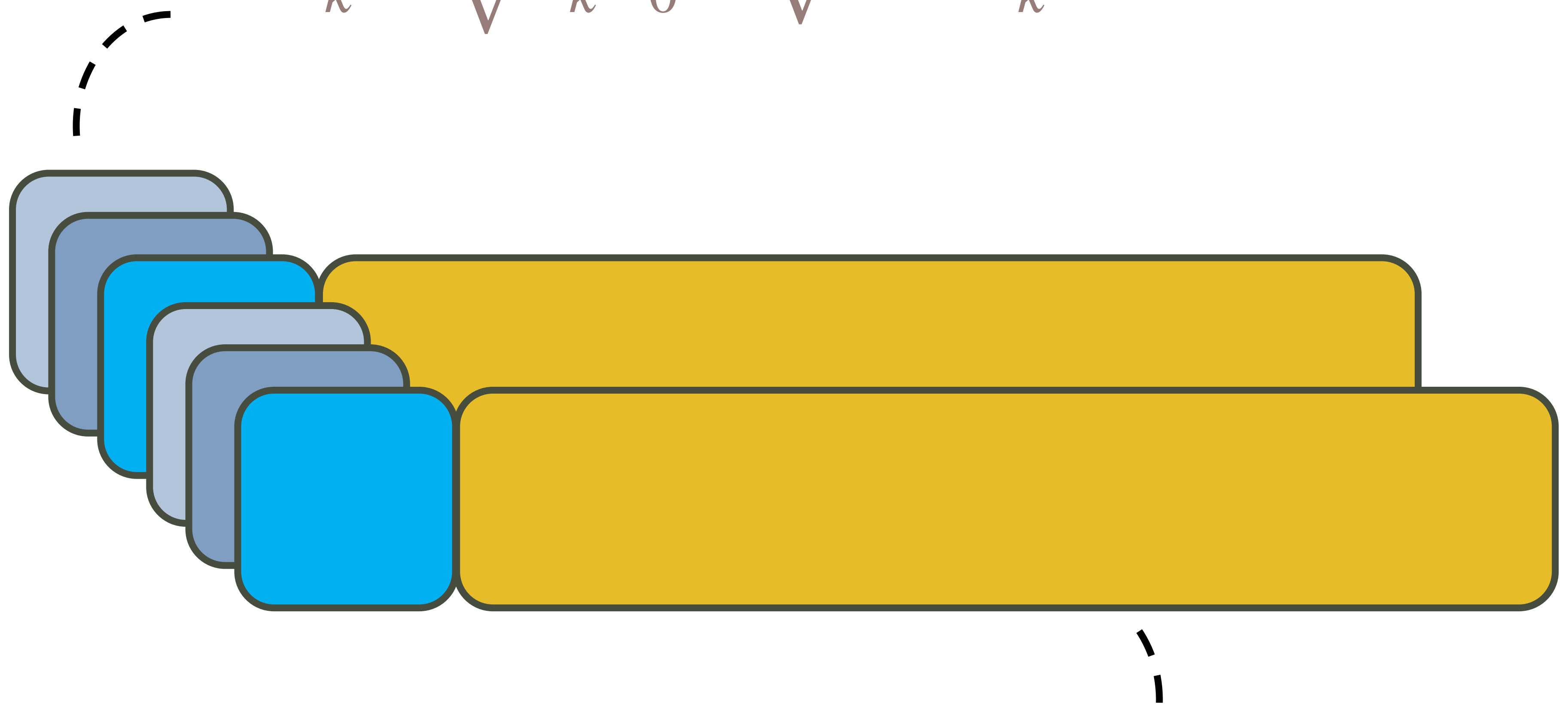




3D Diffuser Actor

# Two-Level Batch for Action Diffusion

$$\boldsymbol{a}_k = \sqrt{\alpha_k}\boldsymbol{a}_0 + \sqrt{1-\alpha_k}\boldsymbol{\epsilon}$$



$\boldsymbol{s}$ point clouds, language instructions, current joint configs ...

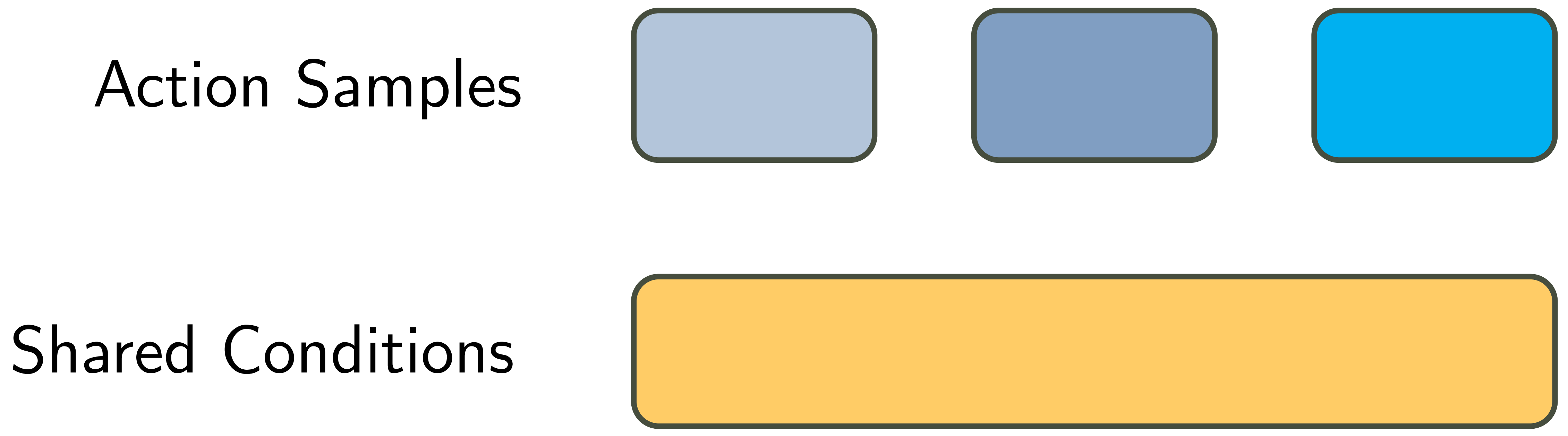
**Level-1:** We first sample $B$ independent state–action pairs

$$\left\{\left(\boldsymbol{s}^{(i)}, \boldsymbol{a}_0^{(i)}\right)\right\}_{i=1}^{B}, \left(\boldsymbol{s}^{(i)}, \boldsymbol{a}_0^{(i)}\right) \sim q(\boldsymbol{a}, s)$$

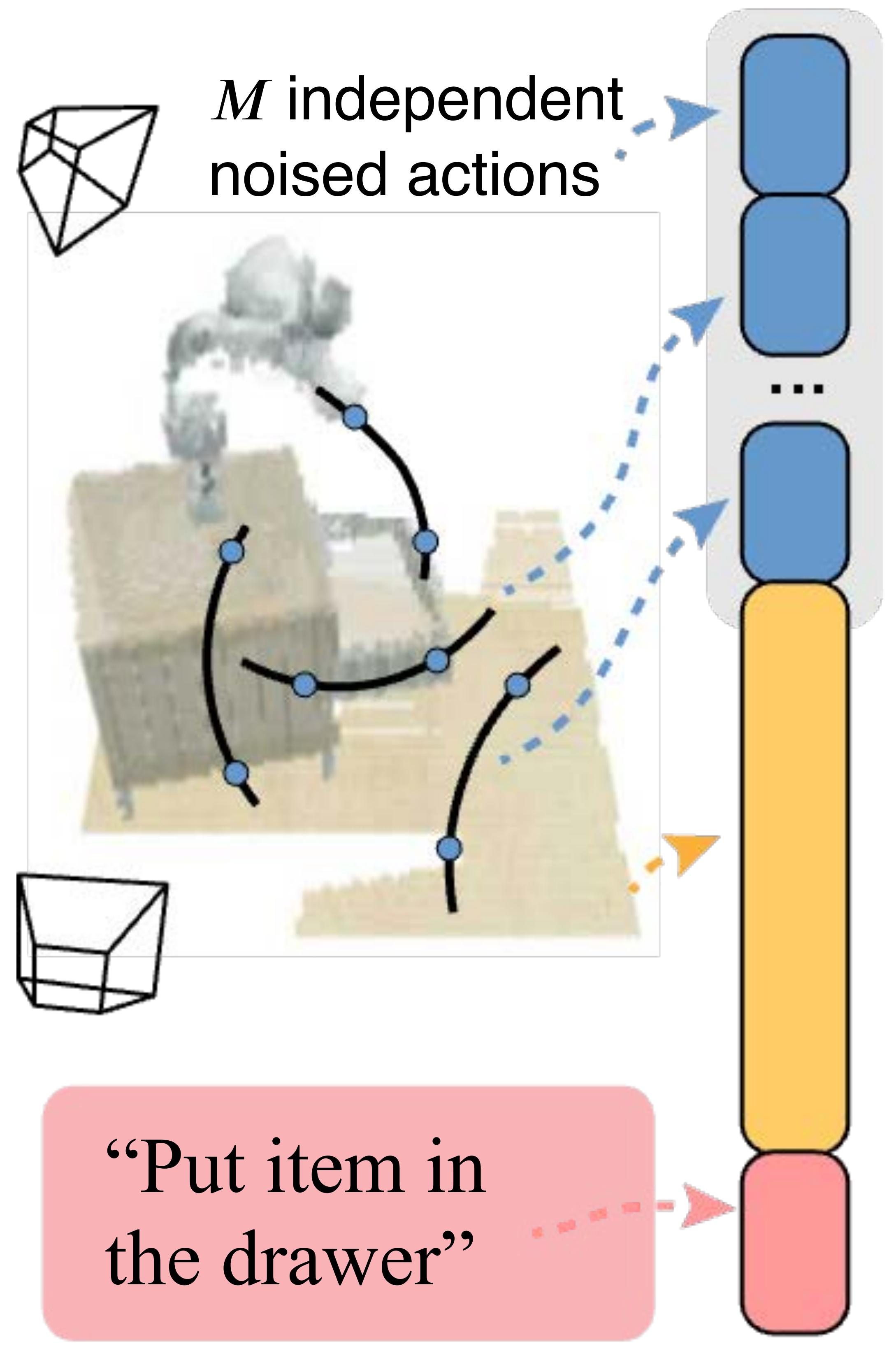
**Level-2:** For each of those $B$, we independently draw $M$ step-noise pairs,

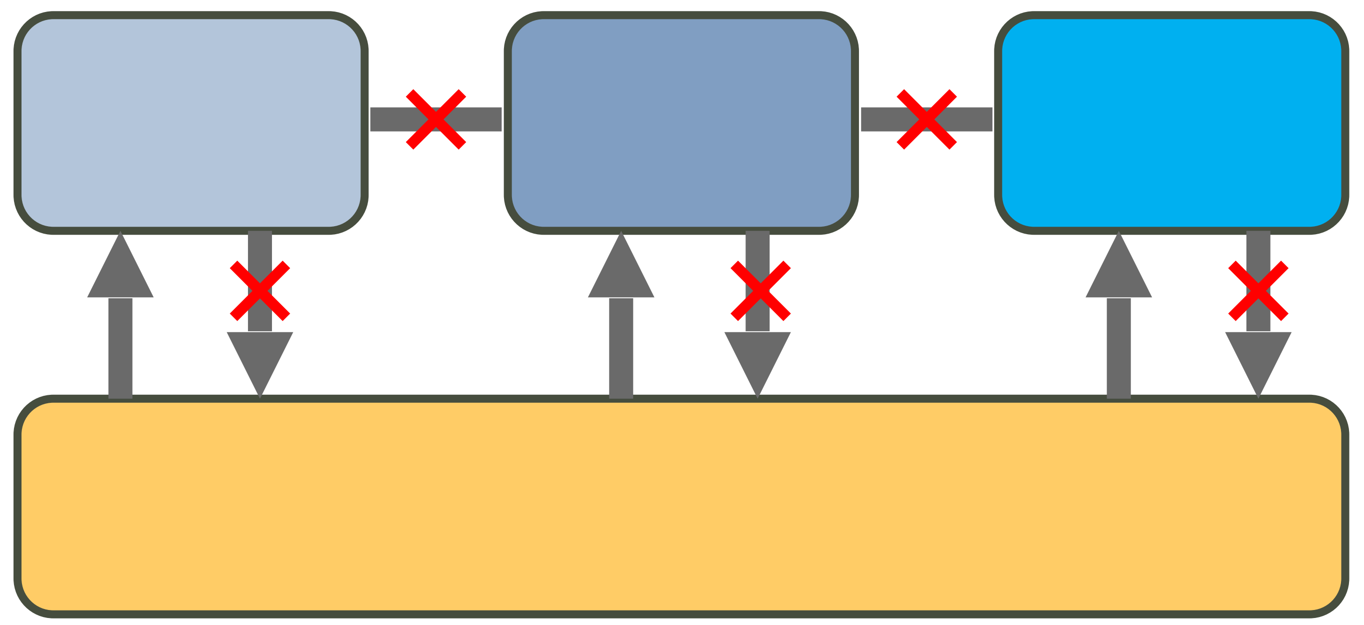
$$\left\{\left(k^{(i,j)}, \boldsymbol{\epsilon}^{(i,j)}\right)\right\}_{j-1}^{M}, k^{(i,j)} \sim \mathcal{U}(1, K), \boldsymbol{\epsilon}^{(i,j)} \sim \mathcal{N}(0, \mathbf{I})\,.$$

# Two-Level Batch for Action Diffusion



# Masked Attention Protects Action Sample Independence

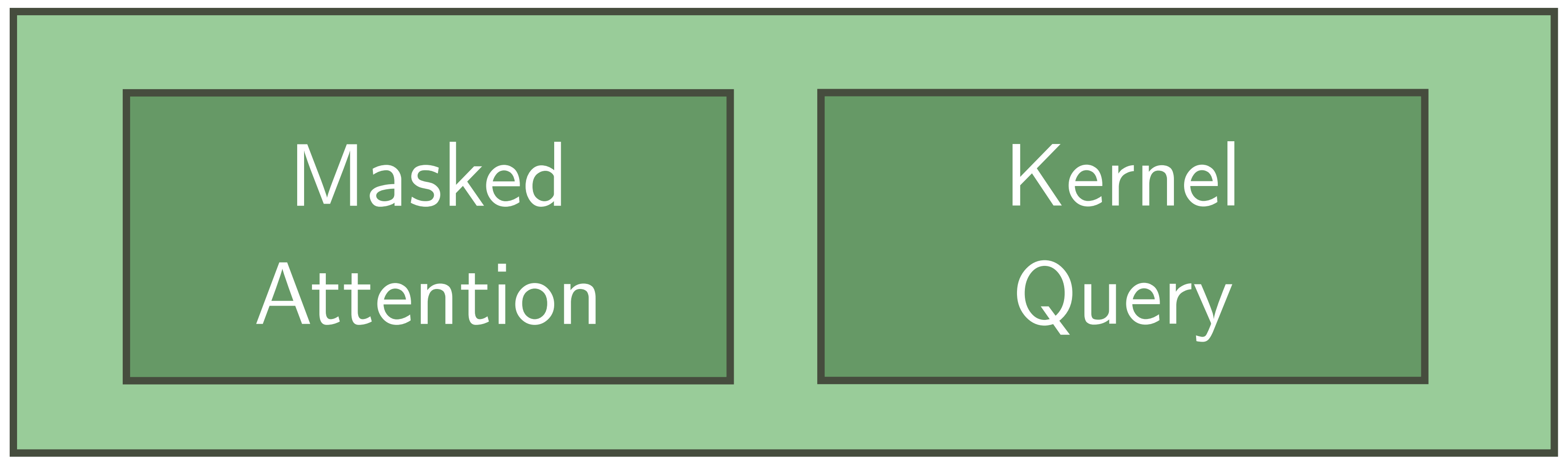


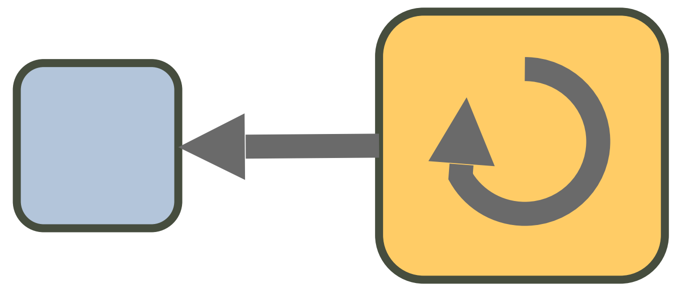


(i) An action sample attends to itself and shared conditions, but not to other action samples

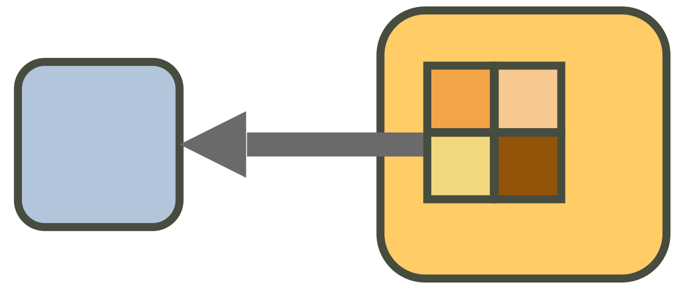
(ii) shared conditions do not attend back to action samples.

Two modules can perform such a "Non-invasive extraction"



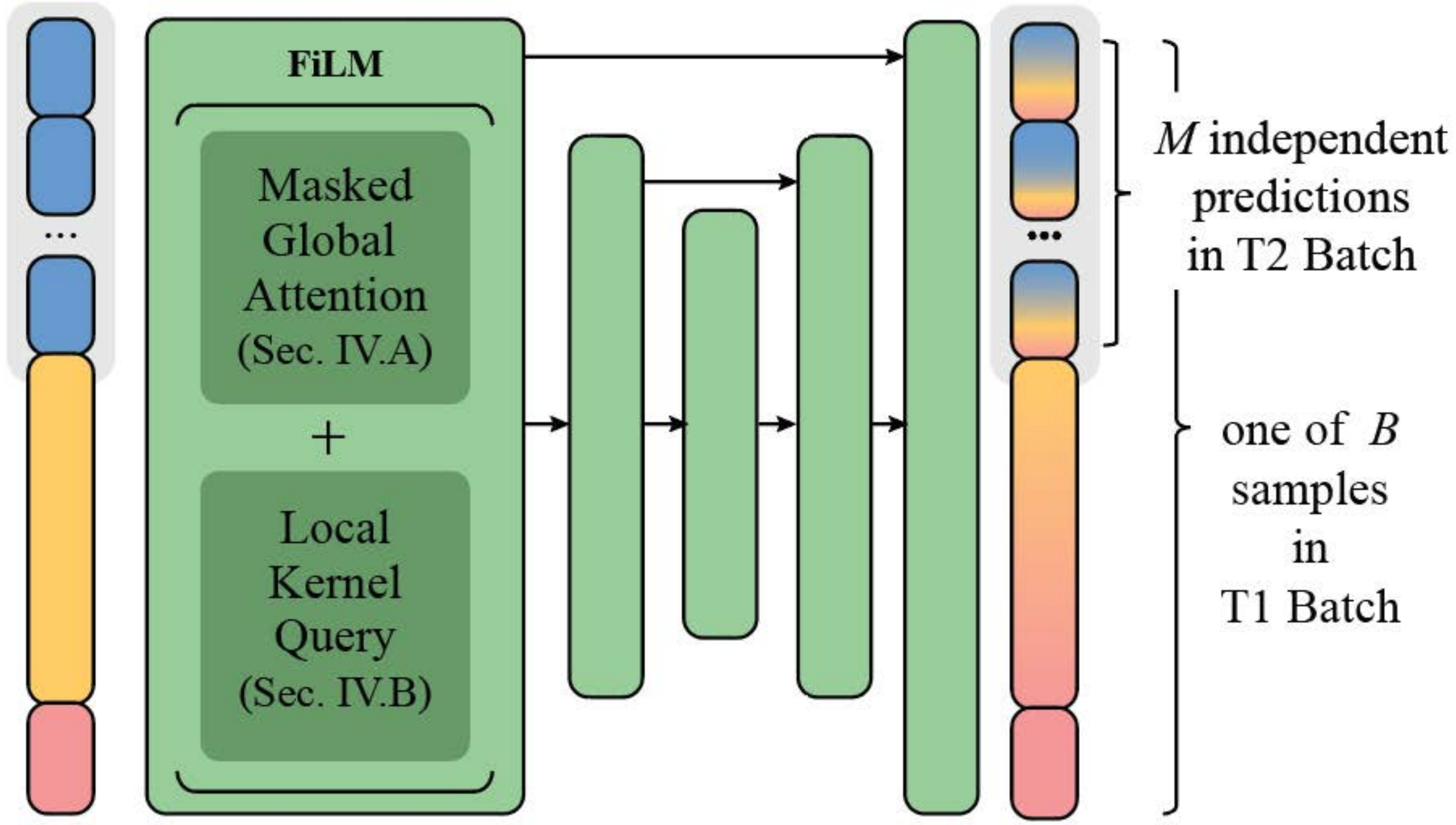


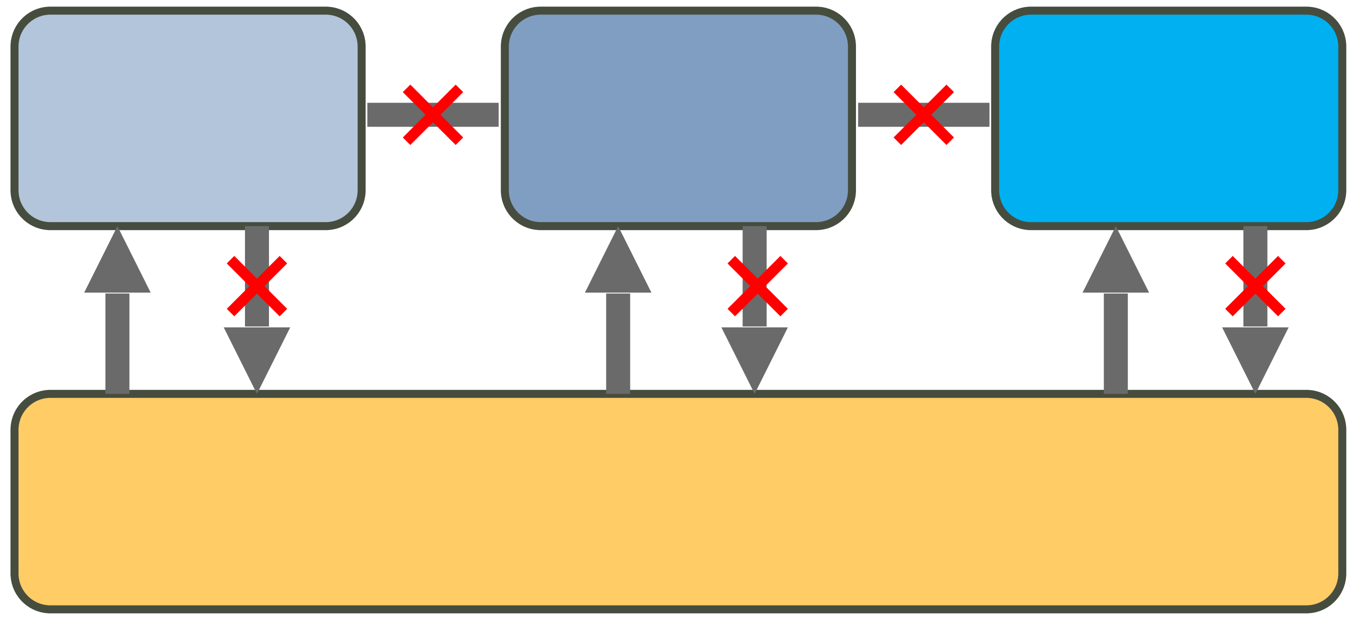
Cross attention



Kernel query

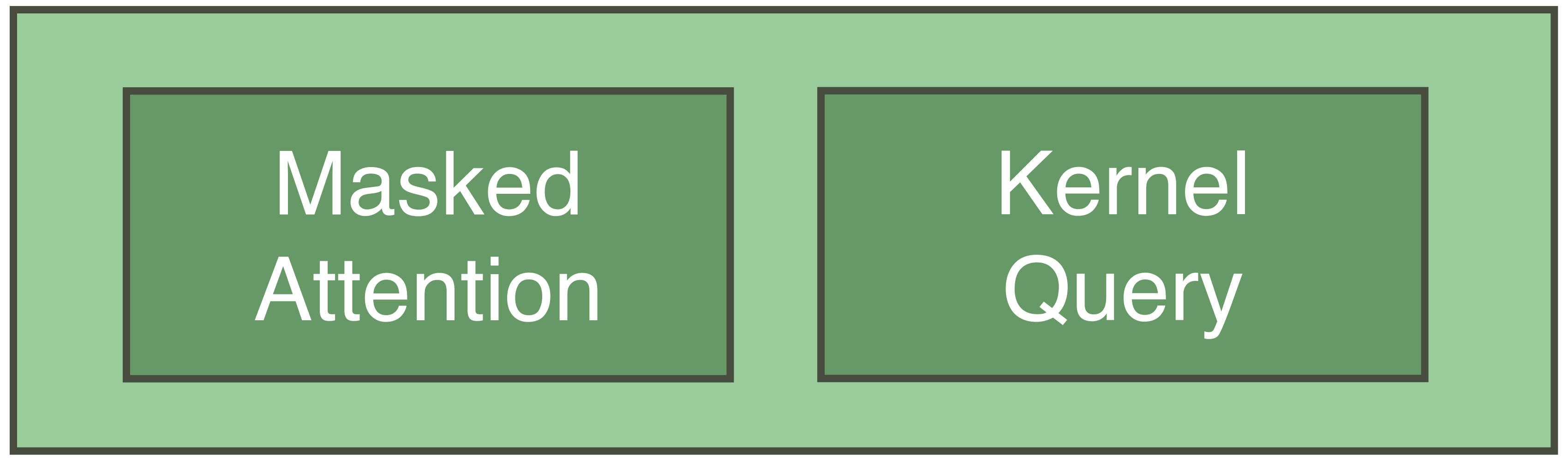
# Masked Attention Protects Action Sample Independence

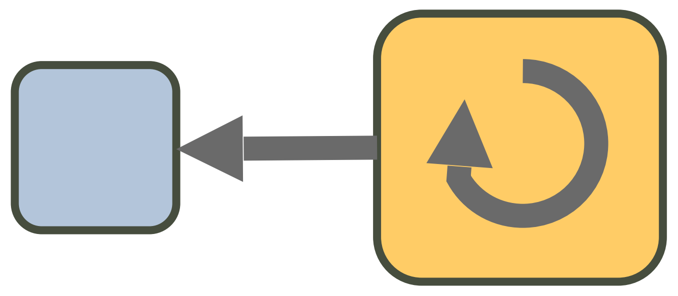

FiLM
Masked Global Attention (Sec. IV.A)
+
Local Kernel Query (Sec. IV.B)
M independent predictions in T2 Batch
one of B samples in T1 Batch

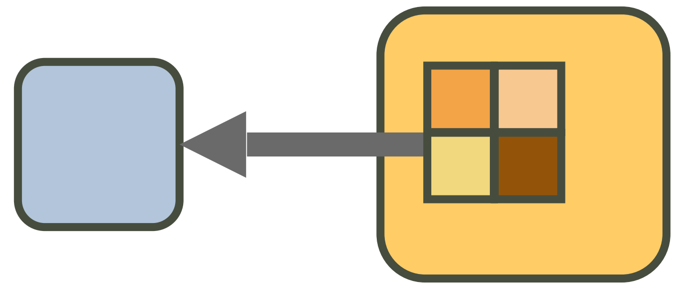


(i) An action sample attends to itself and shared conditions, but not to other action samples
(ii) shared conditions do not attend back to action samples.

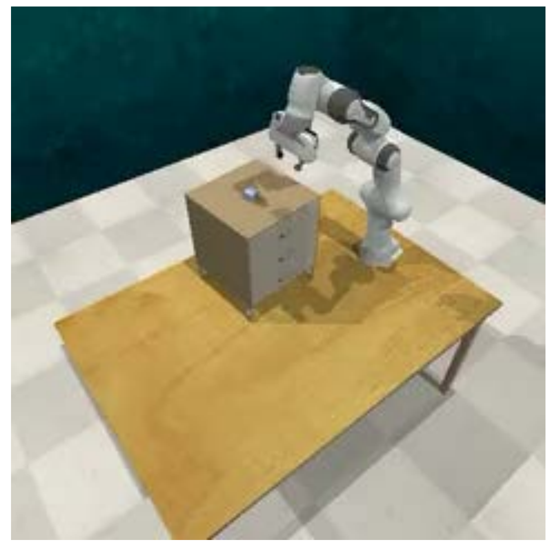
Two modules can perform such a "Non-invasive extraction"

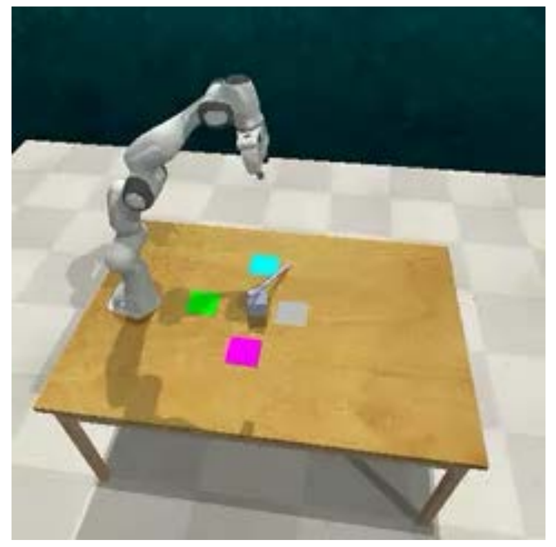






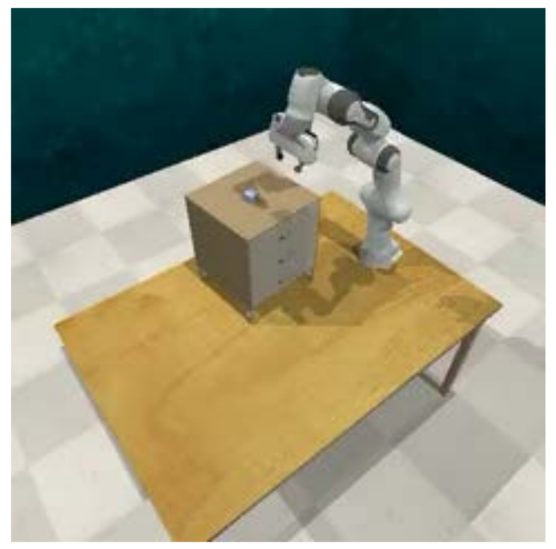
## Training an 18-in-1 multi-task model for RL-Bench

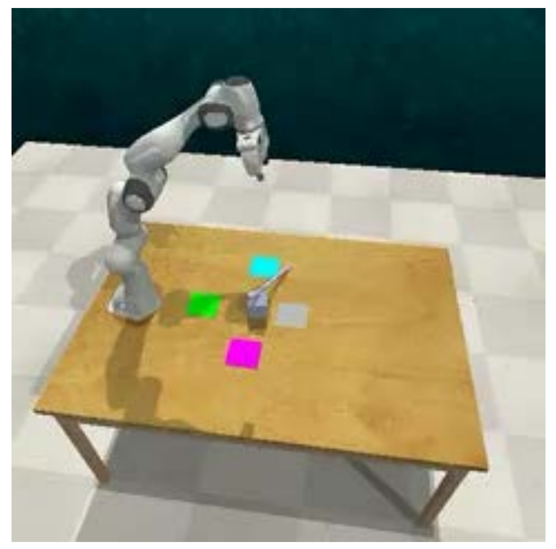




| Method | Avg. Suc. (%) | Norm. Time | Memory (GB) | Reported Hardware |
|---|---|---|---|---|
| PerAct | 49.4 | 128 | 128 | V100×8×16 days |
| RVT | 62.9 | 8 | 128 | V100×8×1 day |
| Act3D | 63.2 | 40 | 128 | V100×8×5 days |
| RVT-2 | 81.4 | 6.6 | 128 | V100×8×20 hours |
| 3D-Dif-Actor | 81.3 (100%) | 39 (100%) | 240 (100%) | A100×6×6 days |
| SAM2Act | 86.8 | 8.3 | 160 | H100×8×12 hours |
| Mini-diffuser | 77.6 (**95.4%**) | **1.9 (4.8%)** | **16 (6.6%)** | 4090×13 hours or A100×1 day |

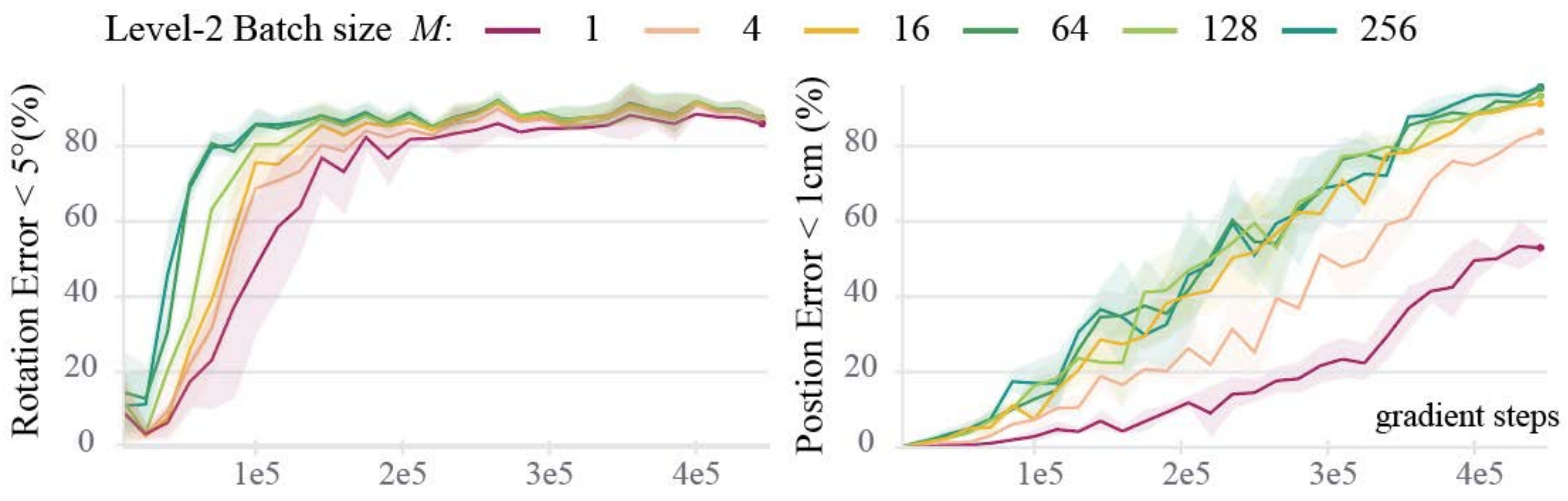
# Training an 18-in-1 multi-task model for RL-Bench





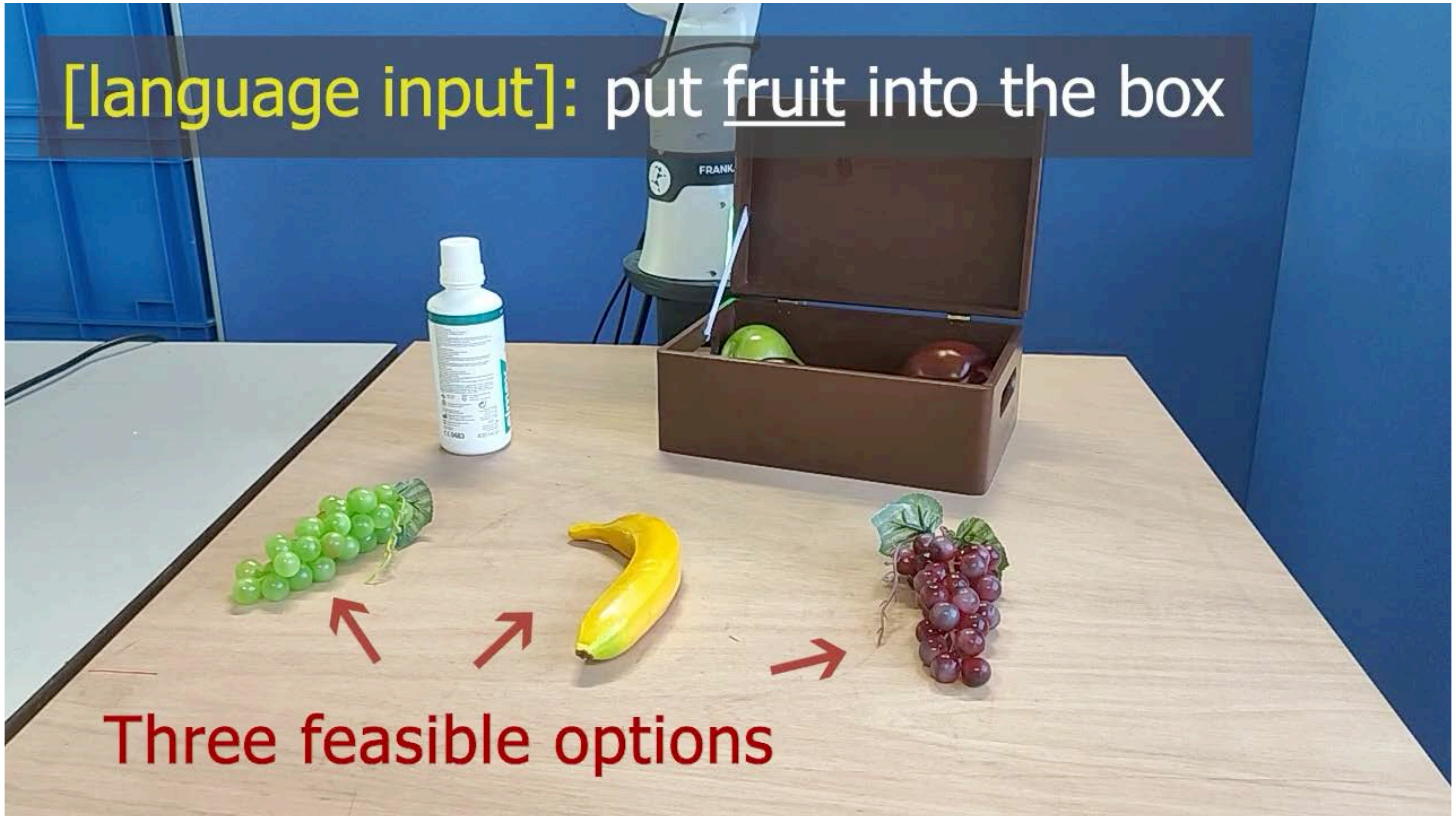
| Method | Avg. Suc. (%) | Norm. Time | Memory (GB) | Reported Hardware |
|---|---|---|---|---|
| PerAct | 49.4 | 128 | 128 | V100×8×16 days |
| RVT | 62.9 | 8 | 128 | V100×8×1 day |
| Act3D | 63.2 | 40 | 128 | V100×8×5 days |
| RVT-2 | 81.4 | 6.6 | 128 | V100×8×20 hours |
| 3D-Dif-Actor | 81.3 (100%) | 39 (100%) | 240 (100%) | A100×6×6 days |
| SAM2Act | 86.8 | 8.3 | 160 | H100×8×12 hours |
| Mini-diffuser | 77.6 (**95.4%**) | **1.9 (4.8%)** | **16 (6.6%)** | 4090×13 hours or A100×1 day |

# Efficiency of Level-2 Batch

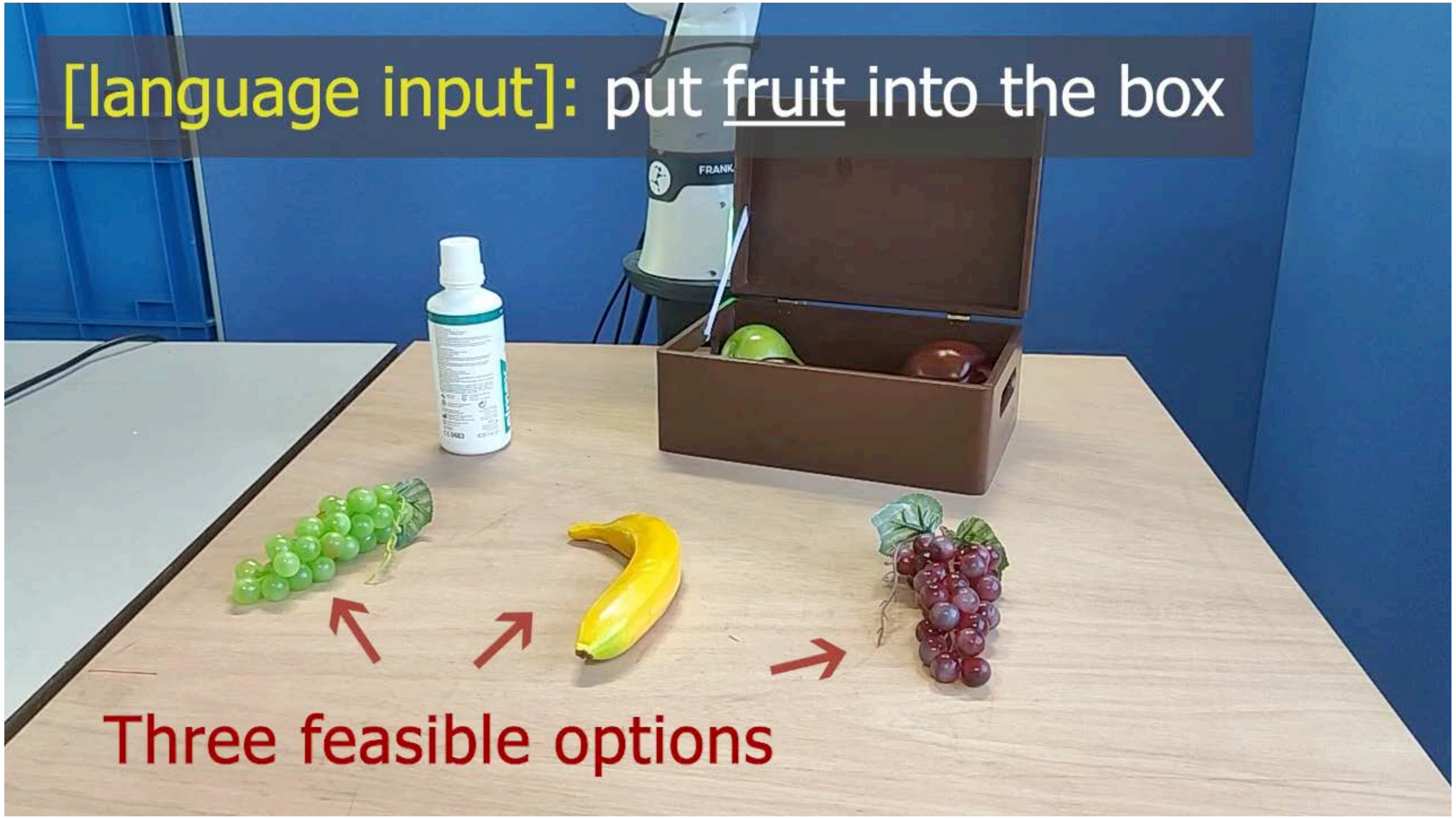


| Level-1 batches: B<br>Level-2 batches: M | Memory Cost | Time per Gradient Step | Avg. Succ. after 1e5 Steps |
|---|---|---|---|
| B=100 M=64 | 102.2% | 106.3% | 78.3 |
| B=100 M=1 | 100% | 100% | 44.1 |
| B=200 M=1 | 188.8% | 176.6% | 50.8 |

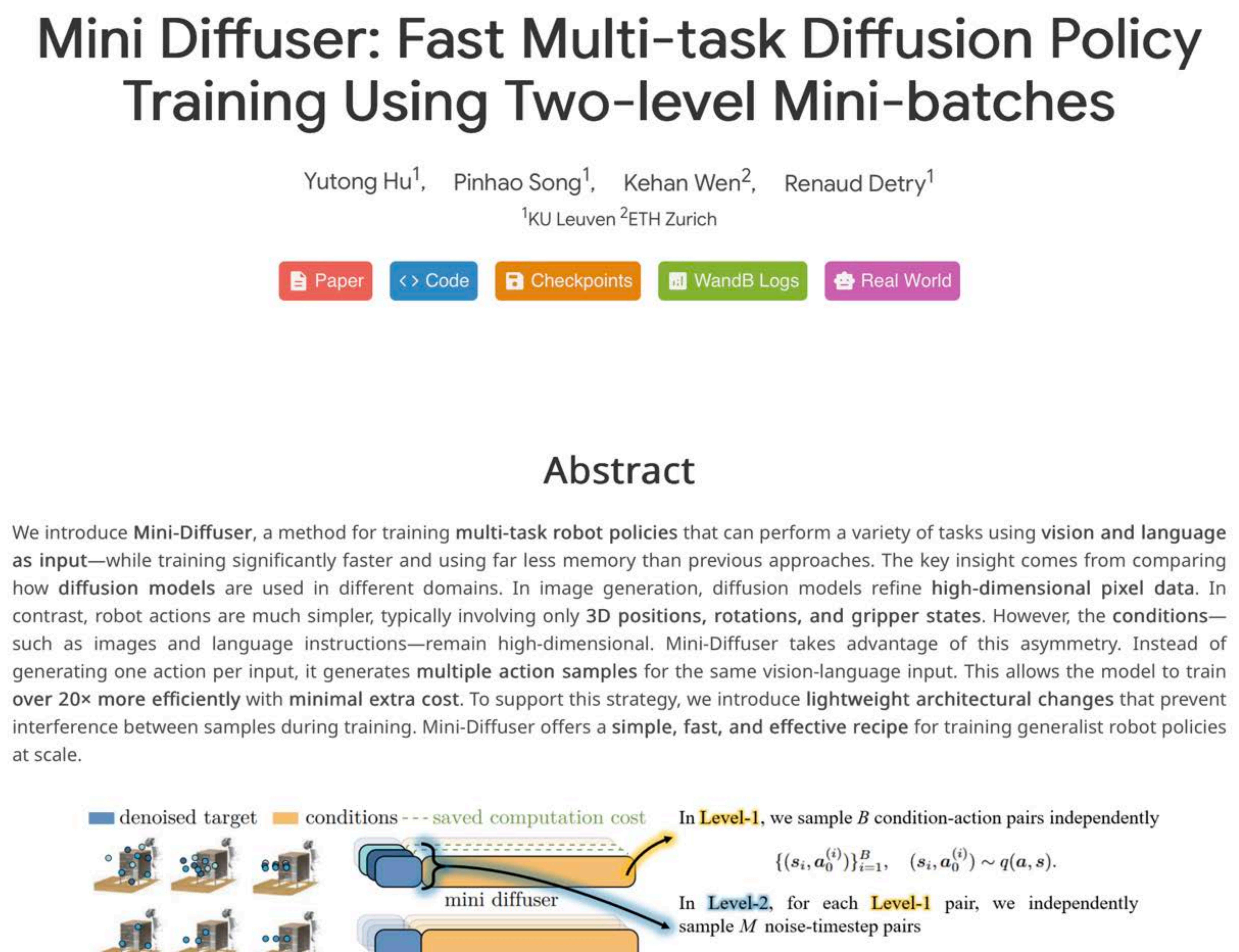
## Train a multi-task Diffuser Actor in the Realworld



## Train a multi-task Diffuser Actor in the Realworld



https://mini-diffuse-actor.github.io

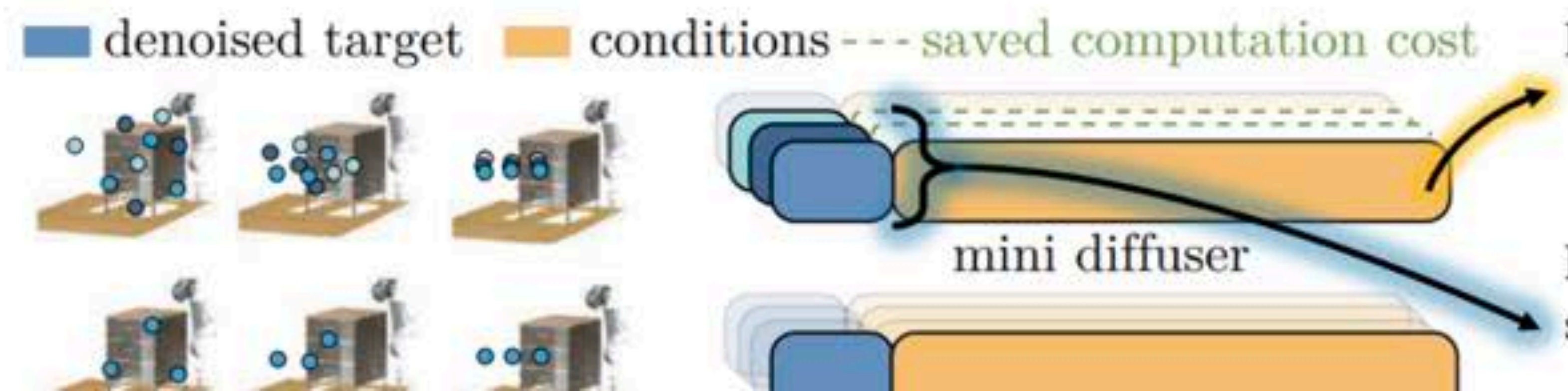


# Mini Diffuser: Fast Multi-task Diffusion Policy Training Using Two-level Mini-batches

Yutong Hu[1], Pinhao Song[1], Kehan Wen[2], Renaud Detry[1]
[1]KU Leuven [2]ETH Zurich

## Abstract

We introduce **Mini-Diffuser**, a method for training **multi-task robot policies** that can perform a variety of tasks using **vision and language as input**—while training significantly faster and using far less memory than previous approaches. The key insight comes from comparing how **diffusion models** are used in different domains. In image generation, diffusion models refine **high-dimensional pixel data**. In contrast, robot actions are much simpler, typically involving only **3D positions, rotations, and gripper states**. However, the **conditions**—such as images and language instructions—remain high-dimensional. Mini-Diffuser takes advantage of this asymmetry. Instead of generating one action per input, it generates **multiple action samples** for the same vision-language input. This allows the model to train **over 20× more efficiently** with **minimal extra cost**. To support this strategy, we introduce **lightweight architectural changes** that prevent interference between samples during training. Mini-Diffuser offers a **simple, fast, and effective recipe** for training generalist robot policies at scale.

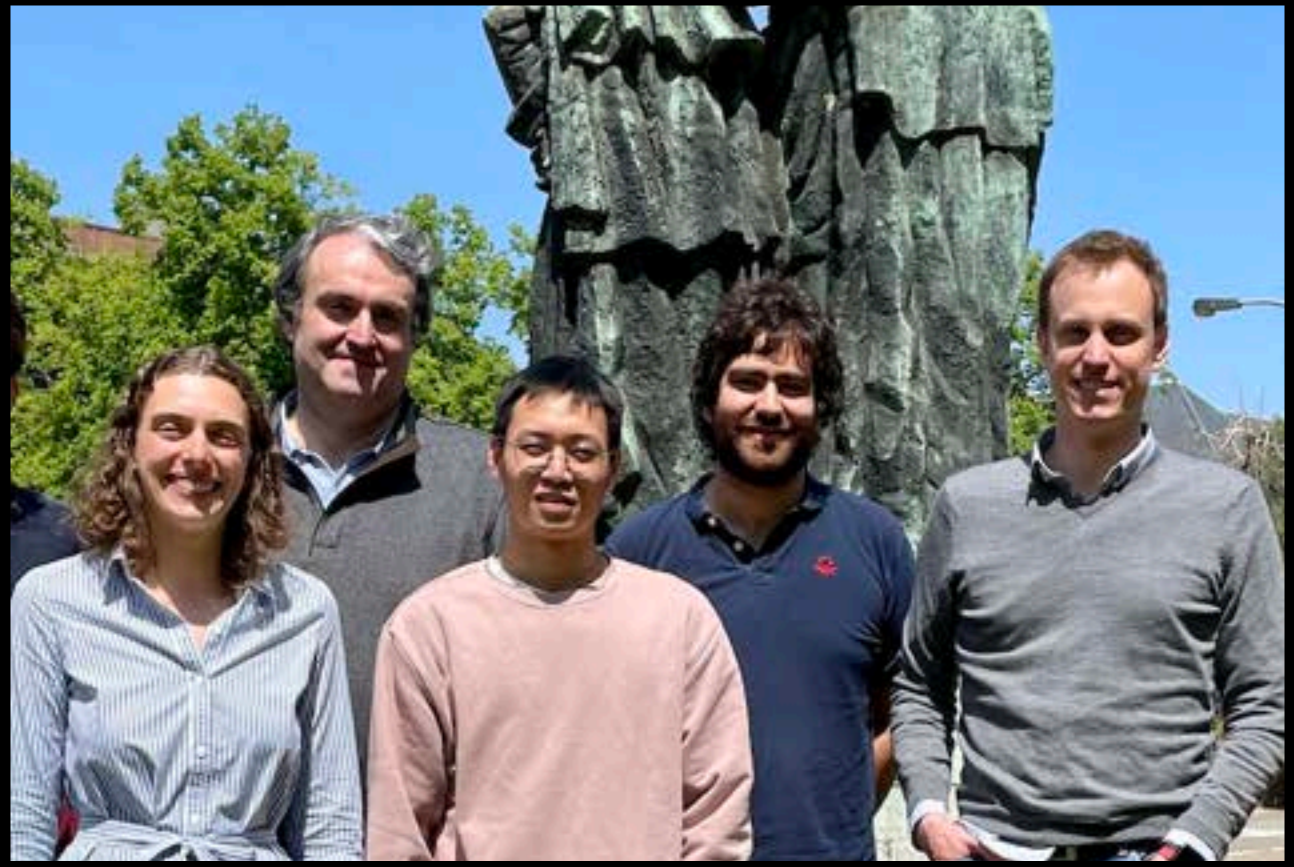


In Level-1, we sample $B$ condition-action pairs independently

$$\{(s_i, a_0^{(i)})\}_{i=1}^{B}, \quad (s_i, a_0^{(i)}) \sim q(a, s).$$

In Level-2, for each Level-1 pair, we independently sample $M$ noise-timestep pairs

# Equivariant Volumetric Grasping



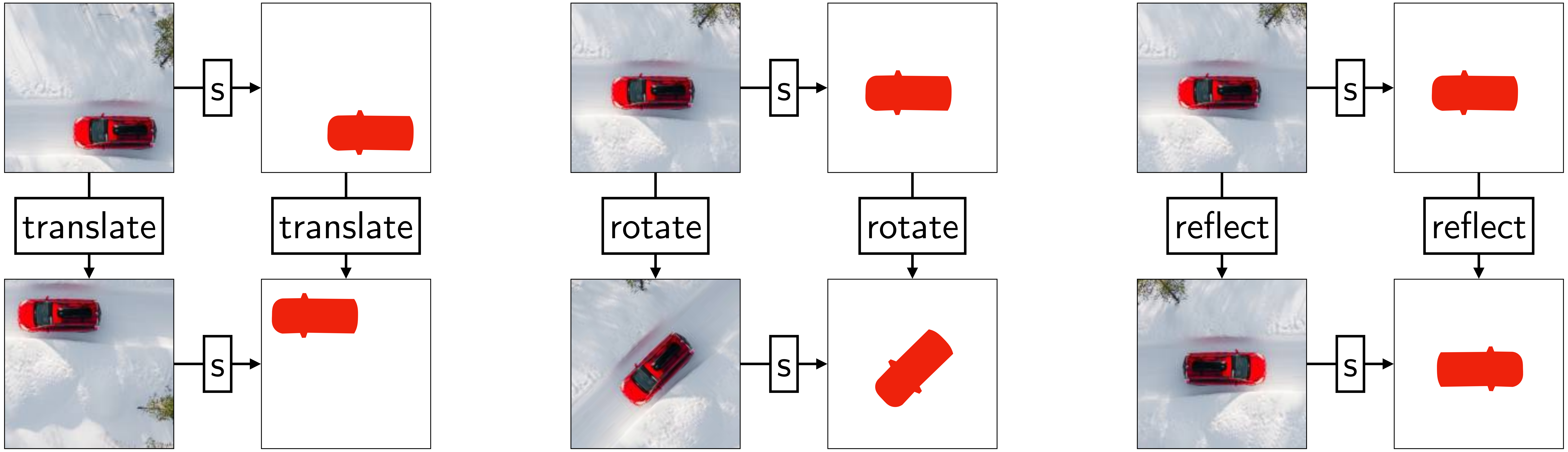
P. Song, Y. Hu, P. Li, and R. Detry

KU LEUVEN

At UCLA on 2025-08-27

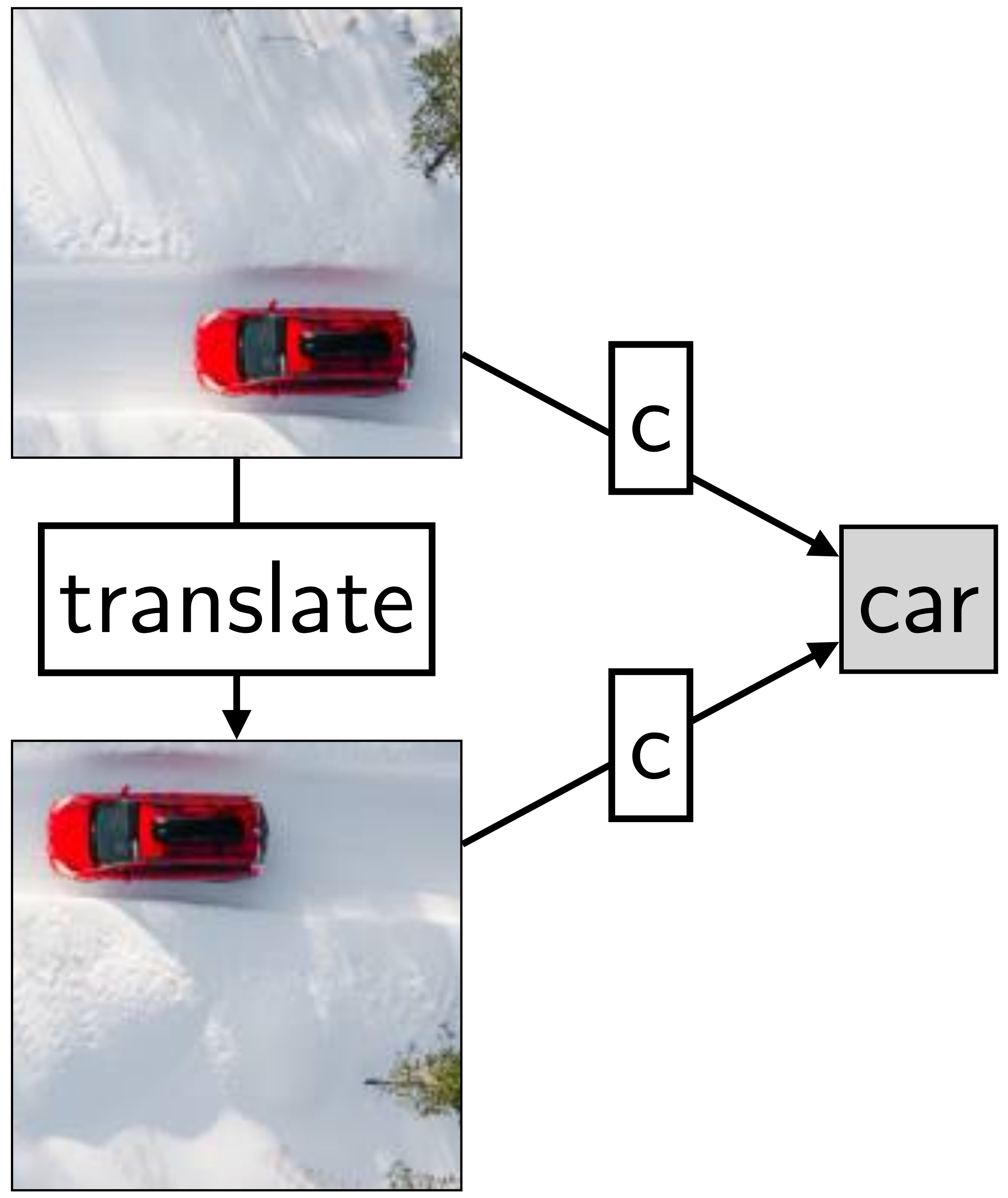
## A model is equivariant if its output responds predictably to transformations of its input

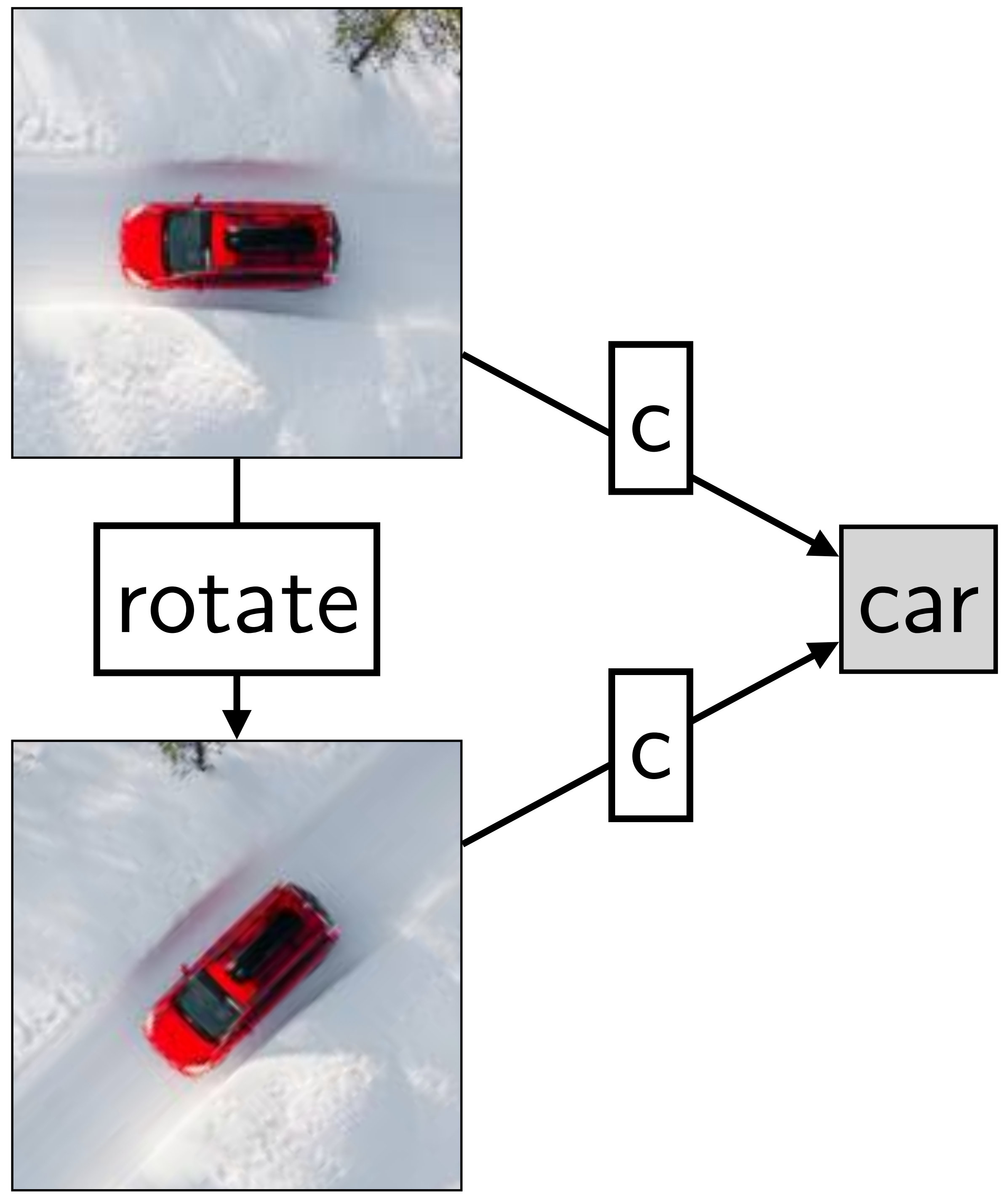
A trivial way of responding predictably: responding *identically*

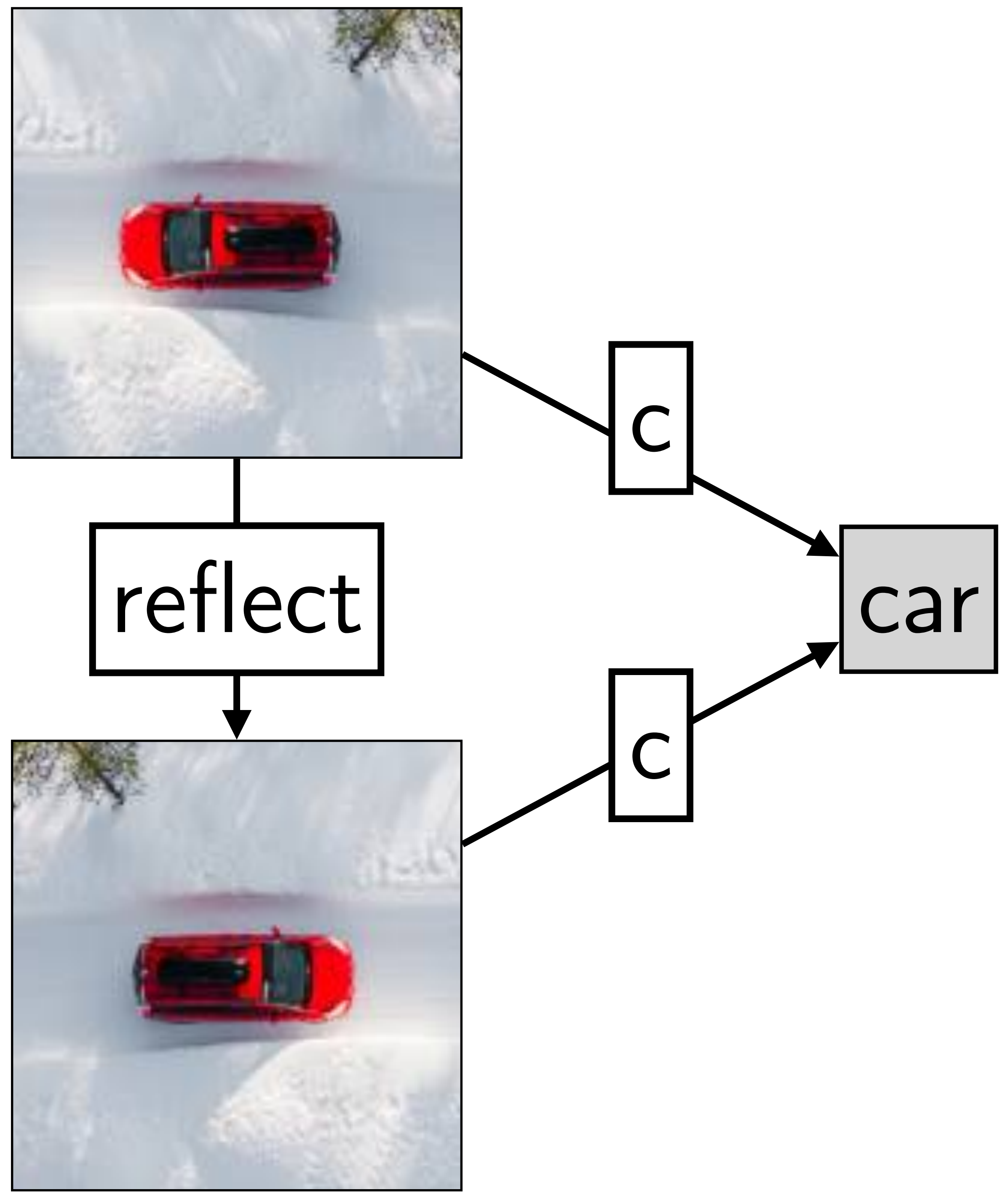


$\boxed{s}$ = image segmentation model

# A model is invariant if its output is immutable to transformations of its input

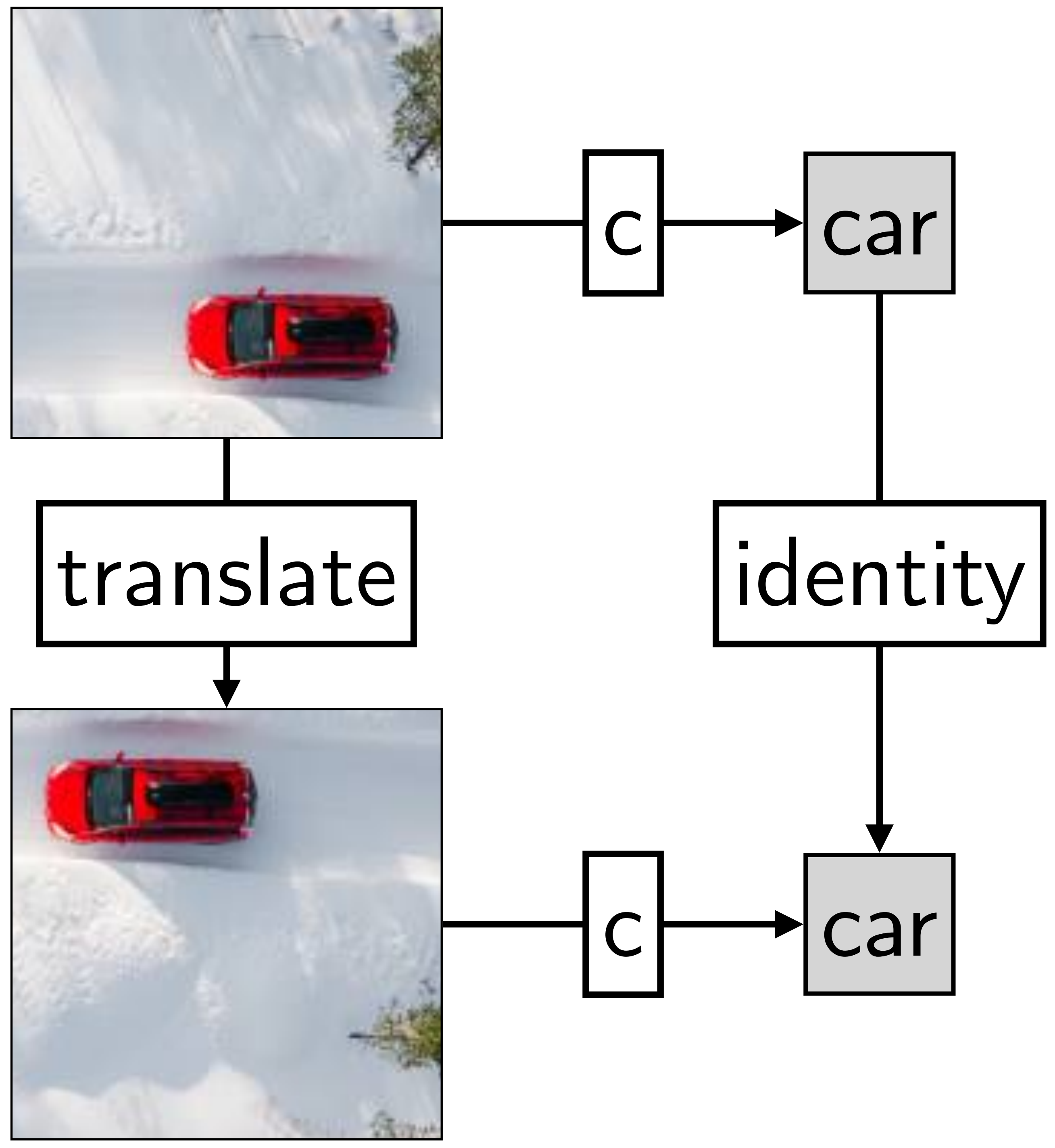


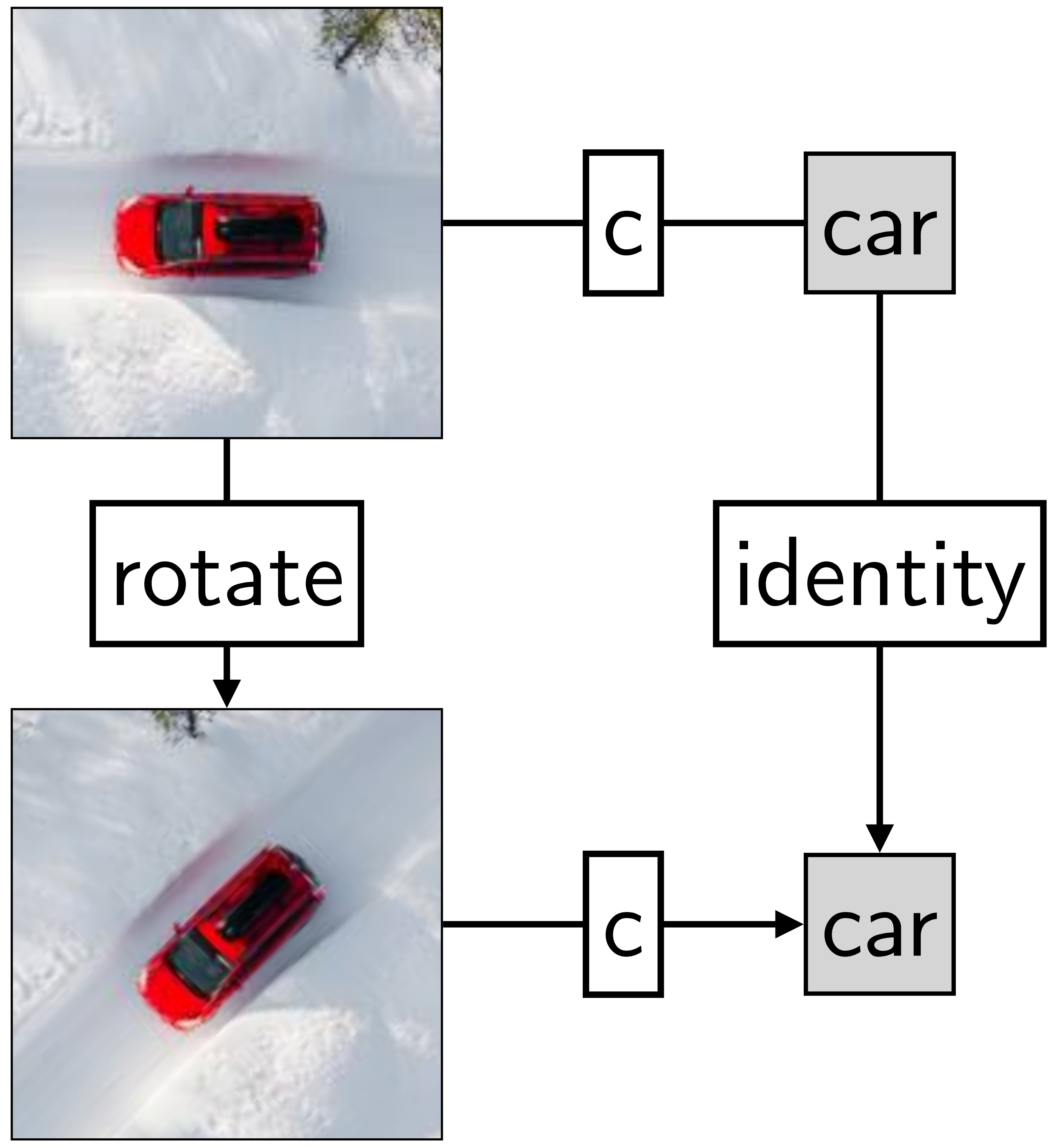


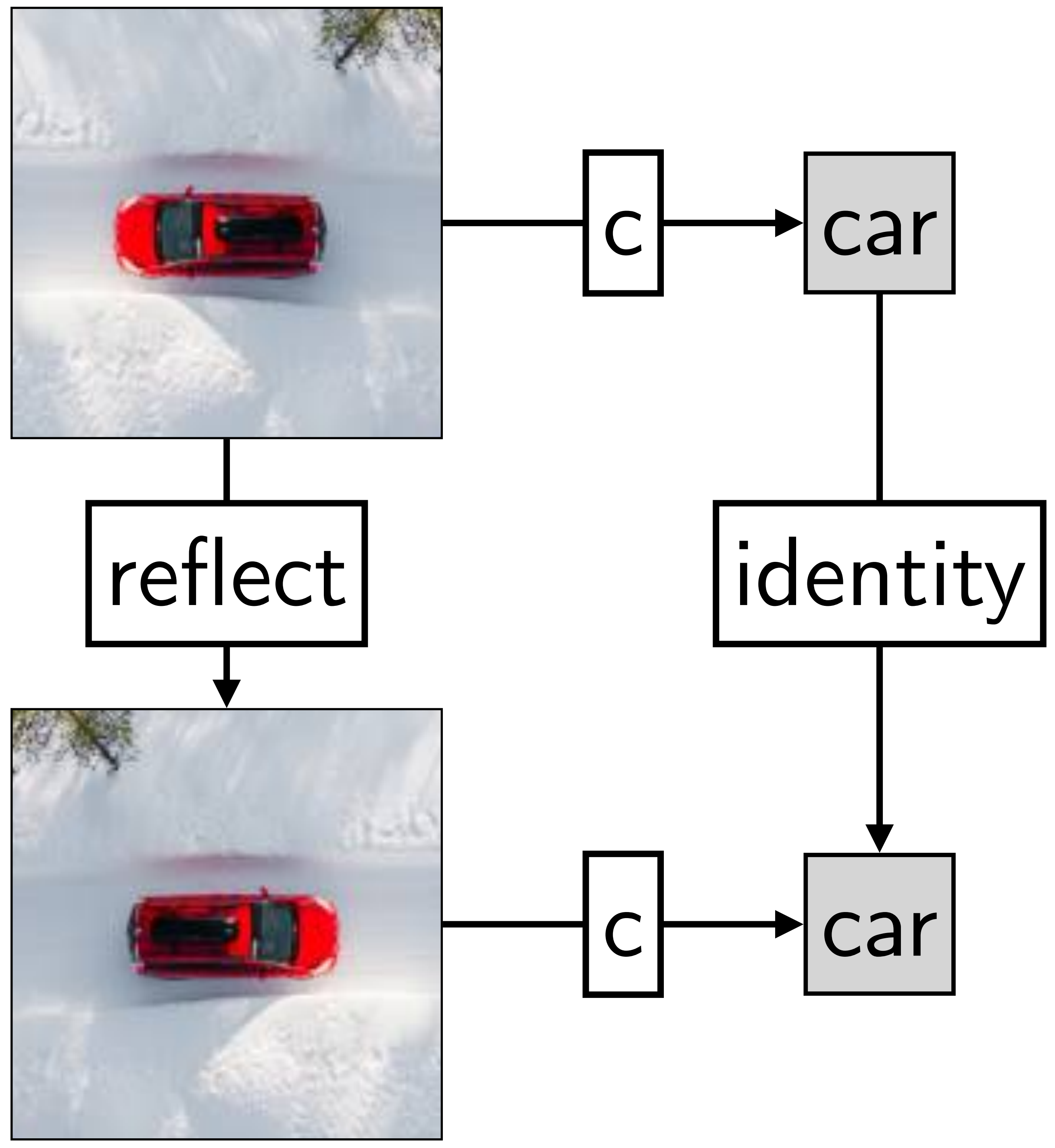


$\boxed{c}$ = image classification model

## Invariance is a special case of equivariance, where the output transformation is the identity transformation



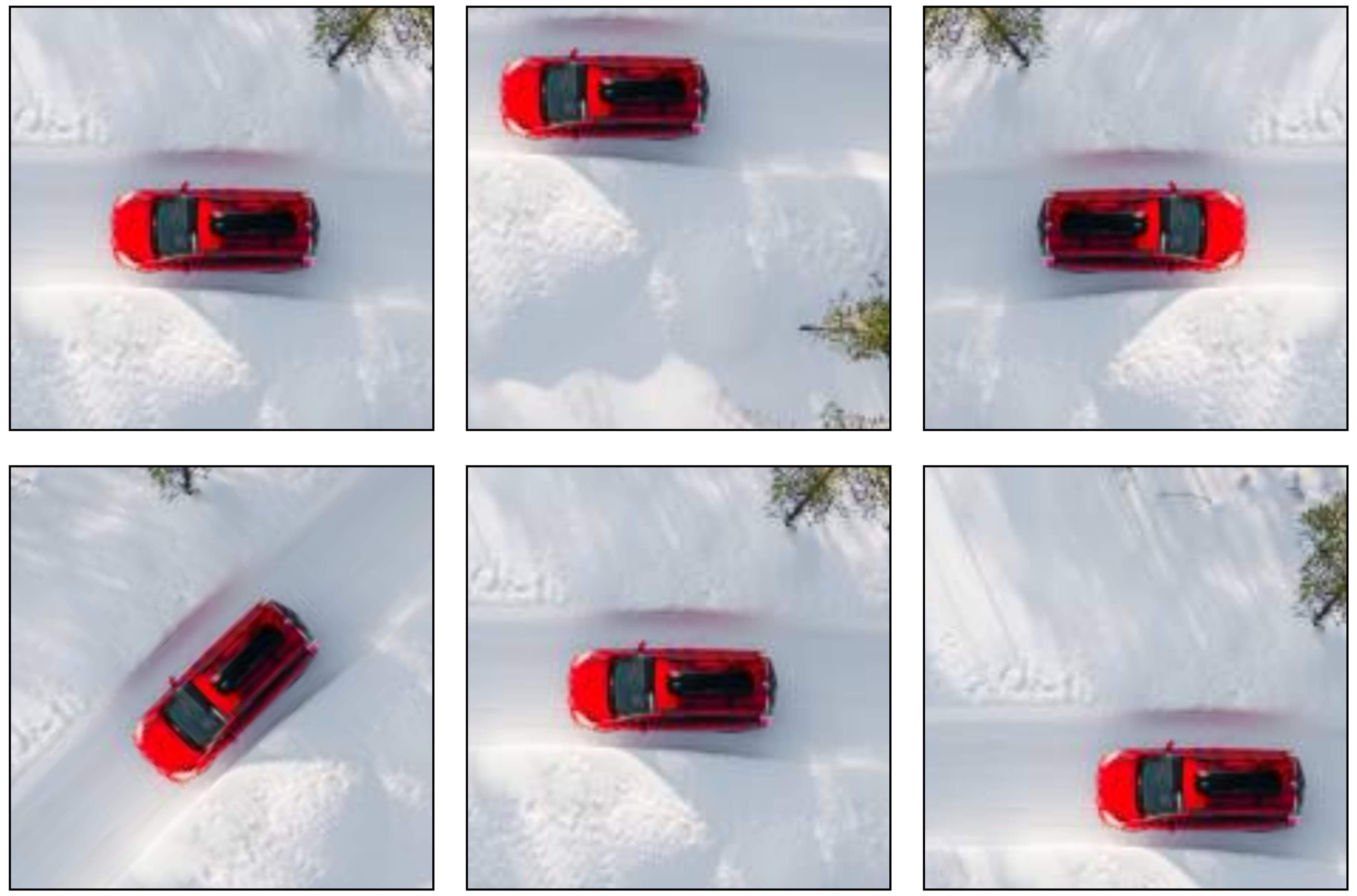




$\boxed{c}$ = image classification model

## Data-driven models can achieve equivariance (a) through exposure to tons of data, or (b) through architectural design

(a)
Vanilla model (e.g., MLP)
trained on:

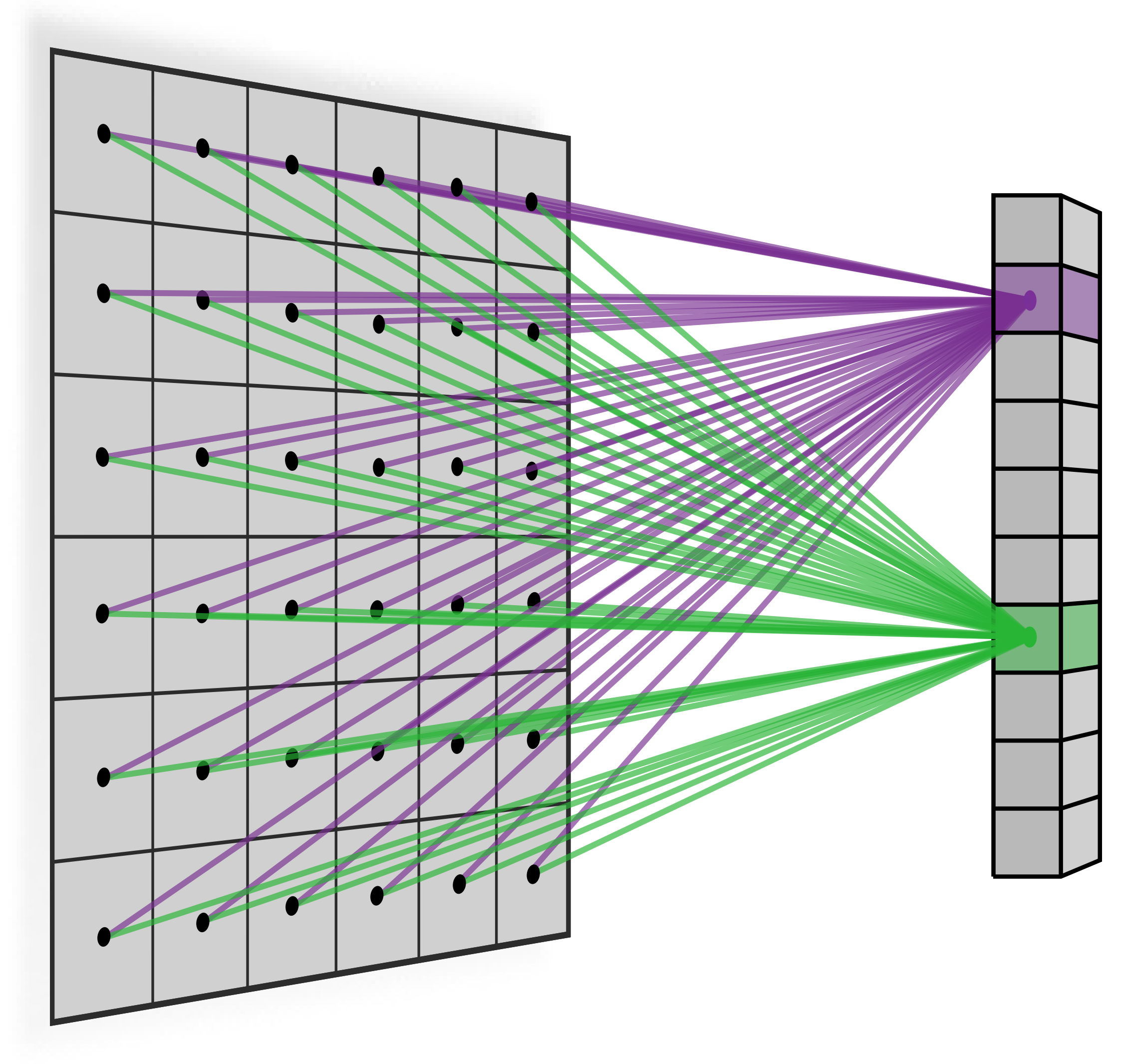


(b)
Architecturally-equivariant model
trained on:

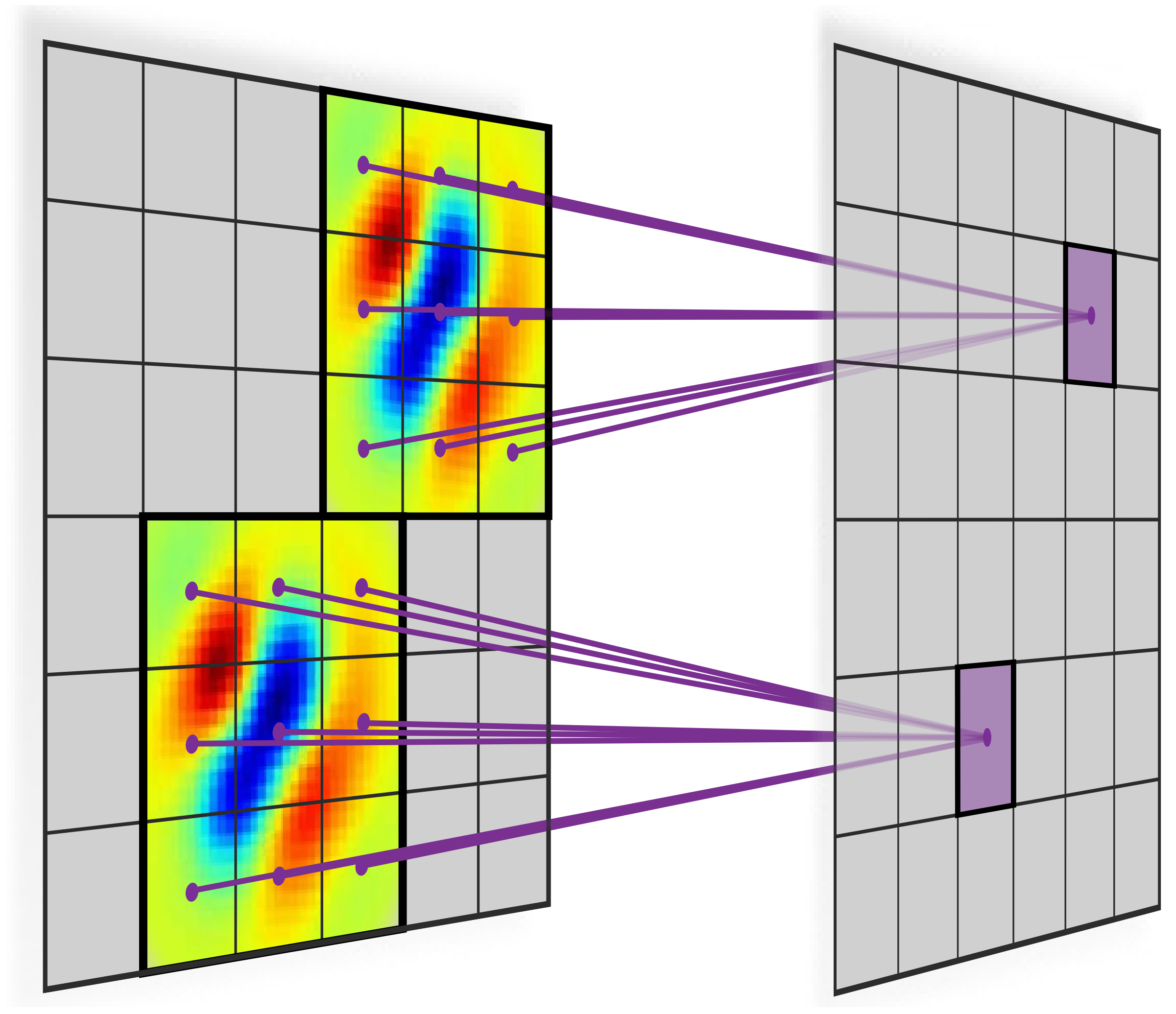
Better *sample efficiency*

## A canonical example of a model designed for translation equivariance: the convolutional neural network

(a)
MLP (no architectural equivariance)

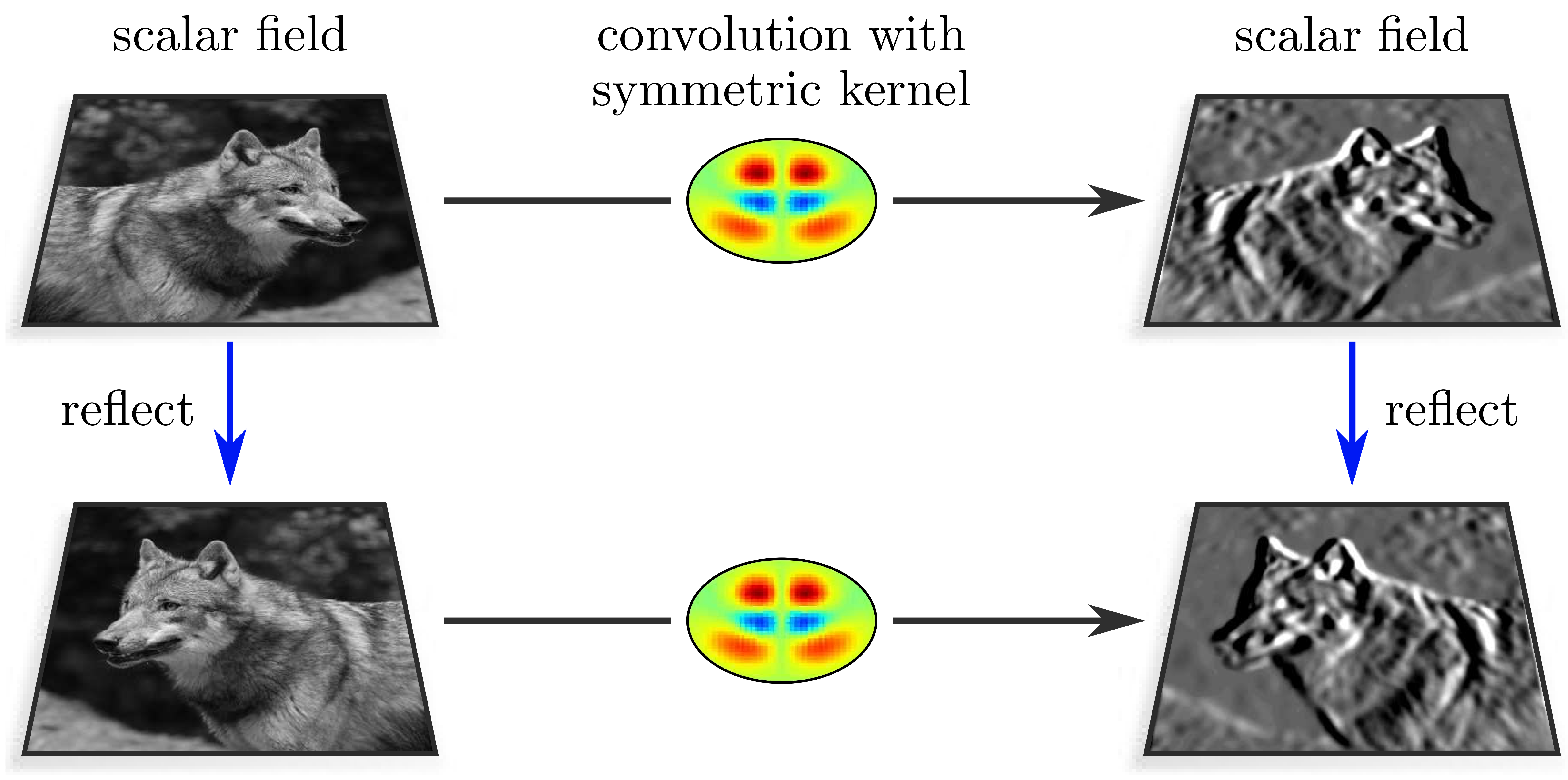


(b)
CNN (architectural equivariance to translations)



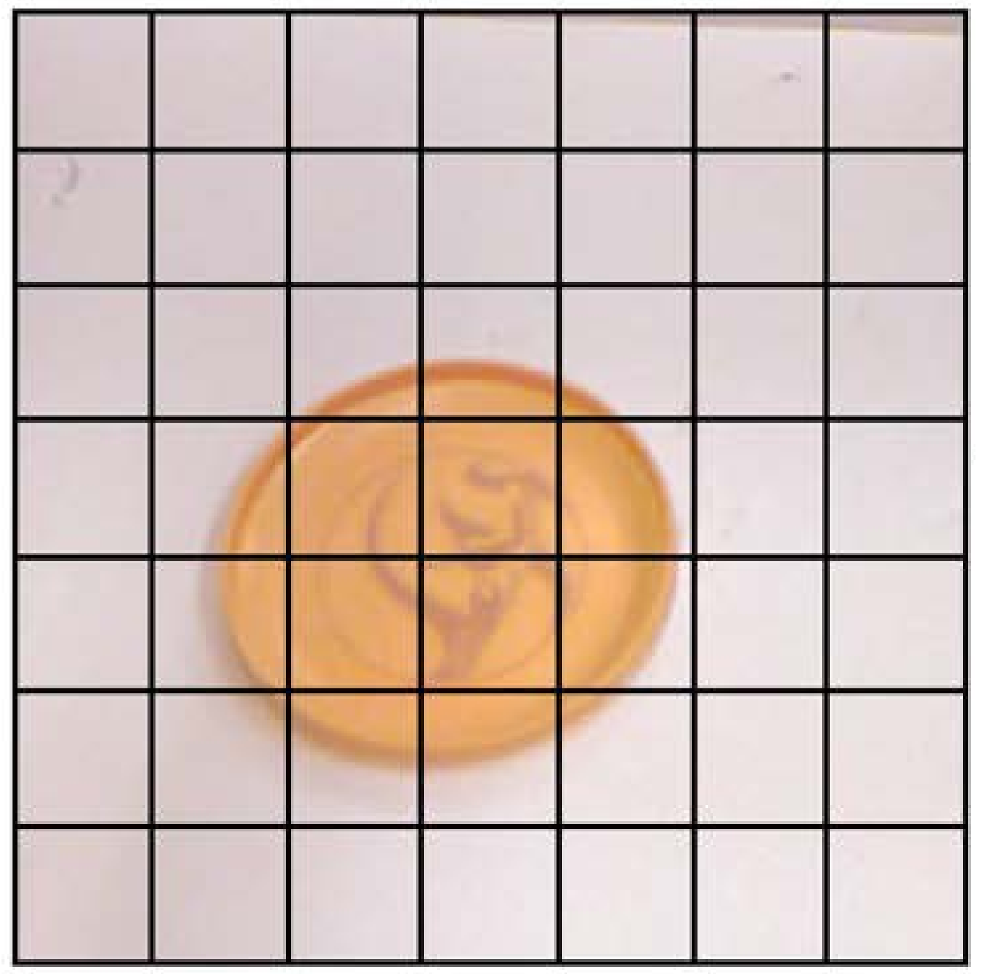
[1] M. Weiler, P. Forre, E. Verlinde, and M. Welling. Equivariant and Coordinate Independent Convolutional Networks. 2023.

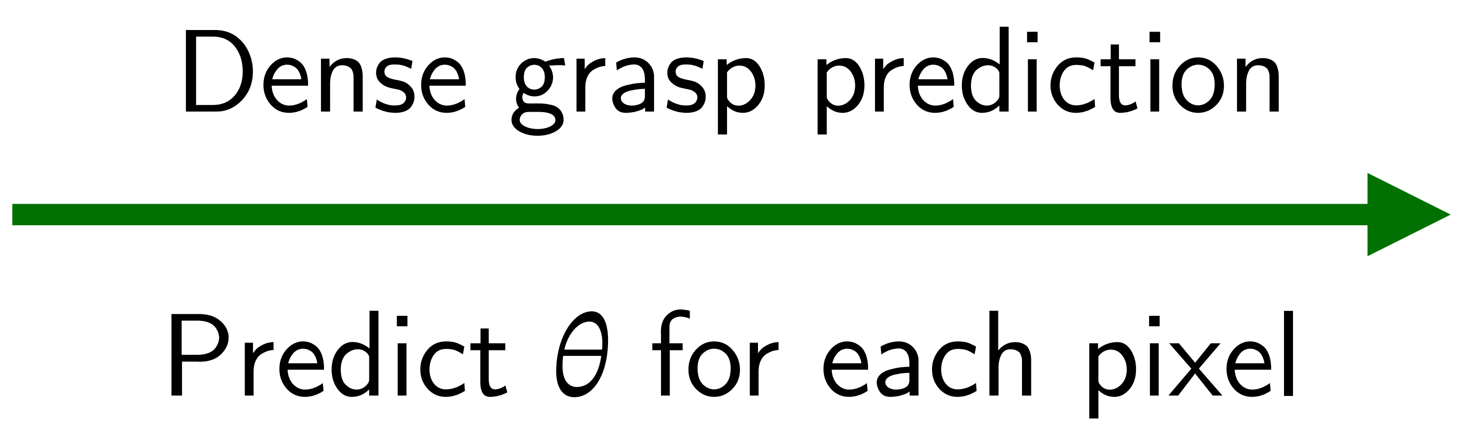
## DNN building blocks that provide equivariance to rotations or reflections readily exist in software libraries. They are generally referred to as steerable kernels.

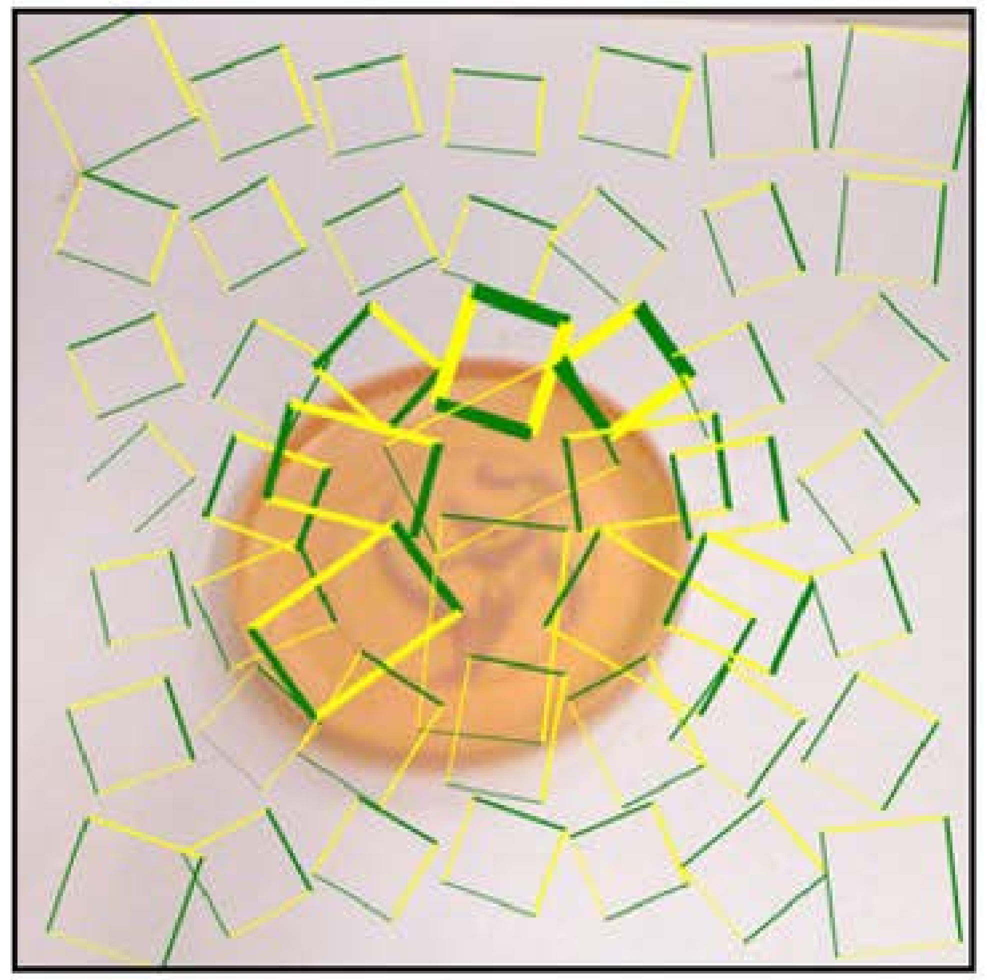


[1] M. Weiler, P. Forre, E. Verlinde, and M. Welling. Equivariant and Coordinate Independent Convolutional Networks. 2023.

## *Dense grasp prediction* is similar in spirit to dense image processing: it predicts parameters for each pixel of an input image

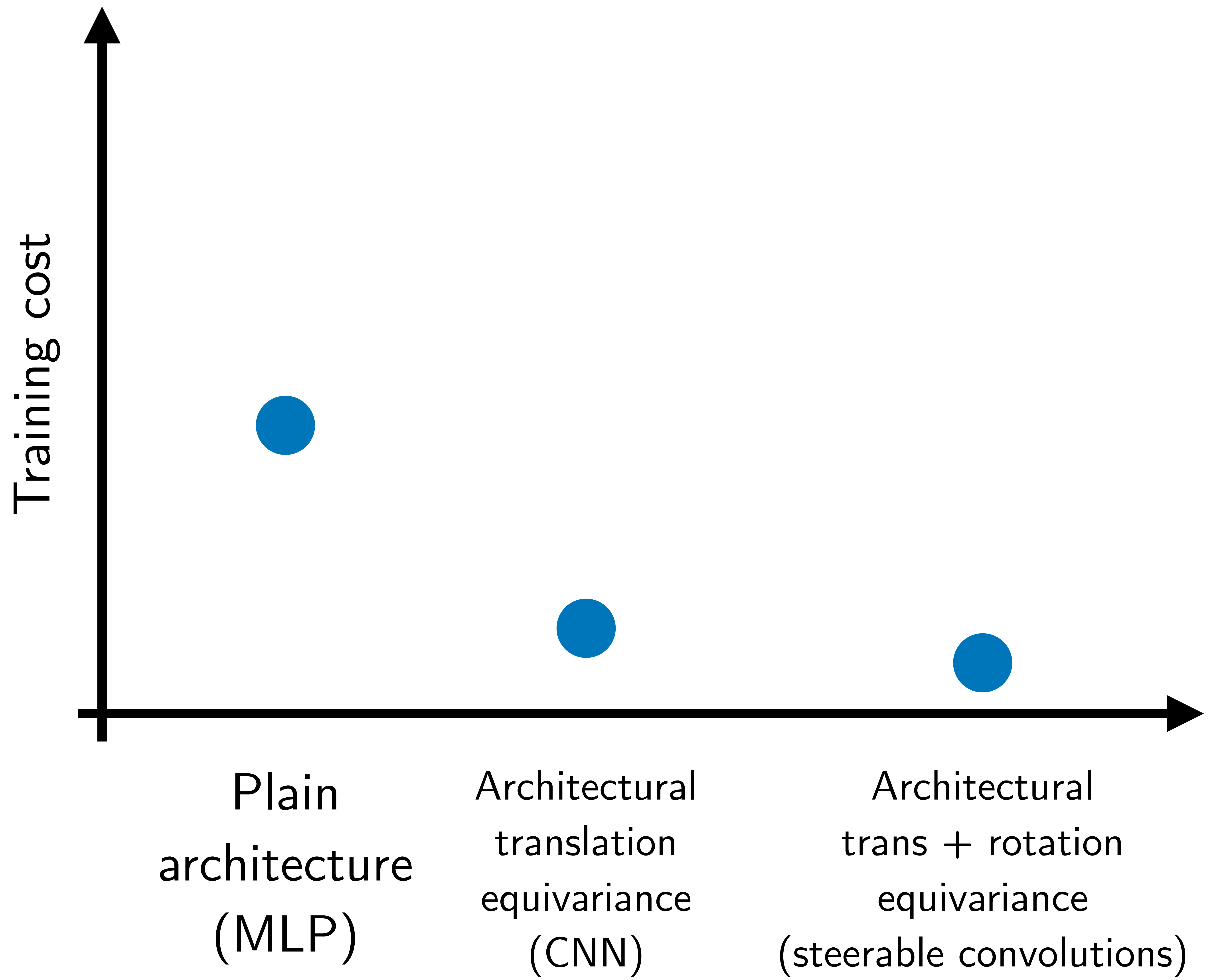


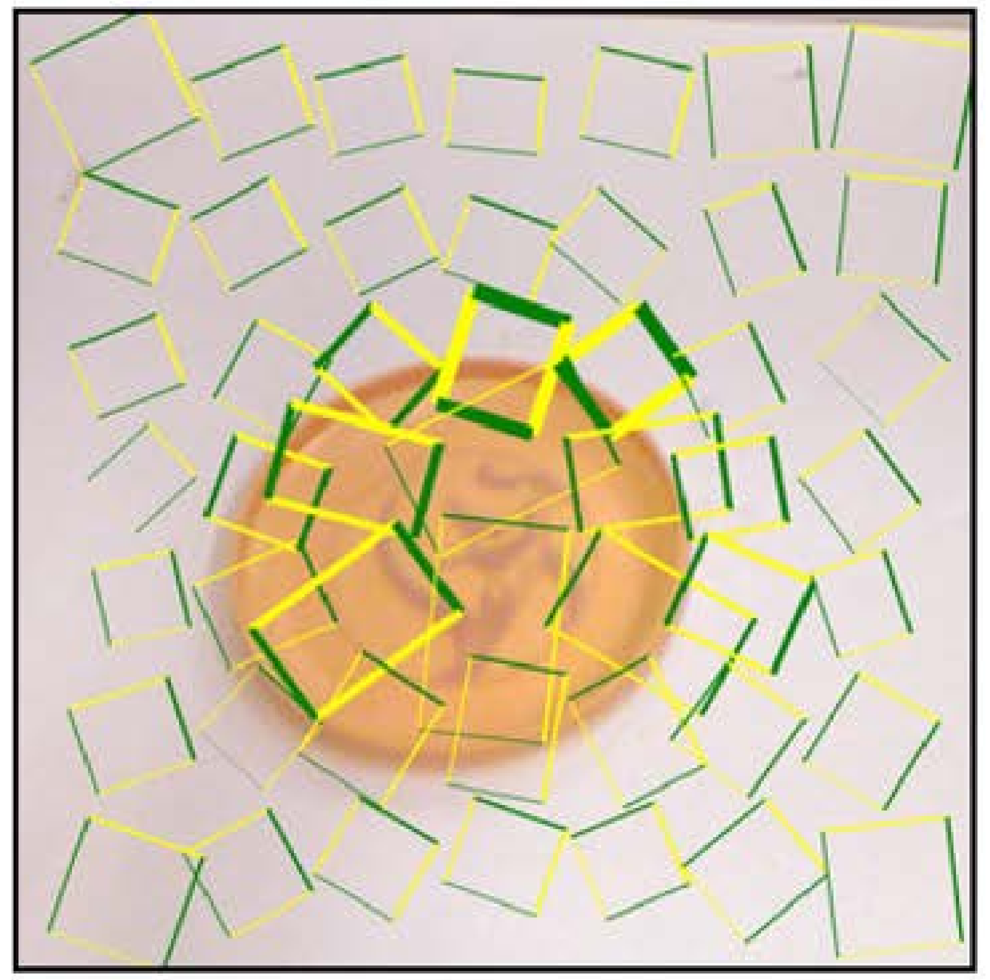




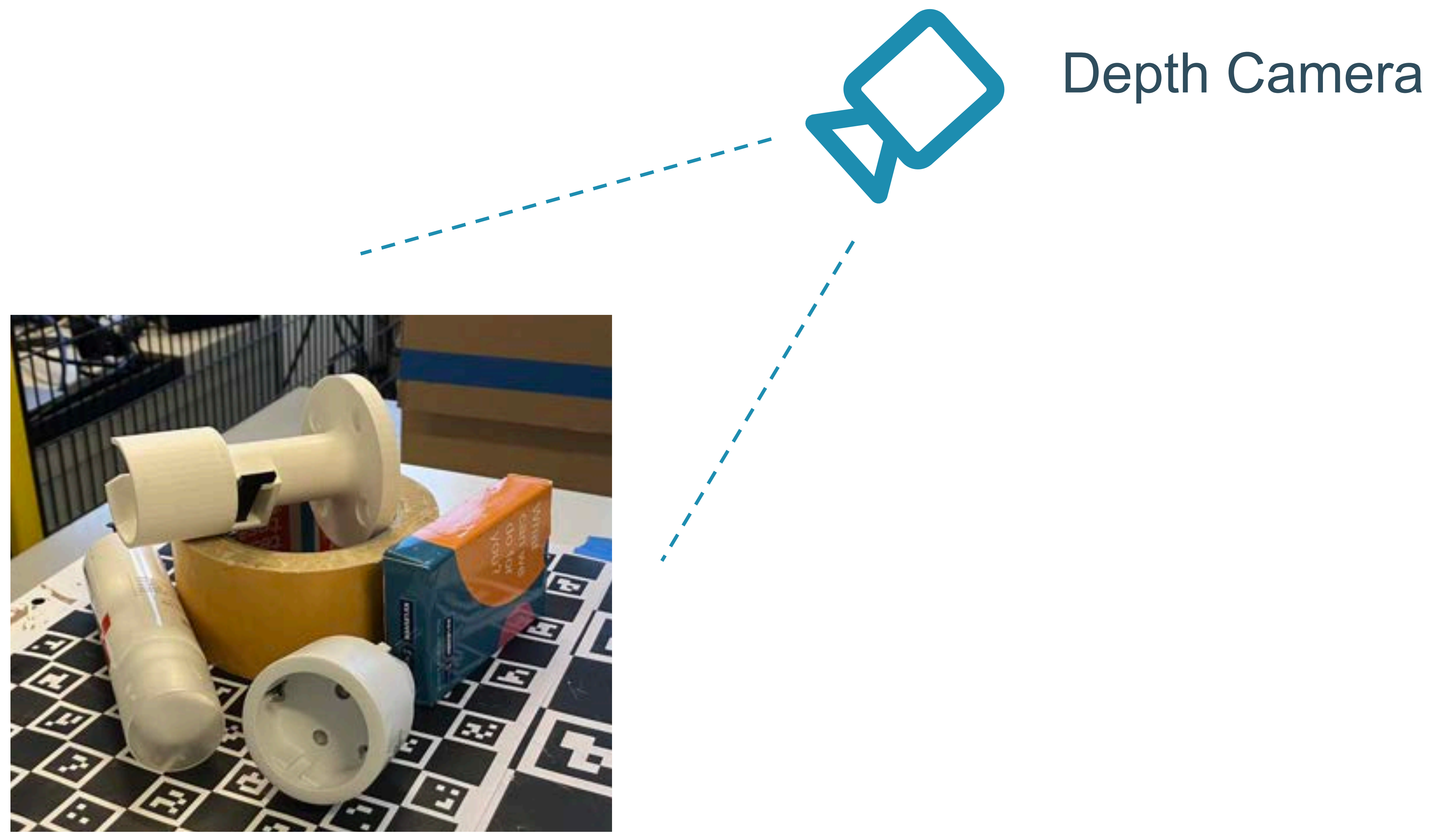
Redmon, J., & Angelova, A. Real-time grasp detection using convolutional neural networks. ICRA 2015.

## Architecturally-equivariant models improve sample efficiency and lower training costs

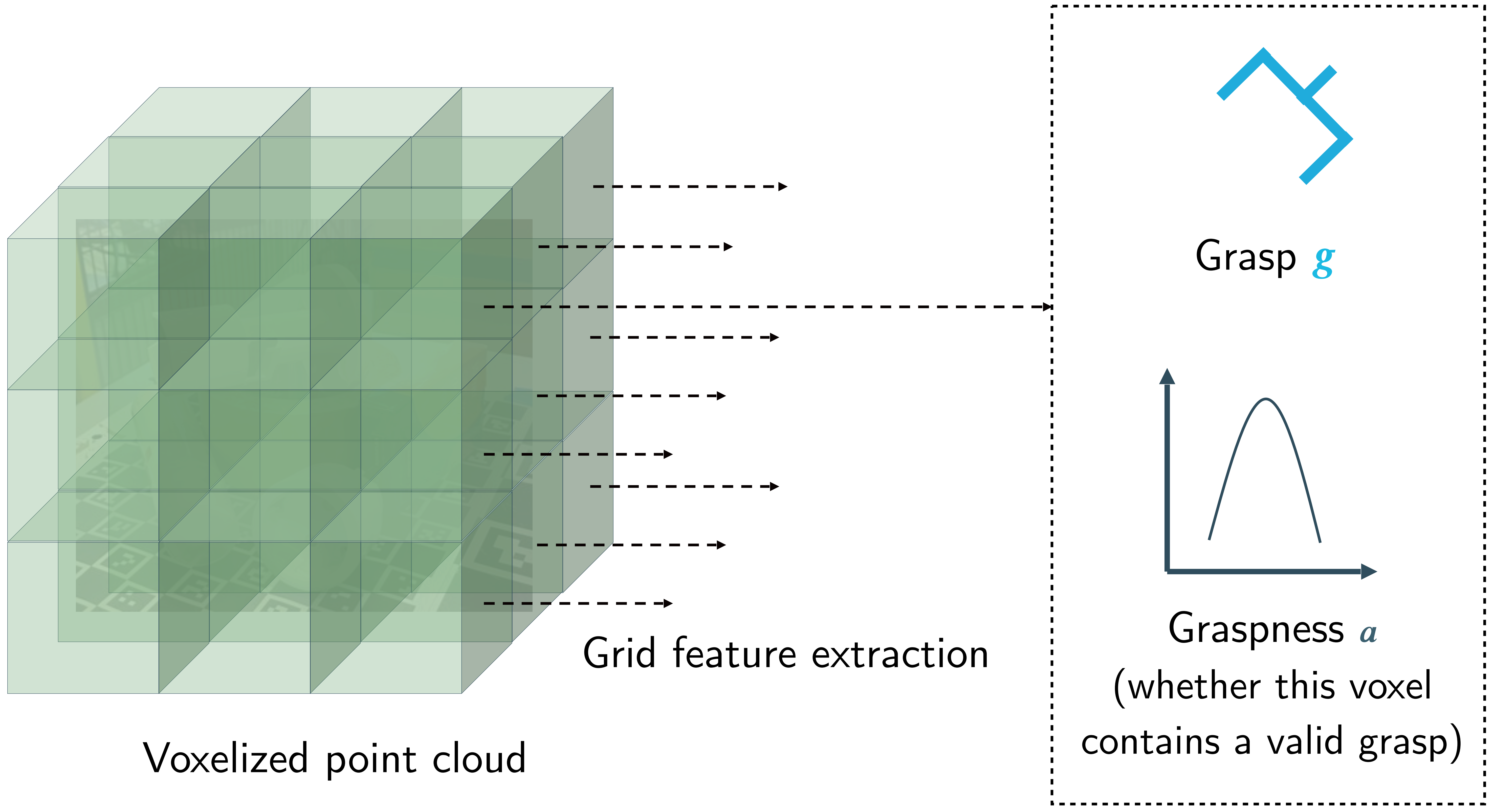




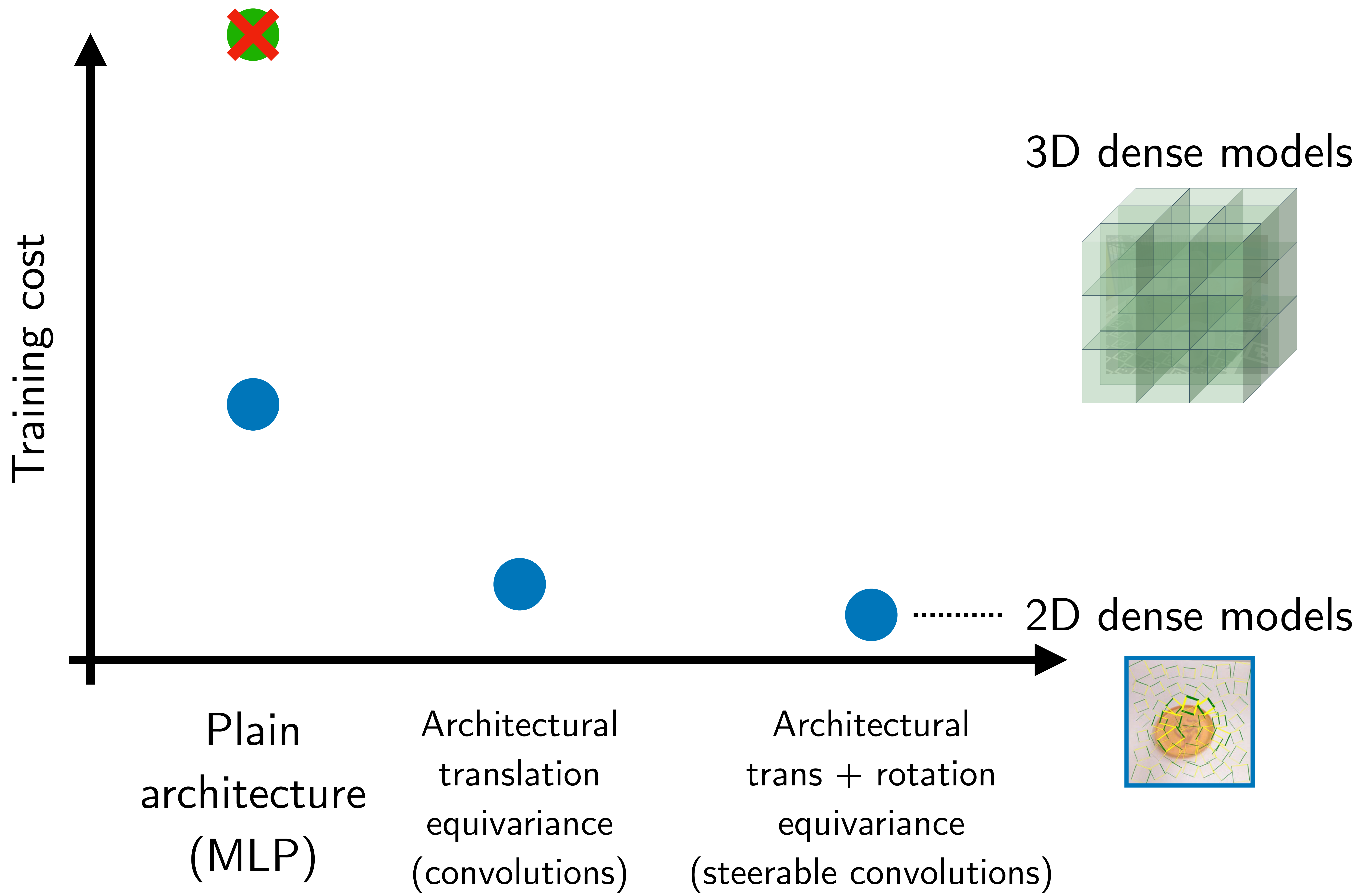
## *Volumetric Grasping* is a popular approach to grasp planning that applies principles of 2D computer vision to 6D grasping



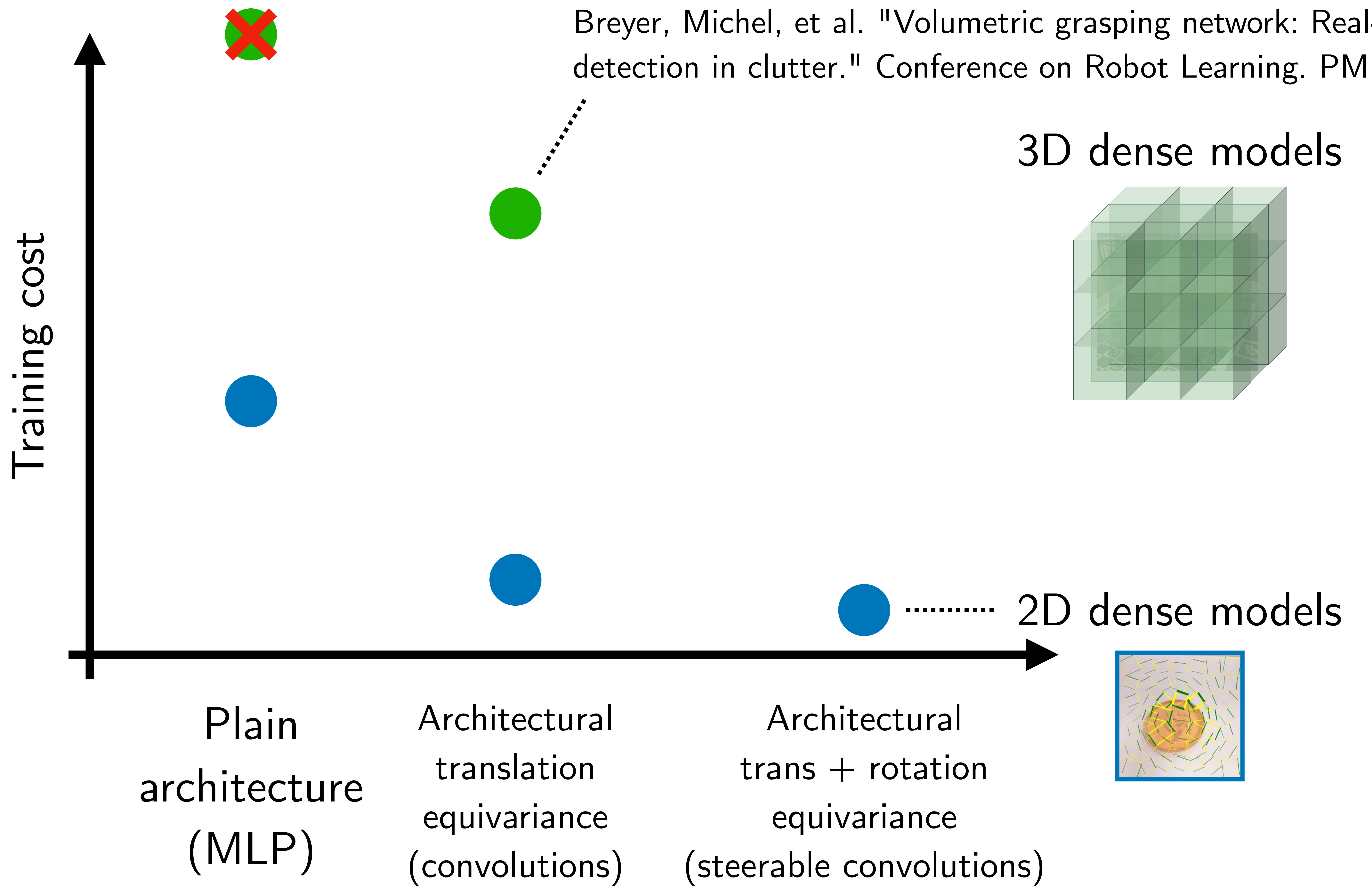
## *Volumetric Grasping* is a popular approach to grasp planning that applies principles of 2D computer vision to 6D grasping



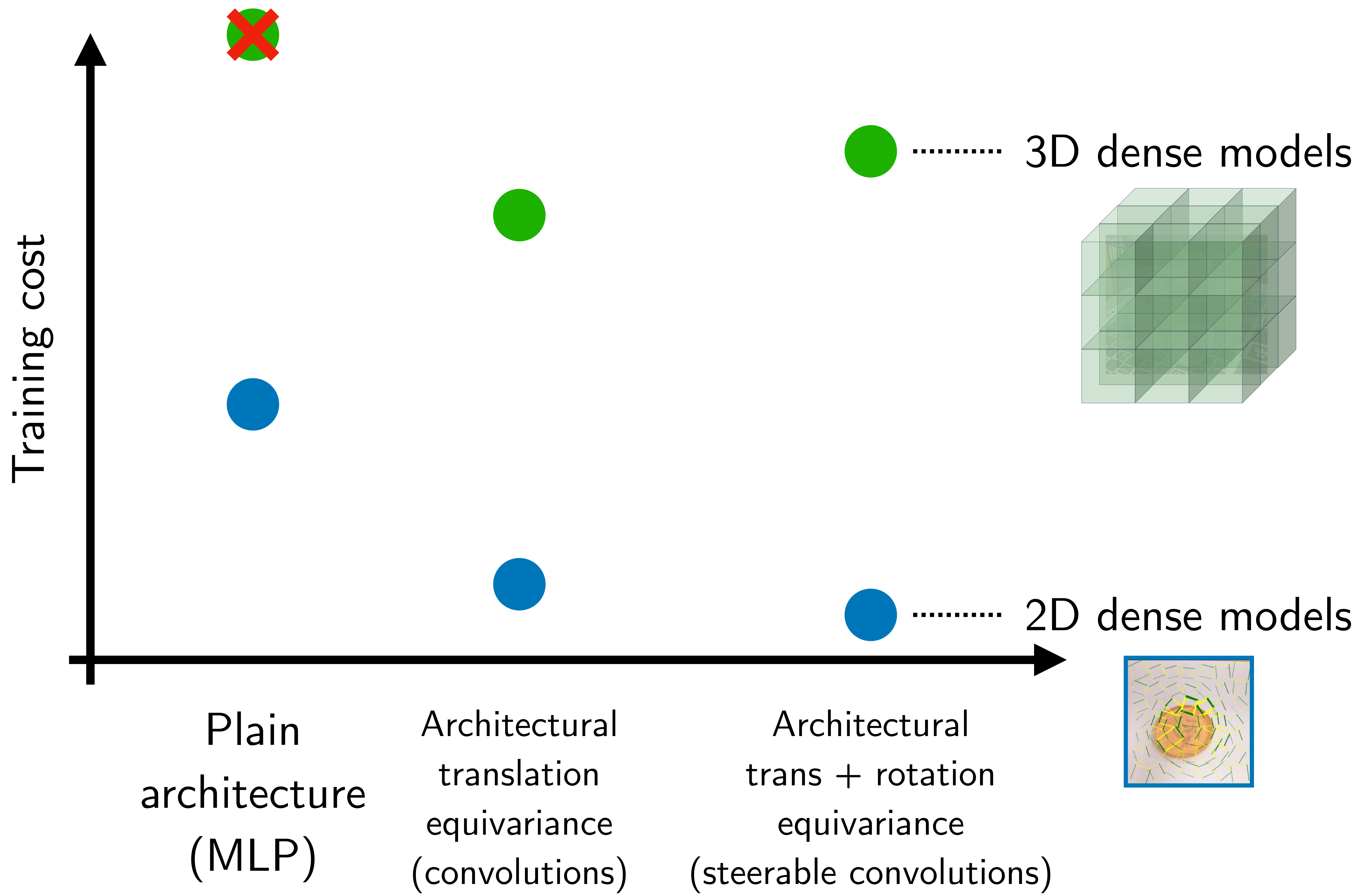
## Volumetric grasping with no architectural equivariance has a prohibitively low sample efficiency


Training cost
3D dense models
2D dense models
Plain architecture (MLP)
Architectural translation equivariance (convolutions)
Architectural trans + rotation equivariance (steerable convolutions)

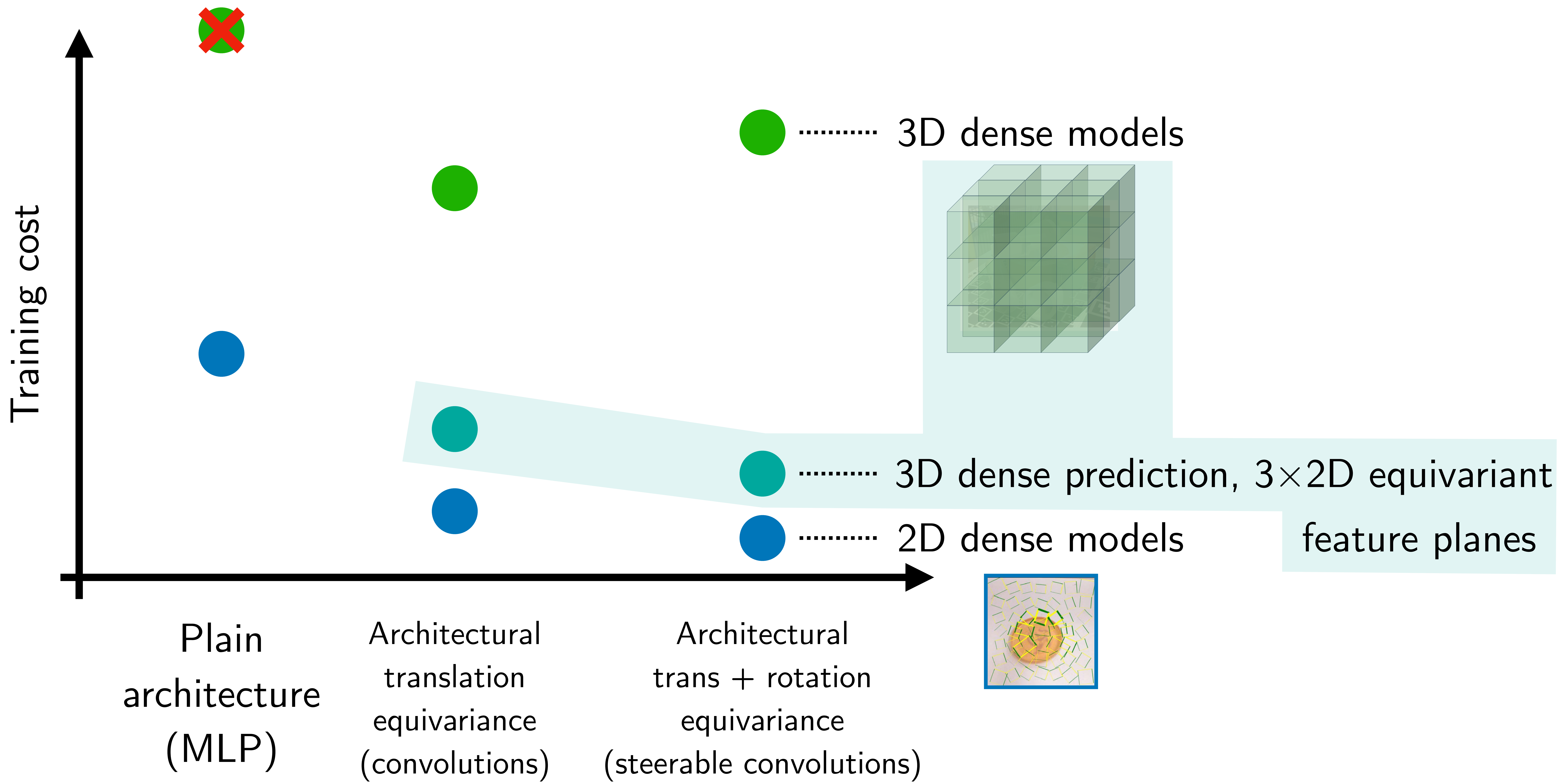
## Volumetric grasping with 3D (translation-equivariant) CNNs is a promising concept



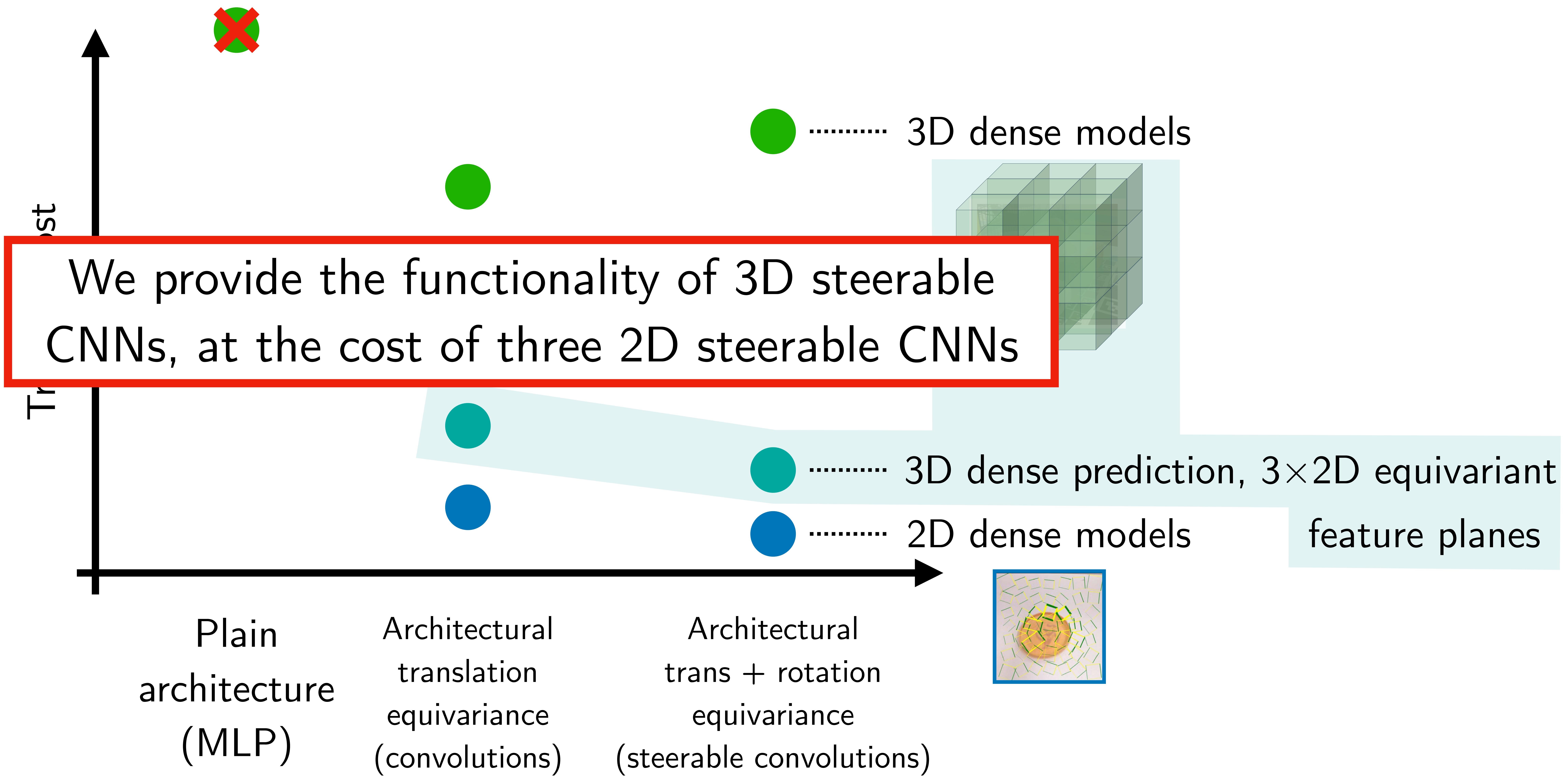
## The increased sample efficiency brought by 3D steerable CNNs is insufficient to justify their computational cost



We propose to achieve 3D translation-rotation equivariance by factorizing the input voxel grid into three orthogonal planar grids, and designing equivariant features in these three planes.

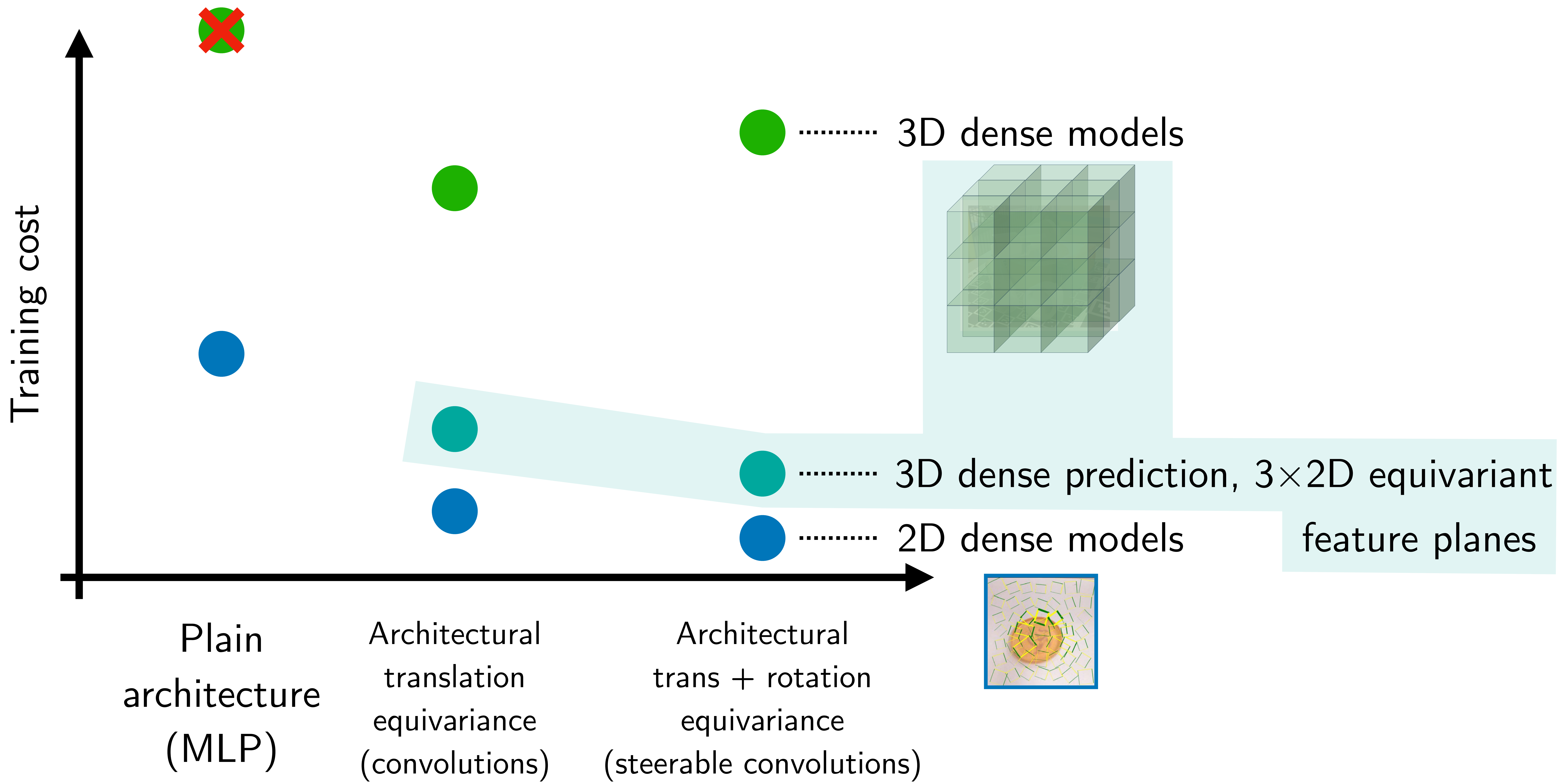


We propose to achieve 3D translation-rotation equivariance by factorizing the input voxel grid into three orthogonal planar grids, and designing equivariant features in these three planes.


3D dense models
3D dense prediction, 3×2D equivariant feature planes
2D dense models
Plain architecture (MLP)
Architectural translation equivariance (convolutions)
Architectural trans + rotation equivariance (steerable convolutions)

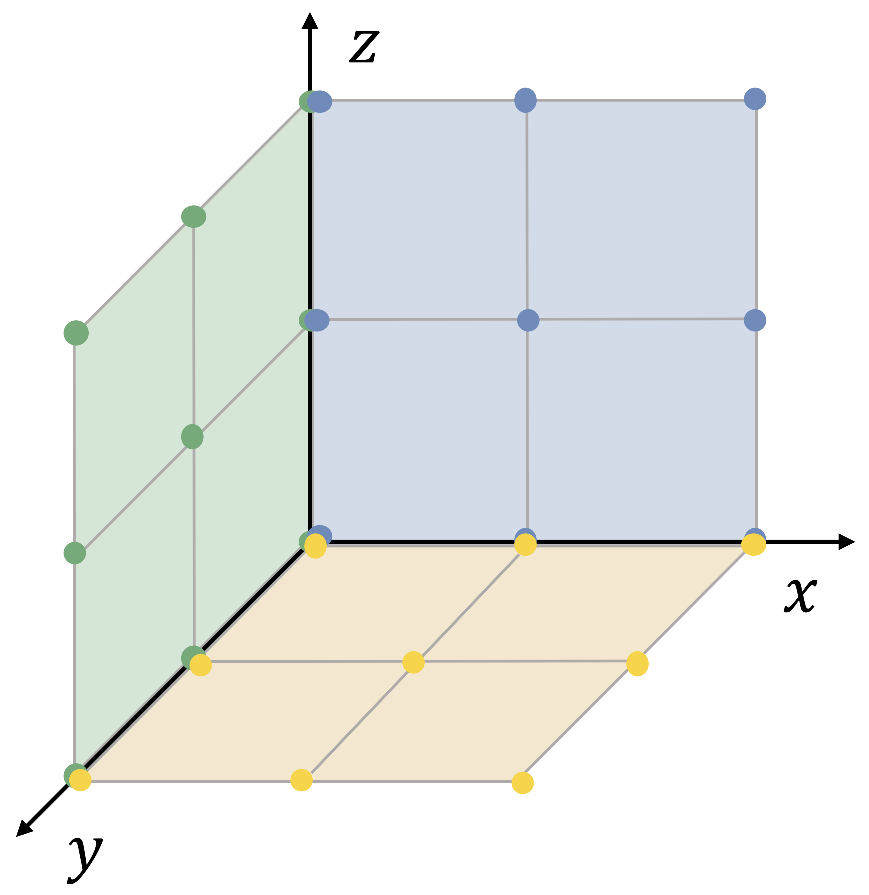
We provide the functionality of 3D steerable CNNs, at the cost of three 2D steerable CNNs

We propose to achieve 3D translation-rotation equivariance by factorizing the input voxel grid into three orthogonal planar grids, and designing equivariant features in these three planes.



# We factorize 3D data into a tri-plane feature grid

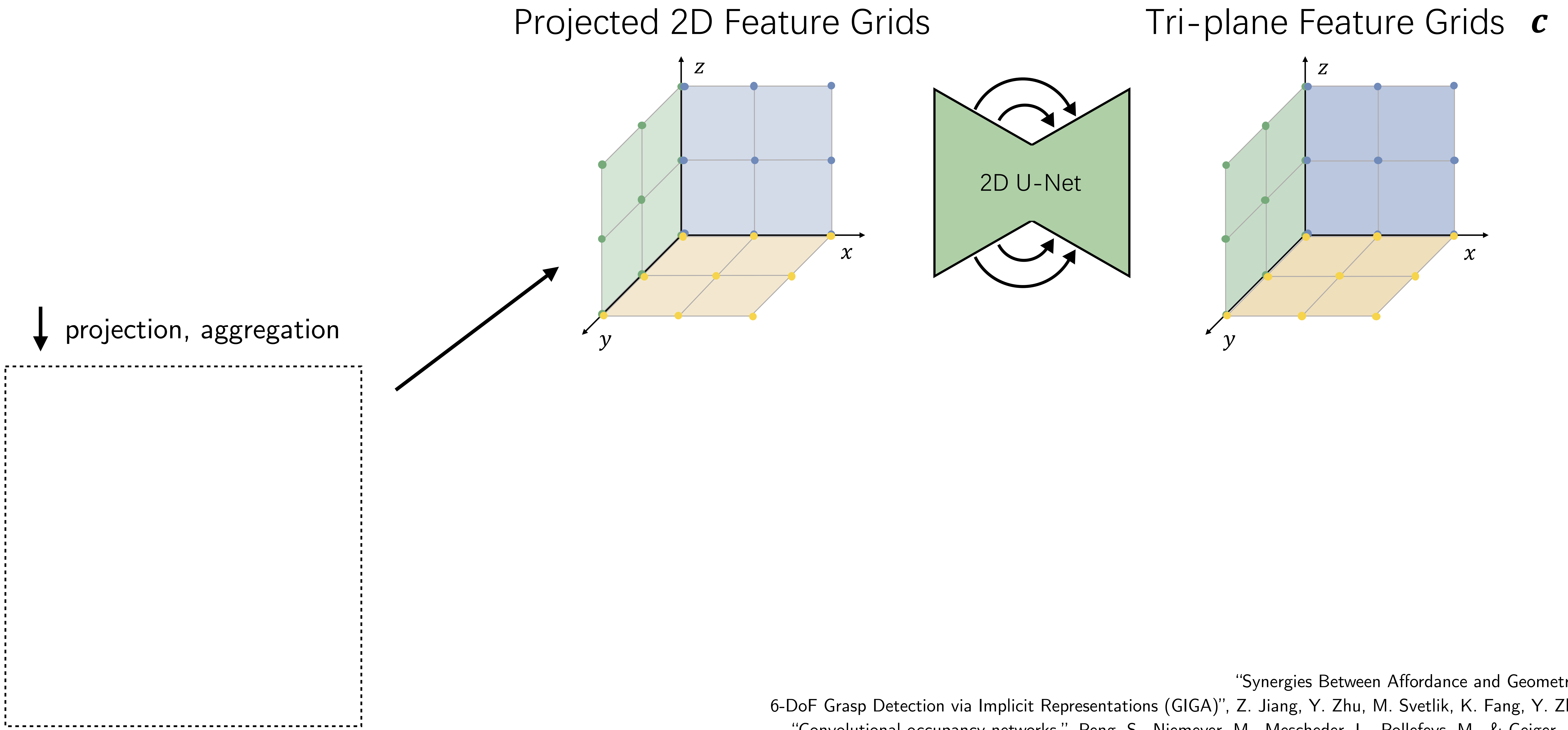
Projected 2D Feature Grids



↓ projection, aggregation

“Synergies Between Affordance and Geometry:
6-DoF Grasp Detection via Implicit Representations (GIGA)”, Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, Y. Zhu
“Convolutional occupancy networks.”, Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., & Geiger, A.

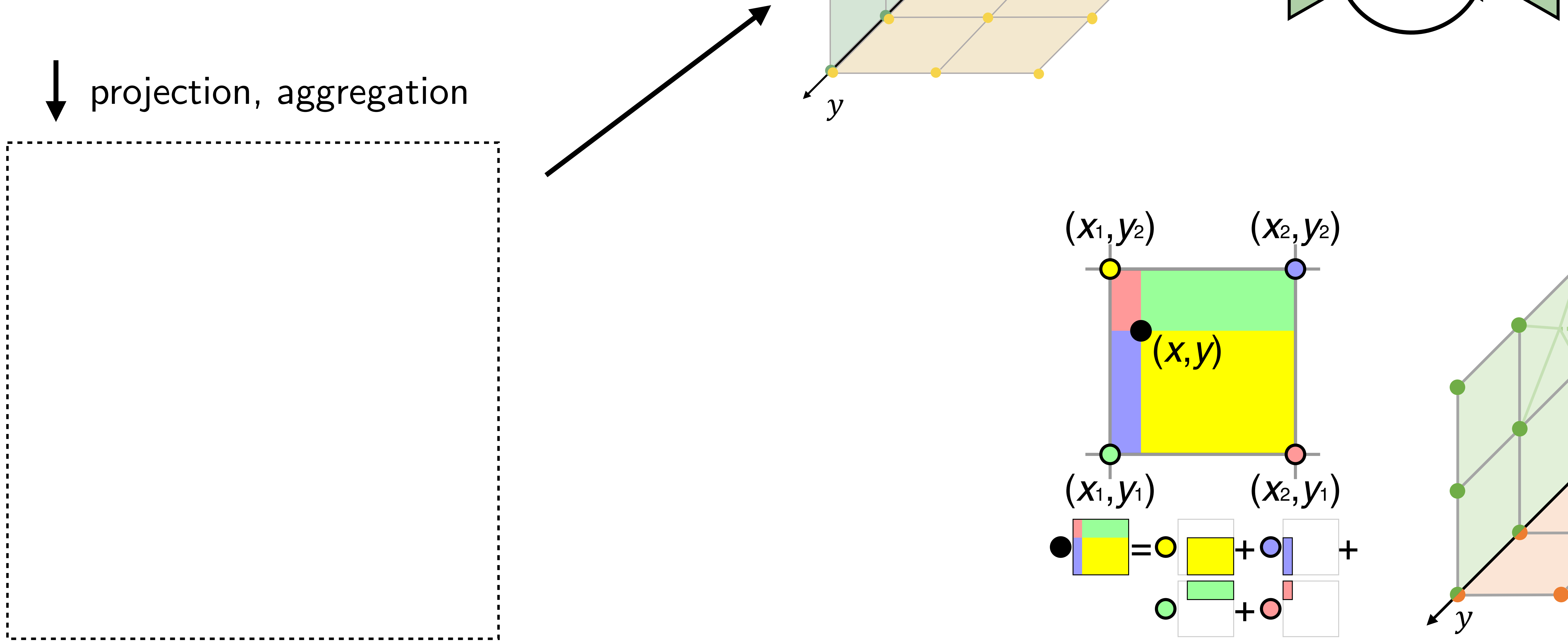
# We extract rich features by applying a 2D UNet to each plane

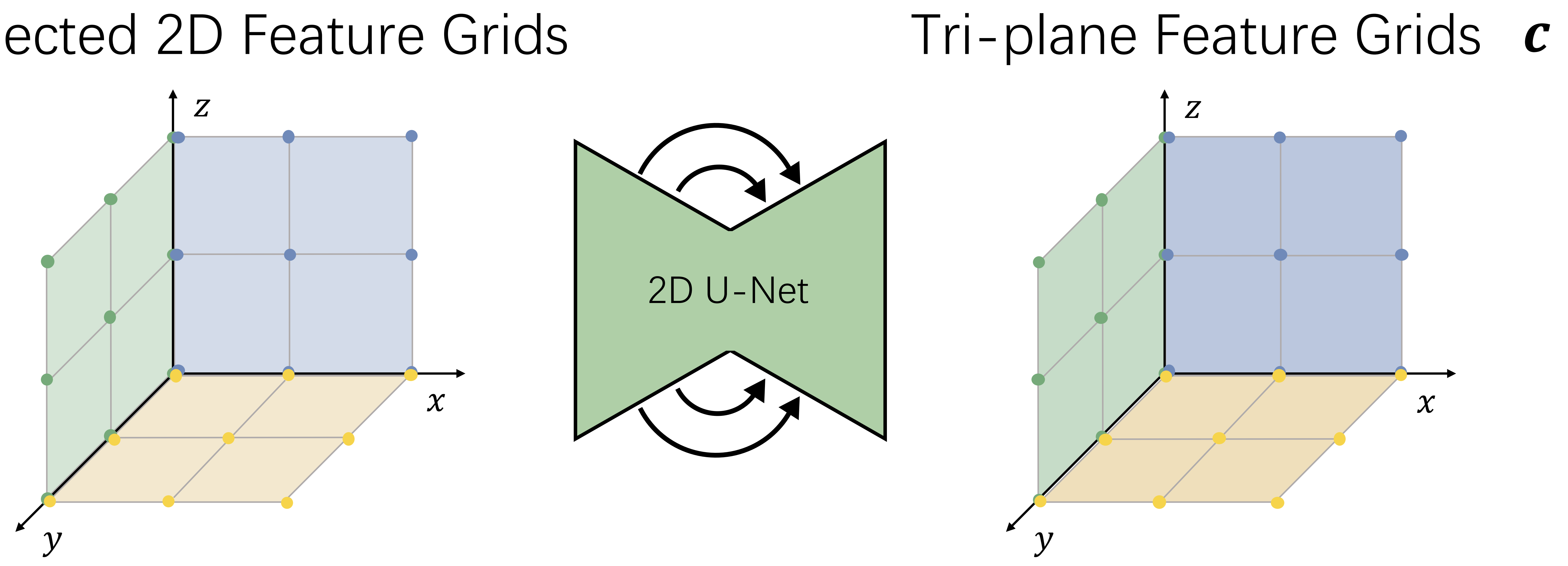

Projected 2D Feature Grids
Tri-plane Feature Grids $\boldsymbol{c}$
projection, aggregation
2D U-Net
$x$
$y$
$z$

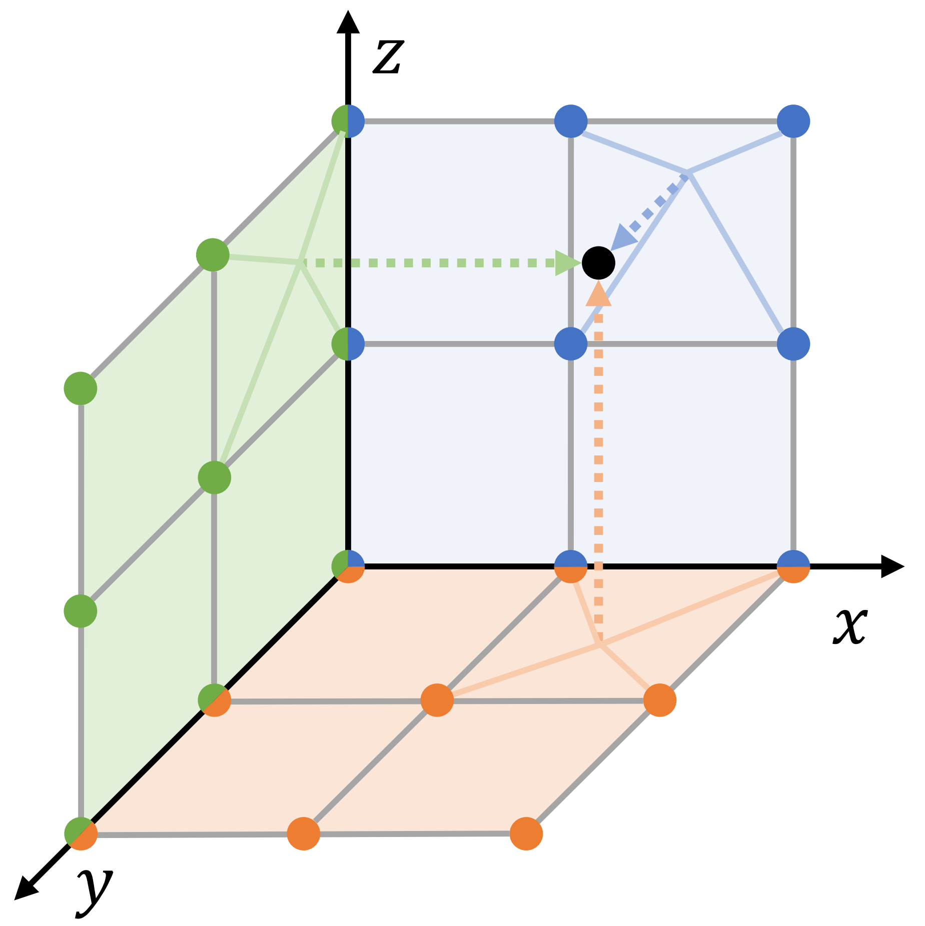
"Synergies Between Affordance and Geometry: 6-DoF Grasp Detection via Implicit Representations (GIGA)", Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, Y. Zhu
"Convolutional occupancy networks.", Peng, S., Niemeyer, M., Mescheder, L., Pollefeys, M., & Geiger, A.

## With bilinear interpolation, we can synthesize a feature at any given 3D point, allowing us to query the model in continuous 3D space

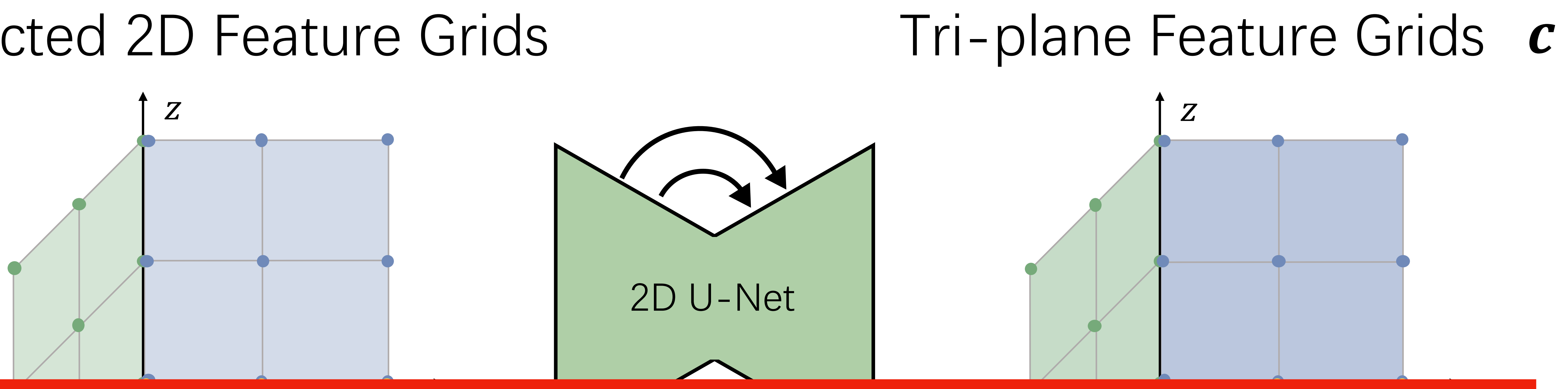
↓ projection, aggregation

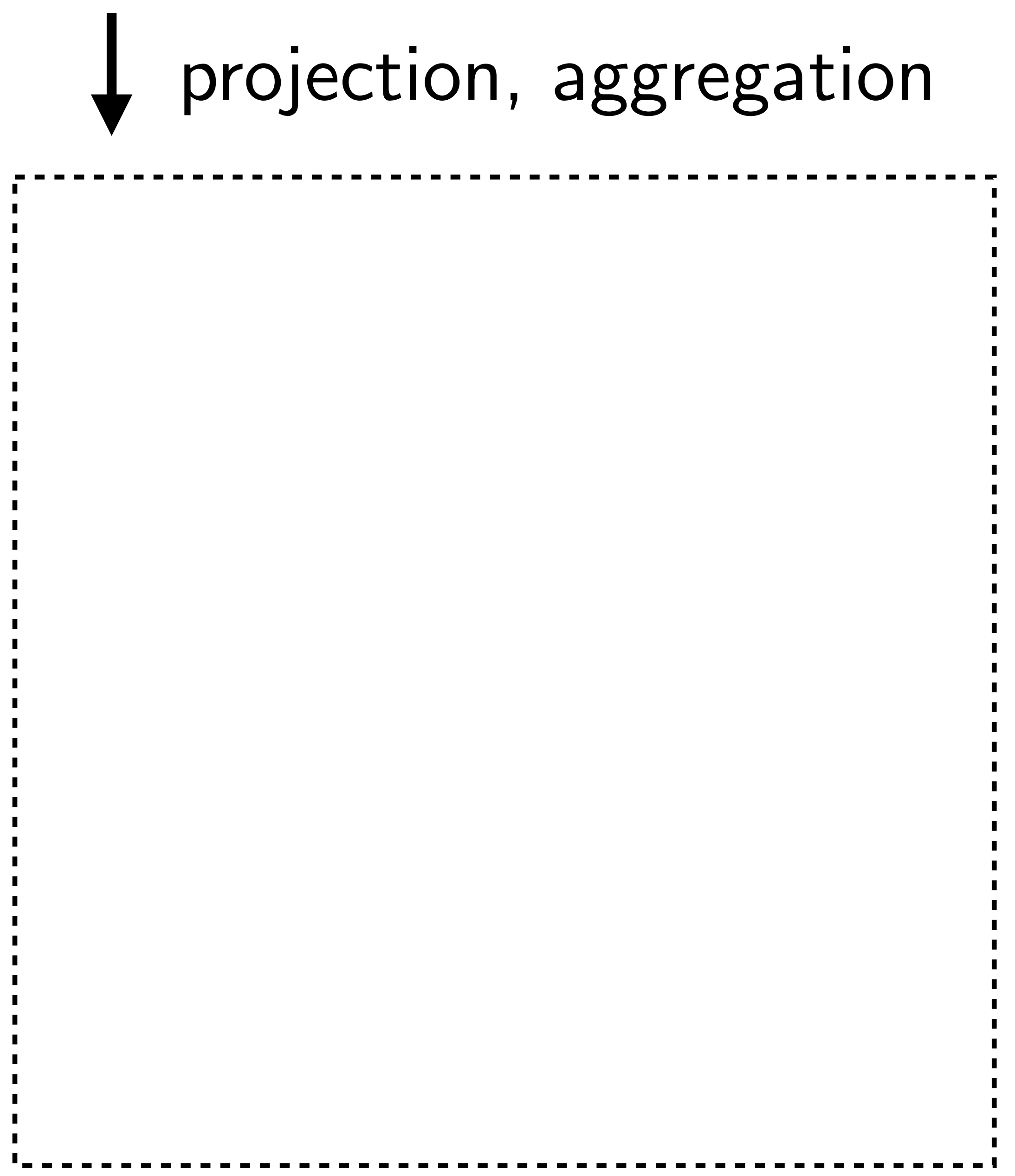




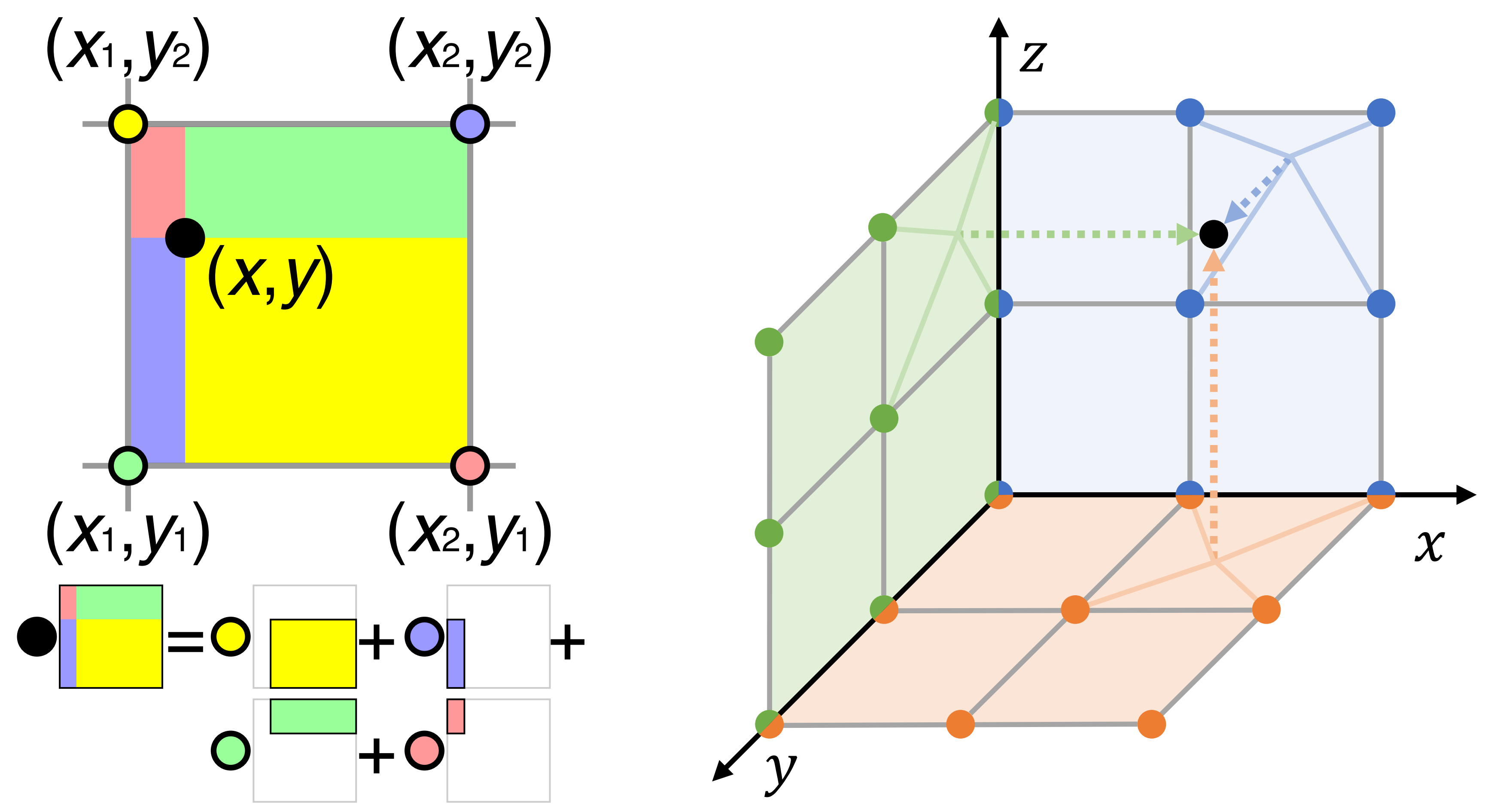


## With bilinear interpolation, we can synthesize a feature at any given 3D point, allowing us to query the model in continuous 3D space

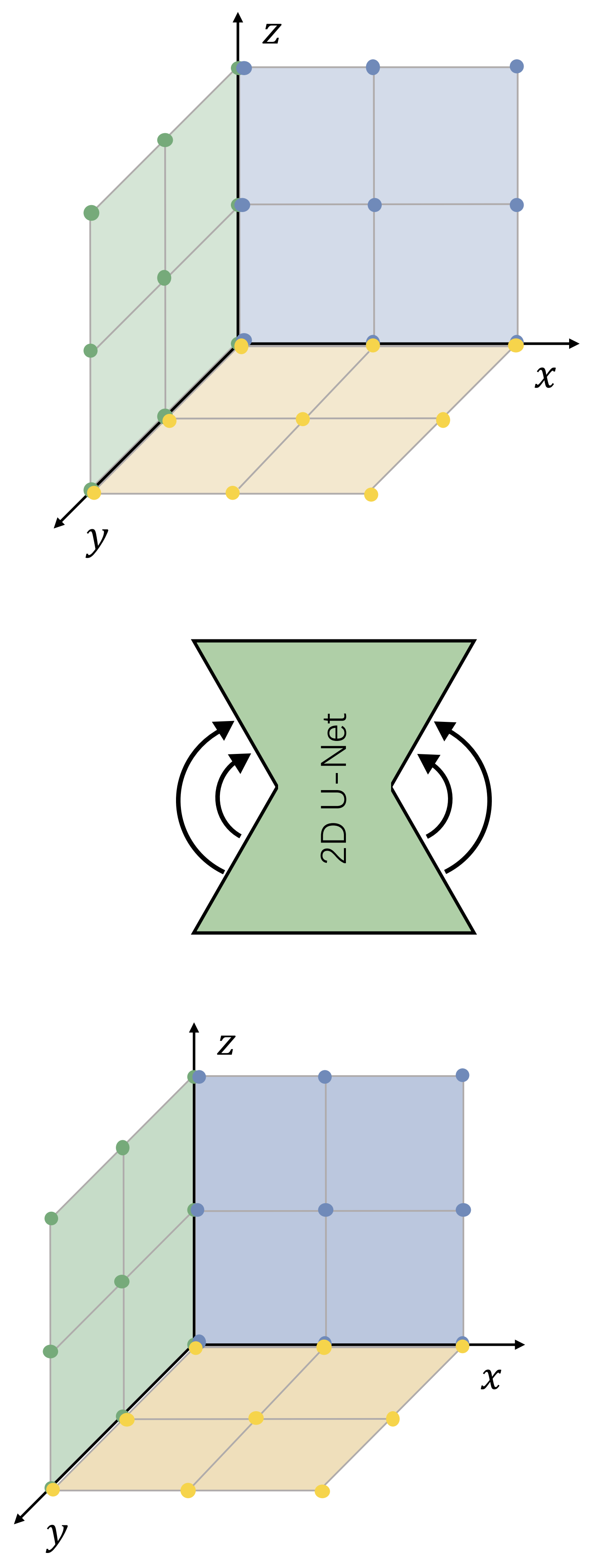

Projected 2D Feature Grids
Tri-plane Feature Grids $\boldsymbol{c}$
z
2D U-Net
z

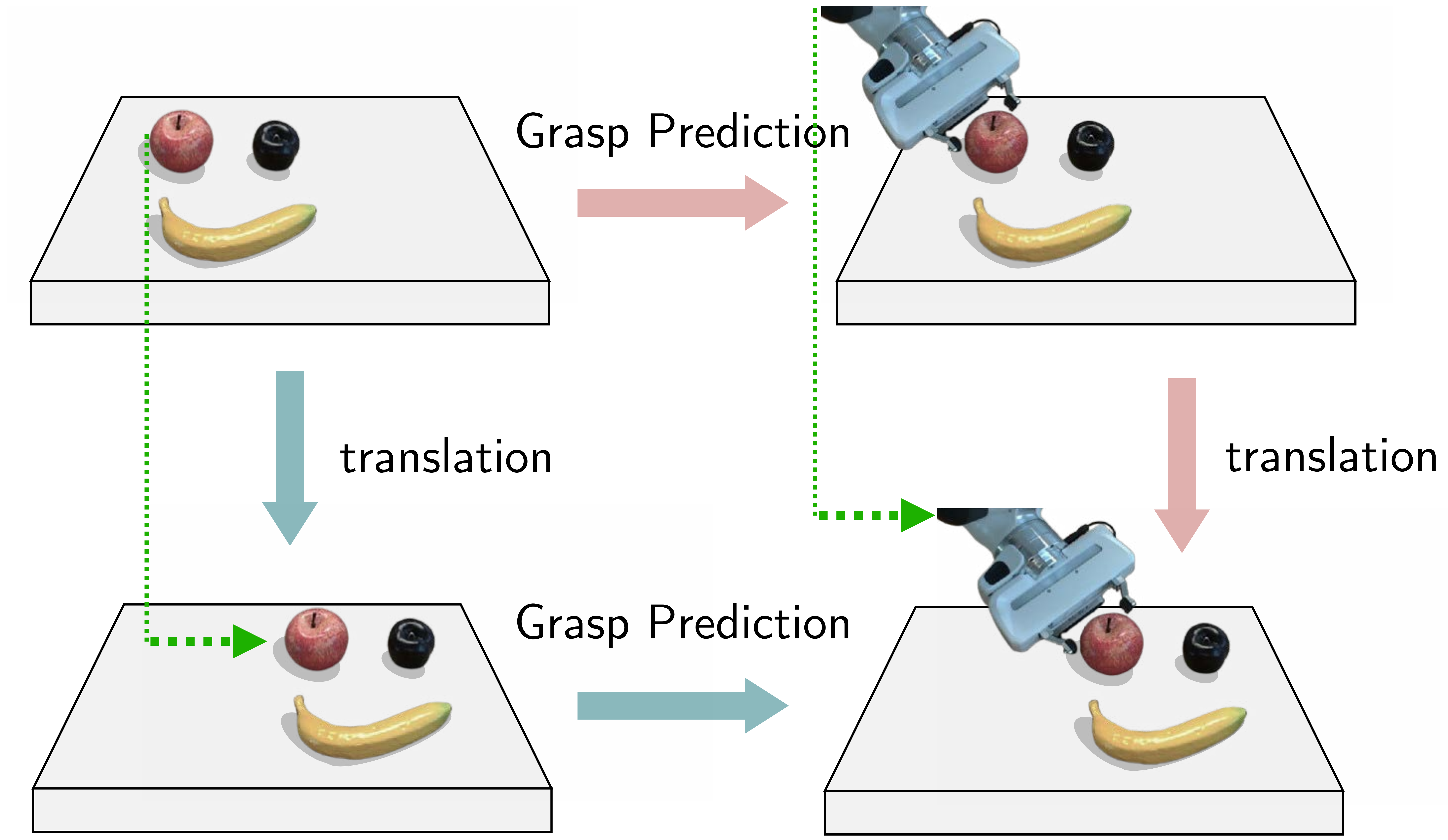


The 3D feature built from interpolated planar features is equivariant to 3D translations



## A grasp model trained atop tri-plane features inherits translation equivariance

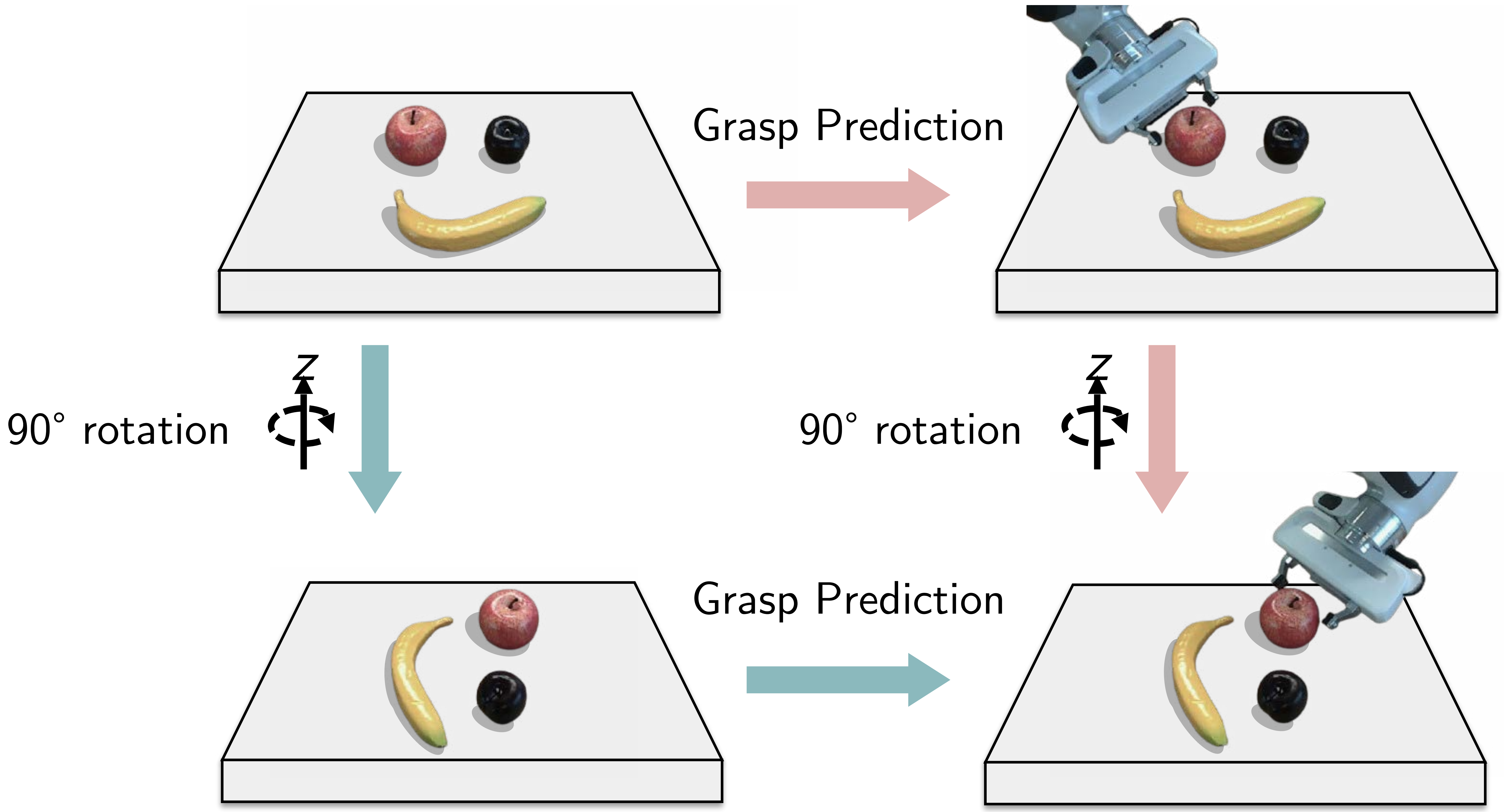




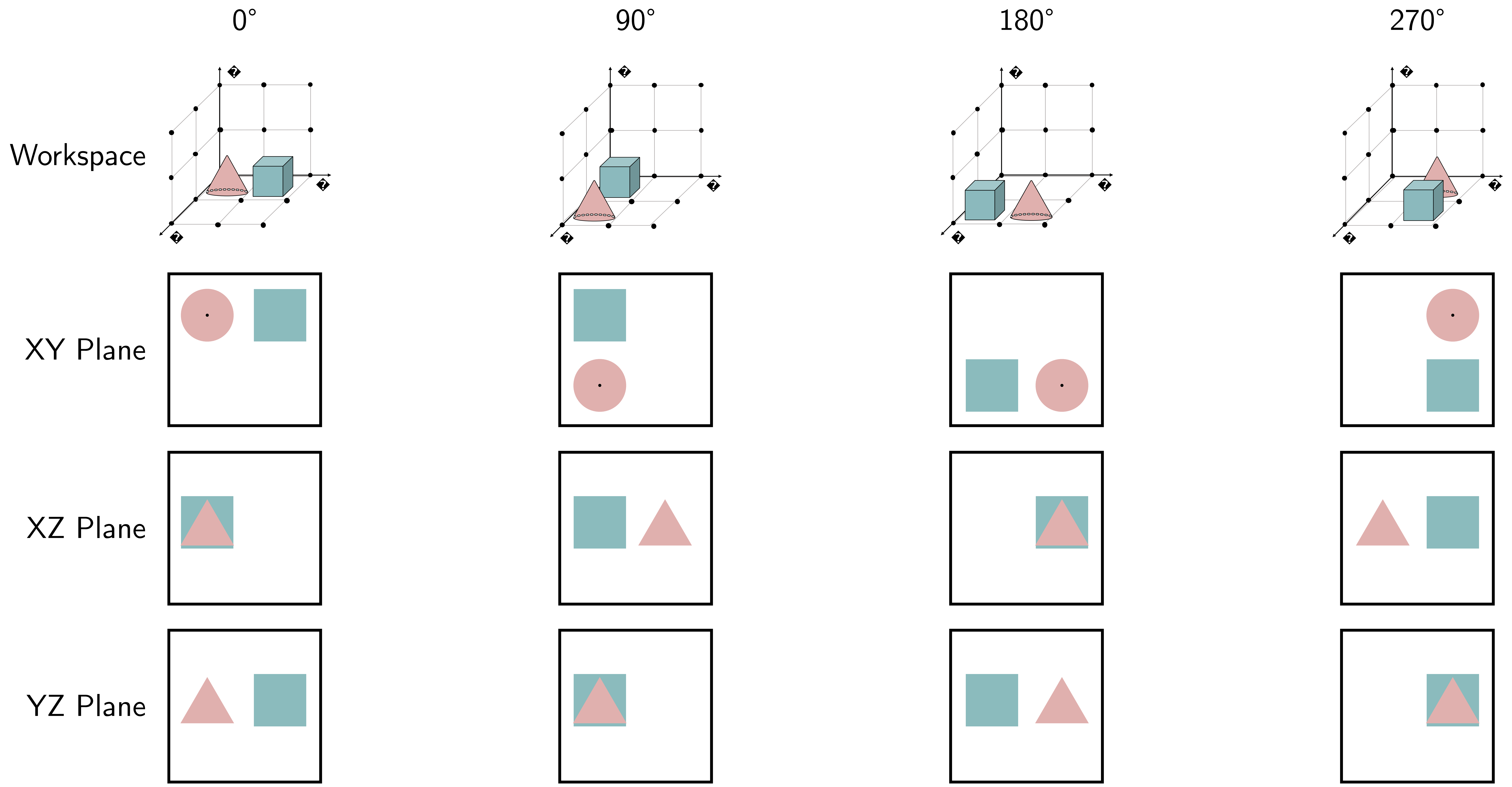
## We opt to design for equivariance to 90° rotations around $Z$

Two relevant observations:

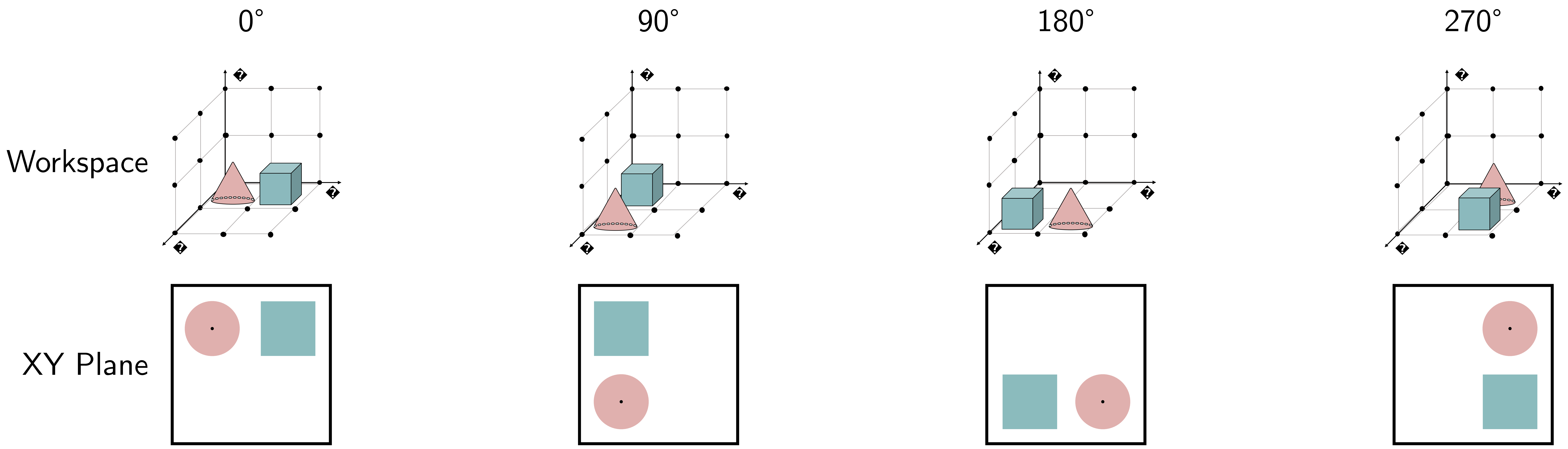
1. The orthogonality of the three feature planes would make it particularly convenient to design equivariance to 90° rotations,
2. Objects set on a table rotate more often around vertical axis.



# We must design planar features that are equivariant to 90° rotations of the workspace around $Z$

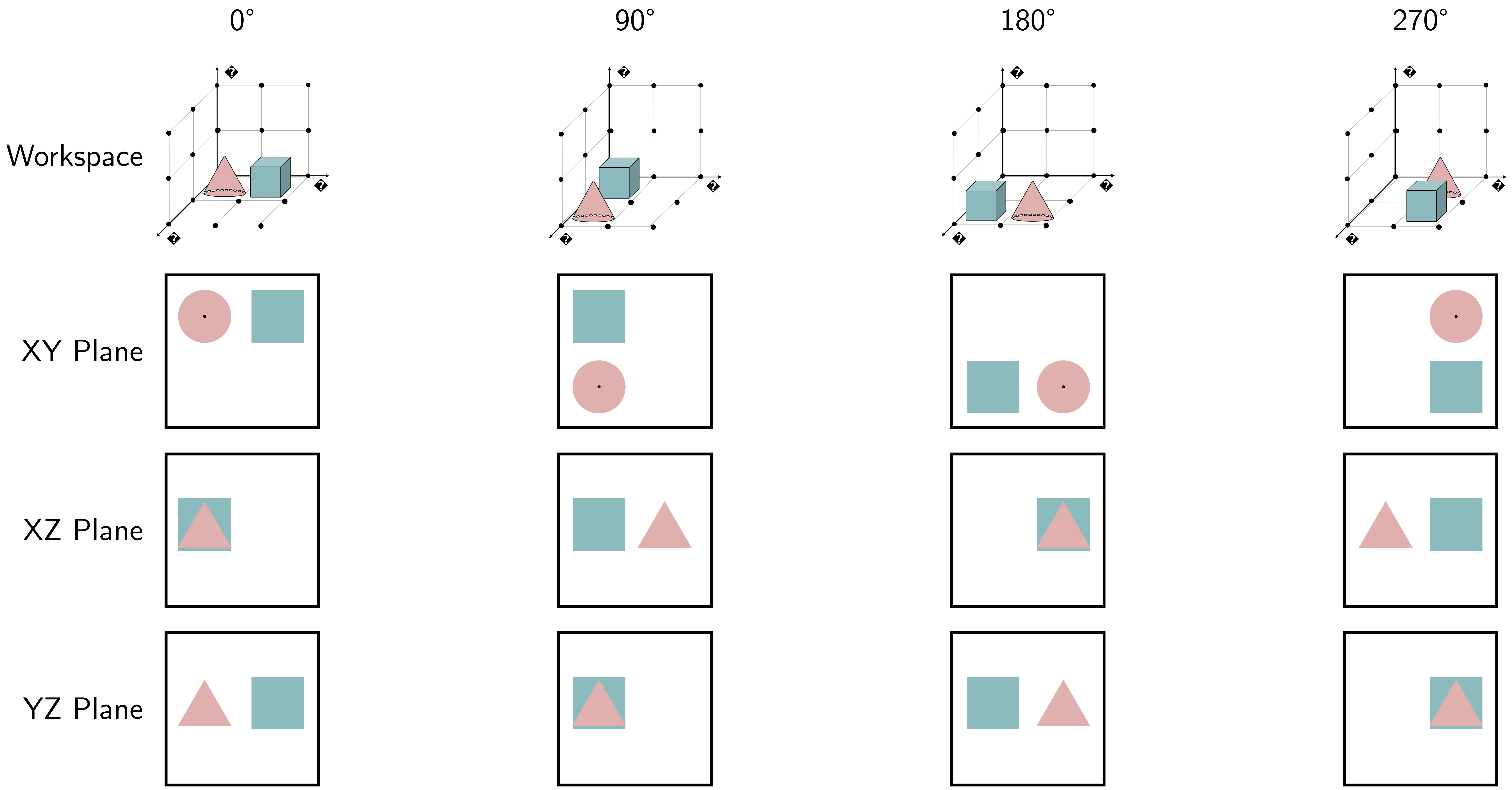


## We must design planar features that are equivariant to 90° rotations of the workspace around $Z$

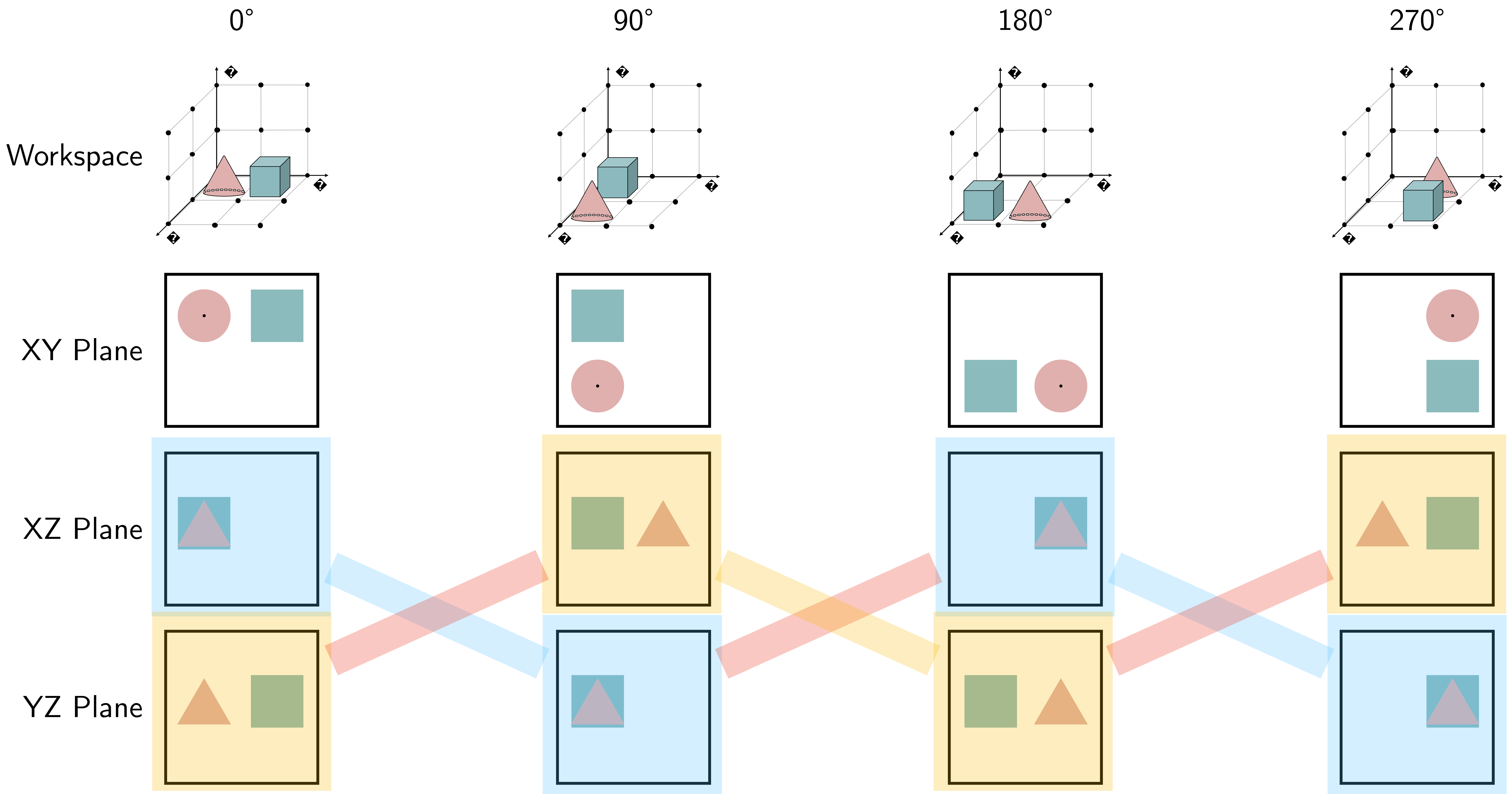


- For the XY plane, equivariance to a 90° rotation around $Z$ is equivalent to equivariance to in-plane rotations.
- It can be achieved by equipping the XY UNet with $C_4$-equivariant steerable convolutions.

## For XZ and XZ, equivariance to 90° rotations around Z is not equivalent to in-plane rotations

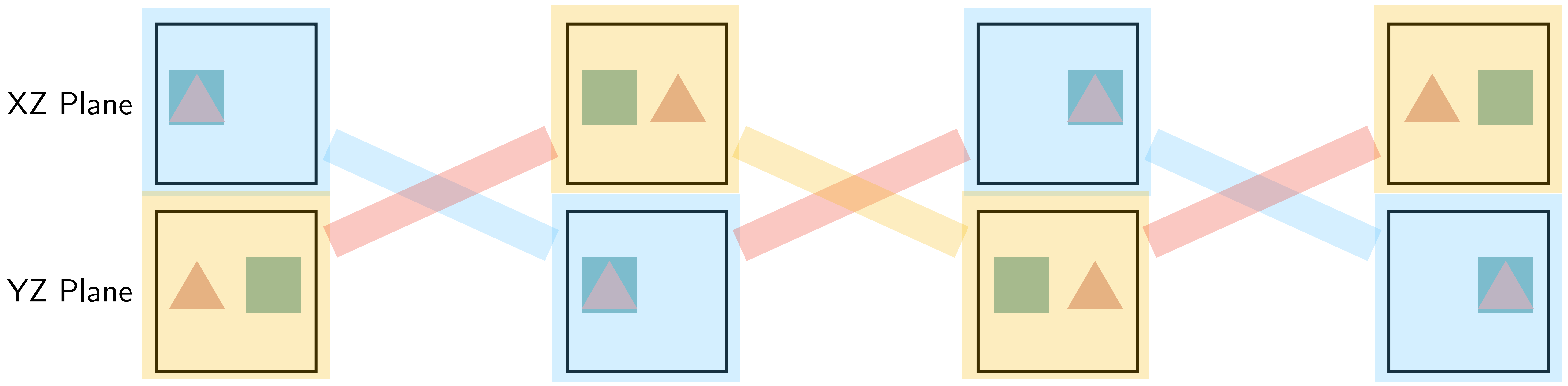


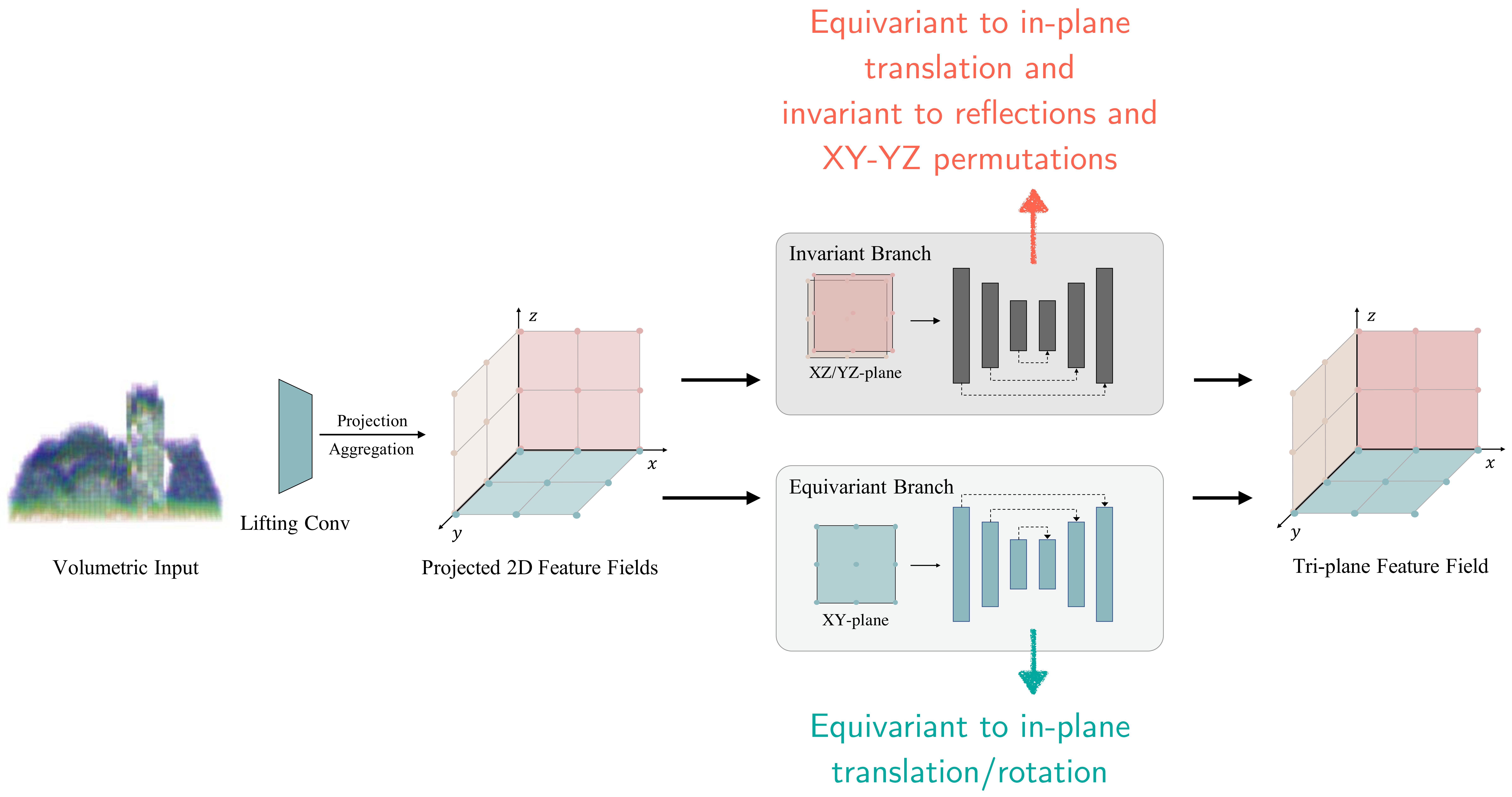
# The predictable effect on XZ and YZ of a 90° $Z$-rotations is a permutation between XZ and YZ, occasionally accompanied by a reflection

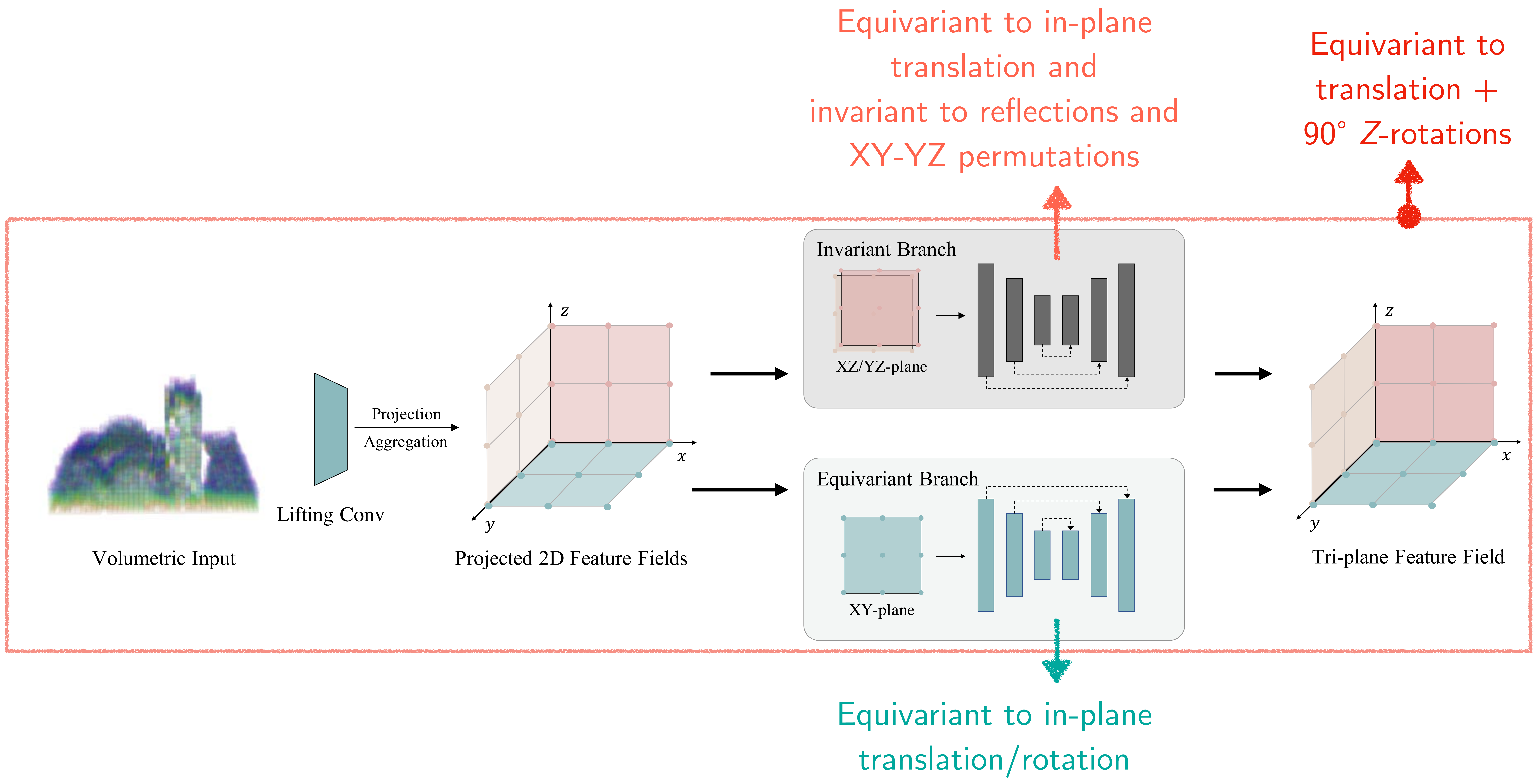


## The predictable effect on XZ and YZ of a 90° $Z$-rotations is a permutation between XZ and YZ, occasionally accompanied by a reflection

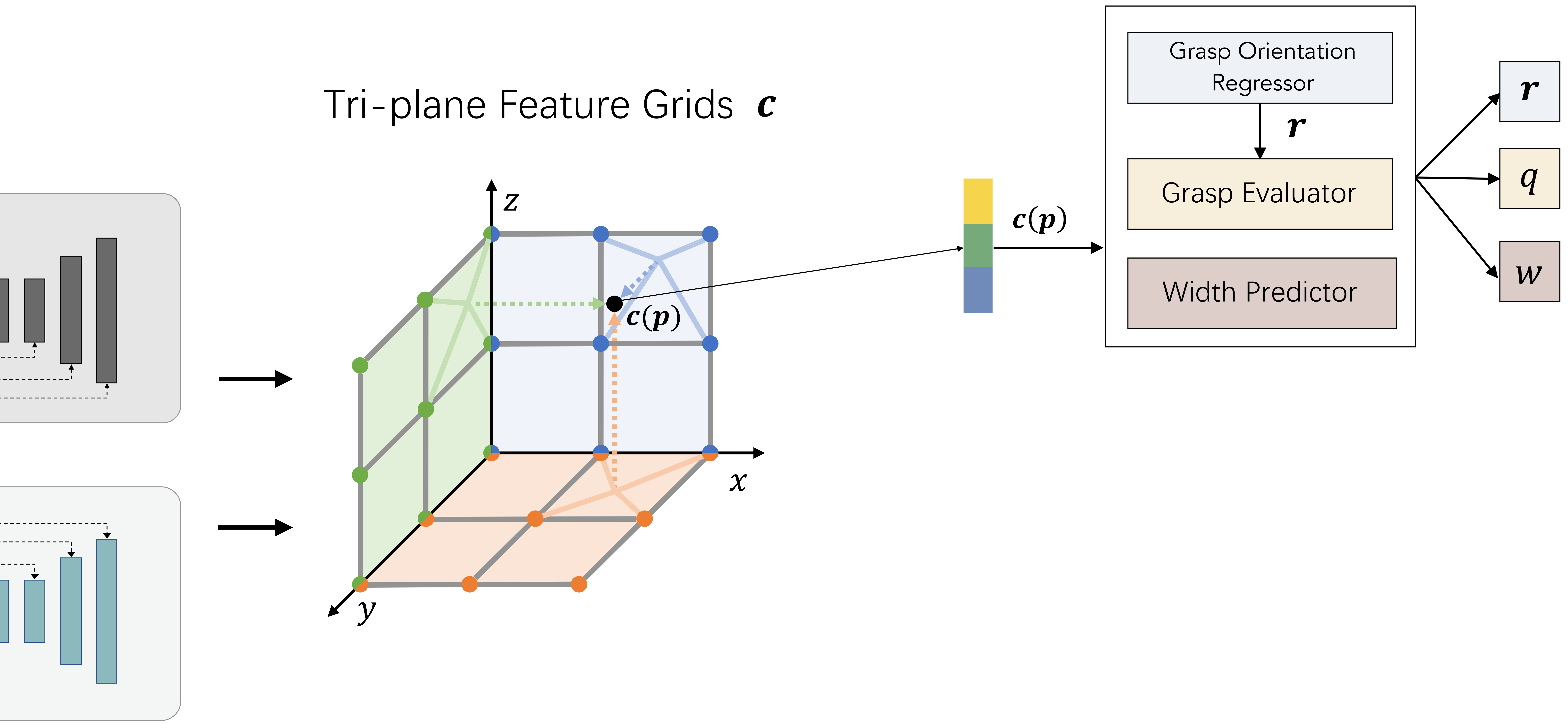
- **If** we design the XZ and YZ UNets for reflection invariance,
- **Then** the pairwise sum of reflection-invariant XZ and YZ features is invariant to the permutations induced by 90° $Z$-rotations.
- **Conclusion**: Downstream tasks (a grasp planner) will use the sum of reflection-invariant XZ/YZ features.




Equivariant to in-plane translation and invariant to reflections and XY-YZ permutations
Invariant Branch
XZ/YZ-plane
Projection
Aggregation
$z$
$x$
$y$
Lifting Conv
Volumetric Input
Projected 2D Feature Fields
Equivariant Branch
XY-plane
Tri-plane Feature Field
Equivariant to in-plane translation/rotation

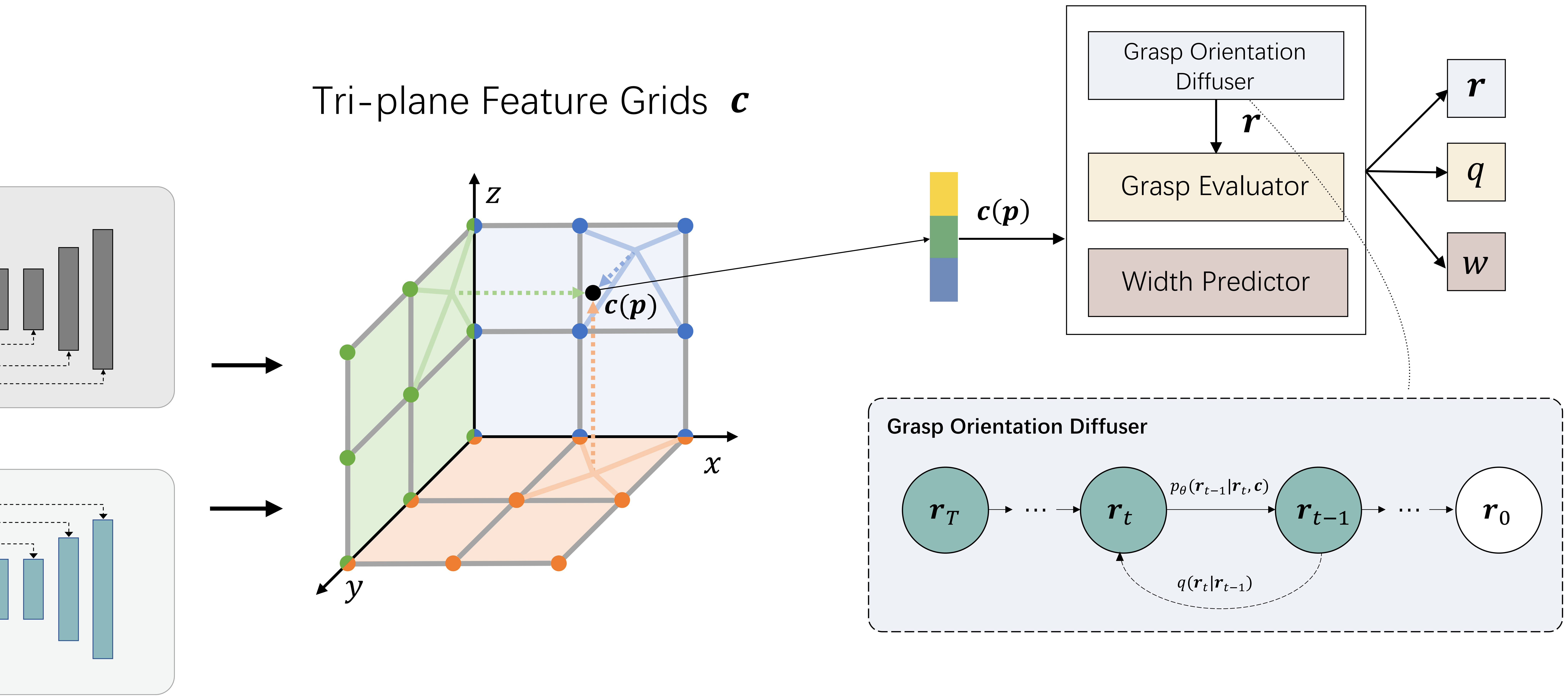

Equivariant to in-plane translation and invariant to reflections and XY-YZ permutations
Equivariant to translation + 90° Z-rotations
Invariant Branch
XZ/YZ-plane
Equivariant Branch
XY-plane
Volumetric Input
Lifting Conv
Projection
Aggregation
Projected 2D Feature Fields
Tri-plane Feature Field
x
y
z
Equivariant to in-plane translation/rotation

## EquiGIGA: Given a grasp location *p*, we ground models of gripper rotation *r*, grasp quality *q* and gripper width *w* in the tri-plane feature *c(p)*



Tri-plane Feature Grids $\boldsymbol{c}$
$z$
$x$
$y$
$\boldsymbol{c}(\boldsymbol{p})$
$\boldsymbol{c}(\boldsymbol{p})$
Grasp Orientation Regressor
$\boldsymbol{r}$
Grasp Evaluator
Width Predictor
$\boldsymbol{r}$
$q$
$w$

EquiIGD: Grasp rotations are encoded with a diffusion model, which effectively captures multi-modal rotation distributions

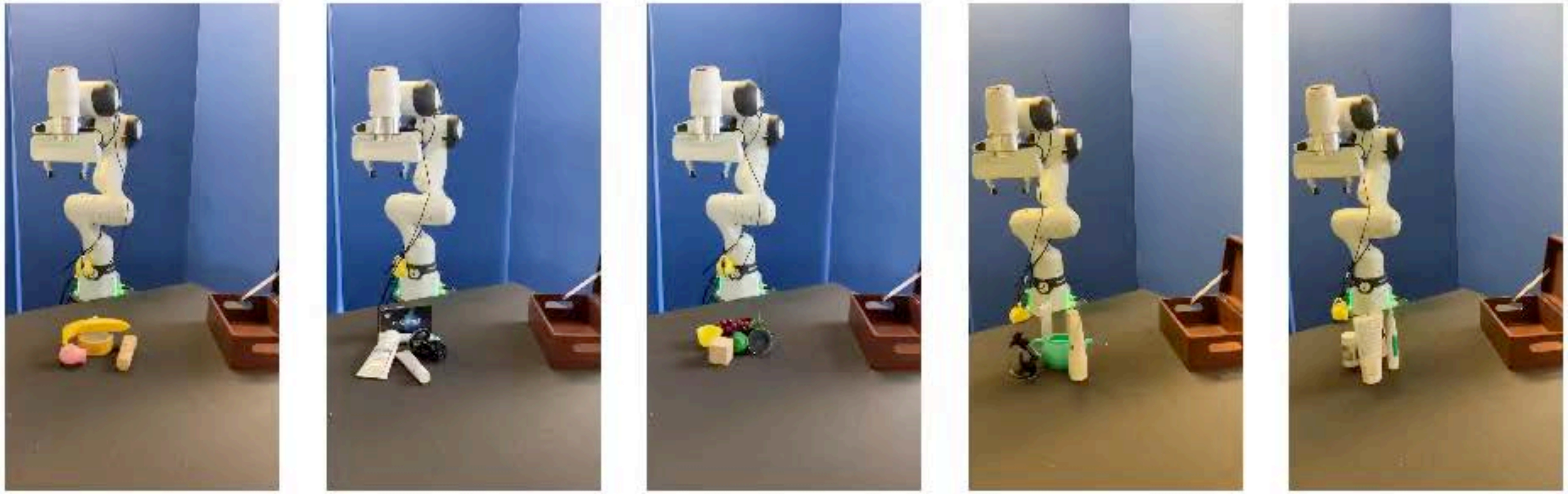

Tri-plane Feature Grids $c$
$z$
$x$
$y$
$c(p)$
$c(p)$
Grasp Orientation Diffuser
$r$
Grasp Evaluator
Width Predictor
$r$
$q$
$w$
Grasp Orientation Diffuser
$r_T$
$r_t$
$p_\theta(r_{t-1}|r_t, c)$
$r_{t-1}$
$r_0$
$q(r_t|r_{t-1})$

| Method | Packed |  | Pile |  | Latency (ms) |
|---|---|---|---|---|---|
|  | GSR (%) | DR (%) | GSR (%) | DR (%) |  |
| VGN* [1] | 72.5±2.6 | 76.7±1.7 | 59.3±2.9 | 43.5±2.9 | **9** |
| GIGA* [3] | 84.8±2.2 | 85.1±2.5 | 69.5±1.3 | 49.0±3.4 | **24** |
| GraspNet-1B Baselines* [12] | 49.9±2.3 | 40.1±2.2 | 50.2±4.2 | 30.0±2.3 | 77 |
| GSNet* [11] | 67.8±2.5 | 60.1±3.2 | 58.3±3.8 | 51.3±4.6 | 156 |
| GPD* [44] | 41.8±2.9 | 34.1±3.4 | 22.7±1.1 | 9.0±0.7 | 2138 |
| 6DoF-GraspNet* [16] | 17.9±0.8 | 11.9±0.9 | 15.5±2.9 | 6.9±1.1 | 2232 |
| SE(3)-Dif* [15] | 7.2±1.5 | 4.3±1.0 | 7.6±1.8 | 3.0±0.8 | 5691 |
| EdgeGraspNet† [13] | 54.1±2.1 | 54.0±2.7 | 50.5±3.7 | 43.0±4.8 | 843/685 |
| VN-EdgeGraspNet† [13] | 60.6±2.2 | 60.1±3.8 | 55.0±2.1 | 50.1±4.0 | 1174/953 |
| ICGNet† [20] | 60.3±4.1 | 64.5±5.9 | 57.3±1.5 | 51.7±3.3 | 806 |
| DexGraspNet2† [21] | 51.6±2.5 | 53.9±4.3 | 39.7±1.3 | 30.9±2.2 | 2781 |
| OrbitGrasp† [6] | 71.1±1.8 | 72.8±1.6 | 69.3±2.1 | 64.7±3.3 | 3193 |
| IGD* (*N*=1) [2] | 92.9±1.8 | 86.7±1.8 | 68.2±1.9 | 50.6±1.5 | 217 |
| IGD* (*N*=11) [2] | 91.2±0.9 | 88.8±1.5 | 71.8±2.2 | 55.7±2.6 | 1823 |
| EquiGIGA | **96.8±1.0** | 88.6±1.3 | 76.6±2.5 | **76.4±2.9** | 65 |
| EquiGIGA (HR) | 93.1±1.2 | **91.8±1.3** | **78.6±1.0** | 75.5±1.3 | 200 |
| EquiIGD | **97.4±1.6** | 91.4±1.4 | **78.6±2.1** | **78.0±3.0** | 147 |
| EquiIGD (HR) | 96.0±0.8 | **92.4±1.4** | 74.9±1.2 | 73.0±0.8 | 240 |

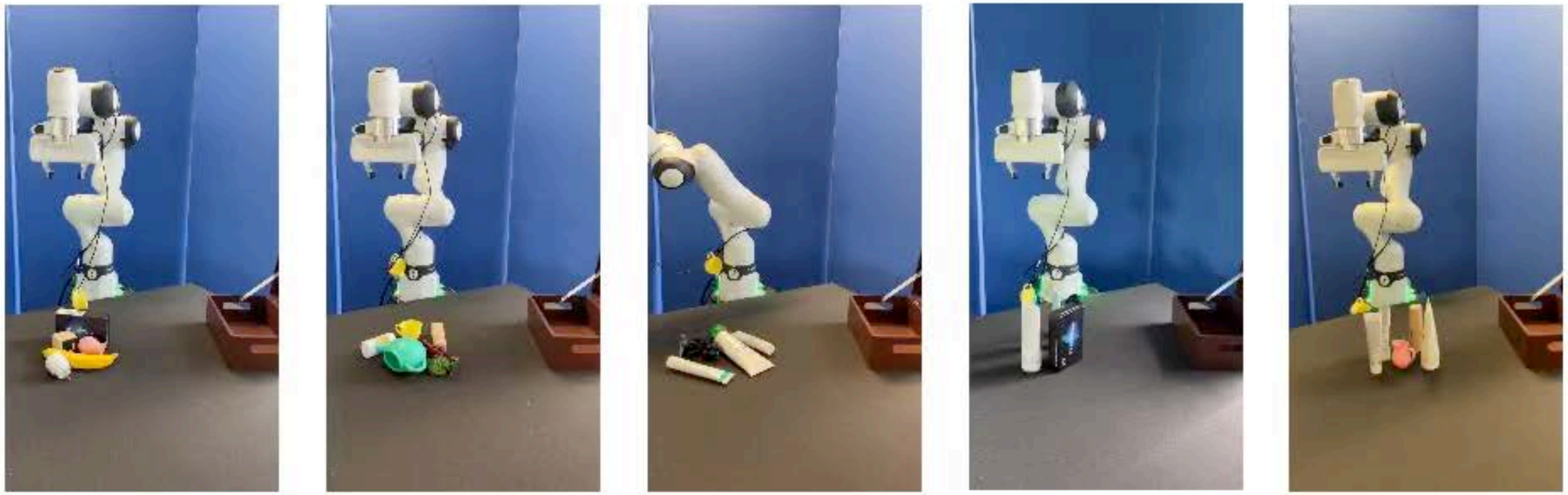
(a) Experimental setup (b) Packed scene (c) Pile scene (d) Adv scene

| Method | Packed | | Pile | | Adv | |
|---|---|---|---|---|---|---|
| | GSR (%) | DR (%) | GSR (%) | DR (%) | GSR (%) | DR (%) |
| GIGA [3] | 76.7 (66/86) | 88.0 | 61.1 (44/72) | 58.7 | 72.5 (66/99) | 88.0 |
| EdgeGraspNet [13] | 73.4 (58/79) | 77.3 | 62.1 (41/66) | 54.7 | 72.2 (57/79) | 76.0 |
| VN-EdgeGraspNet [13] | 71.3 (57/80) | 76.0 | 67.7 (44/65) | 58.7 | 79.5 (58/73) | 77.3 |
| IGD [2] | 78.0 (64/82) | 85.3 | 63.0 (51/88) | 68.0 | 78.2 (61/78) | 81.3 |
| ICGNet [20] | 72.2 (57/79) | 76.0 | 71.1 (54/76) | 72.0 | 69.9 (51/73) | 68.0 |
| EquiGIGA | 82.7 (67/81) | 89.3 | 79.3 (65/82) | 86.7 | 85.6 (71/83) | 94.7 |
| EquiIGD | 89.9 (71/79) | 94.7 | 77.0 (67/87) | 89.3 | 88.1 (74/84) | 98.7 |

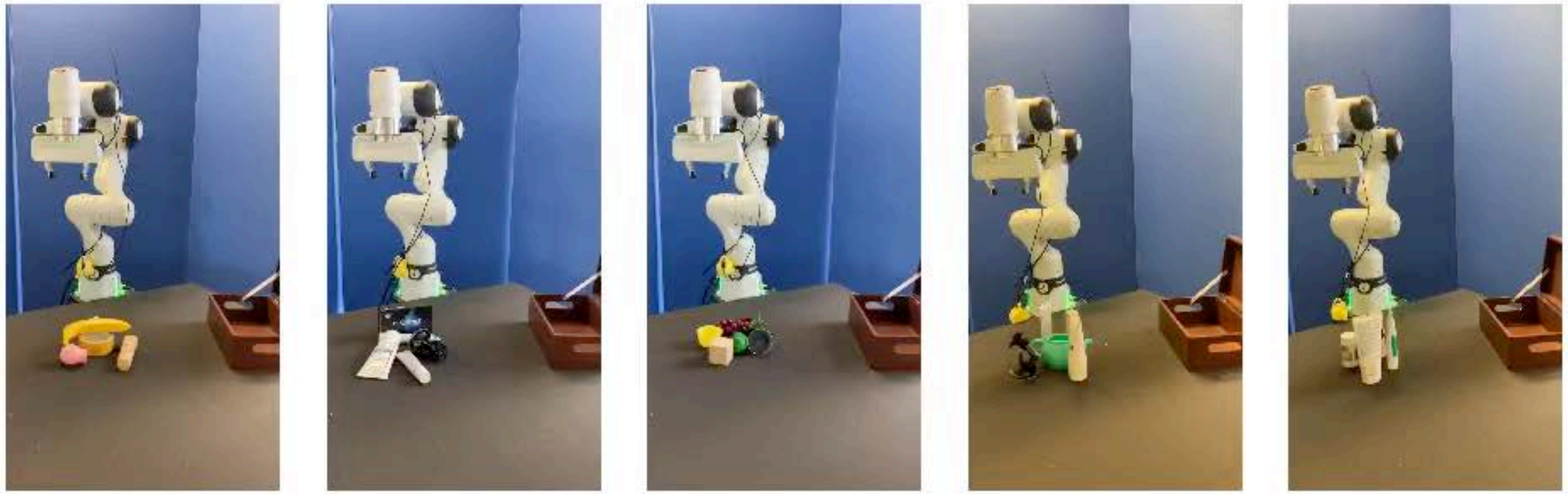
EquiGIGA



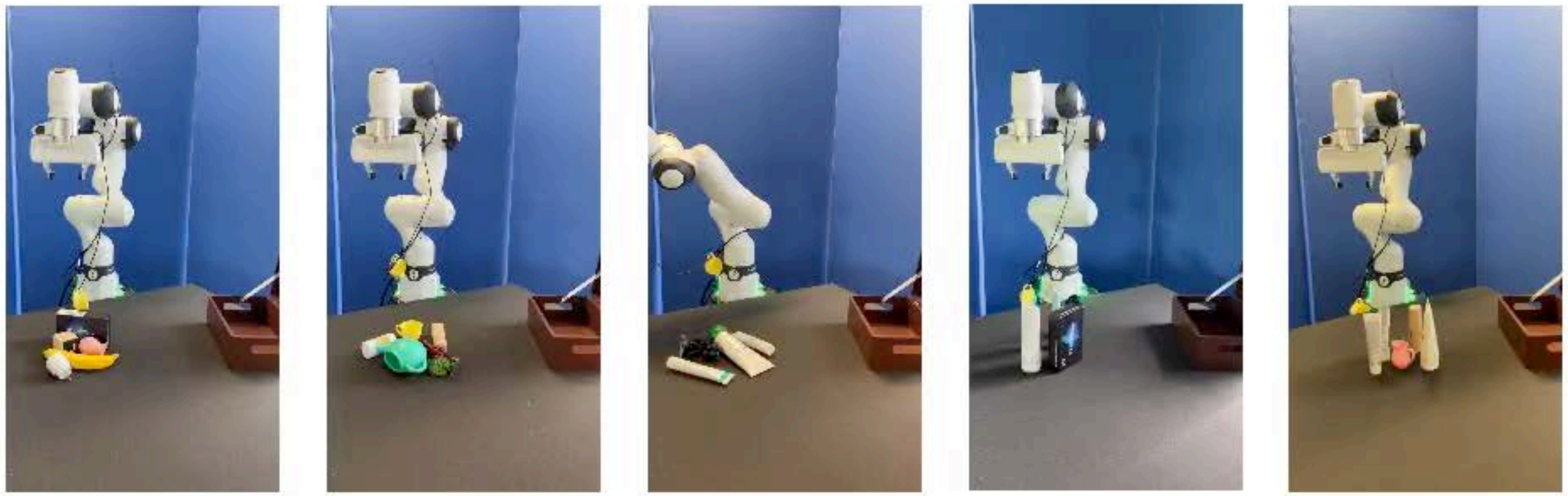
EquiIGD



EquiGIGA

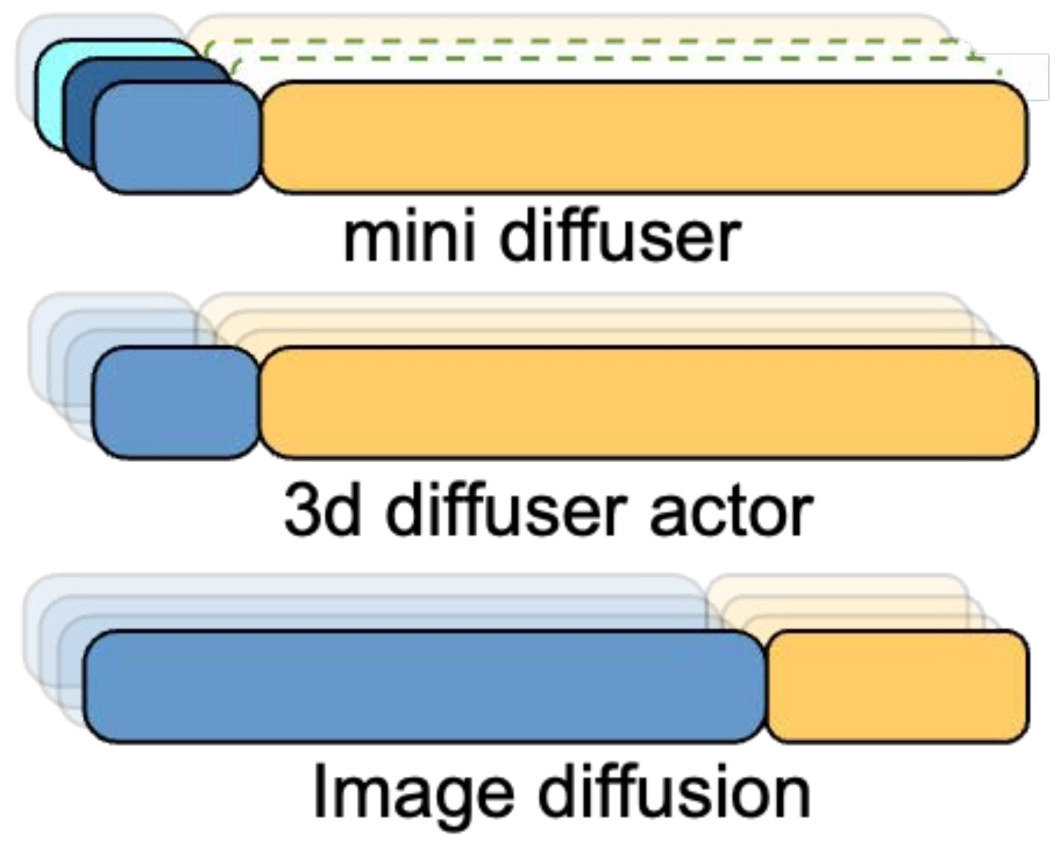


EquiIGD



**Mini Diffuser: Fast Multi-task Diffusion Policy Training Using Two-level Mini-batches**

Yutong Hu, Pinhao Song, Kehan Wen, and Renaud Detry

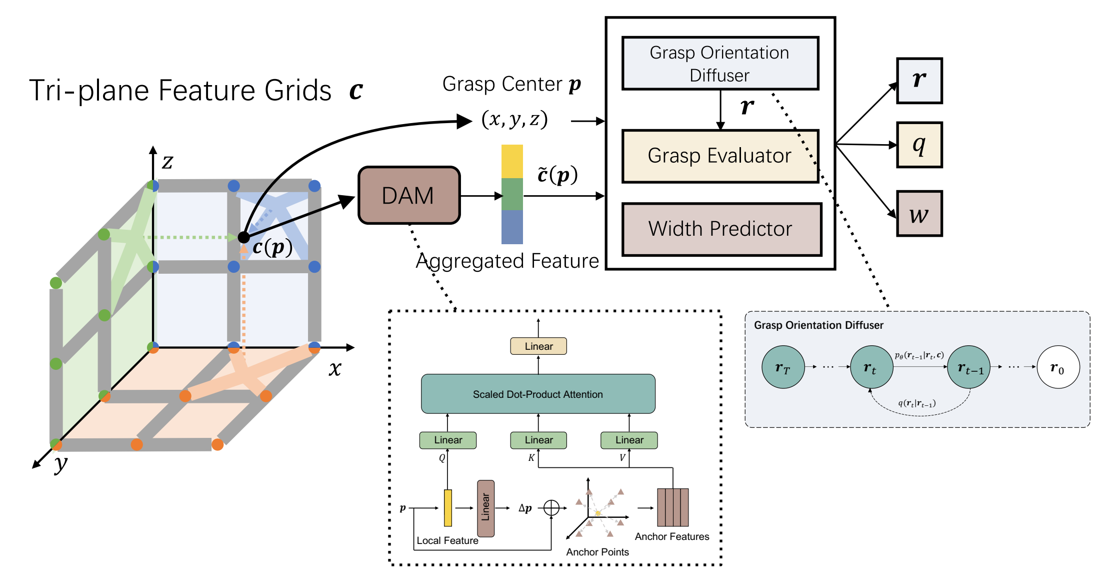


Take-home's:

- Mini-diffuser cuts compute and memory by an order of magnitude. Use it to accelerate model prototyping!
- Equivariant modeling requires delicate trade-offs.
  - The structure of tri-plane feature projection lends itself to $C_4$ $Z$-rotation equivariance.

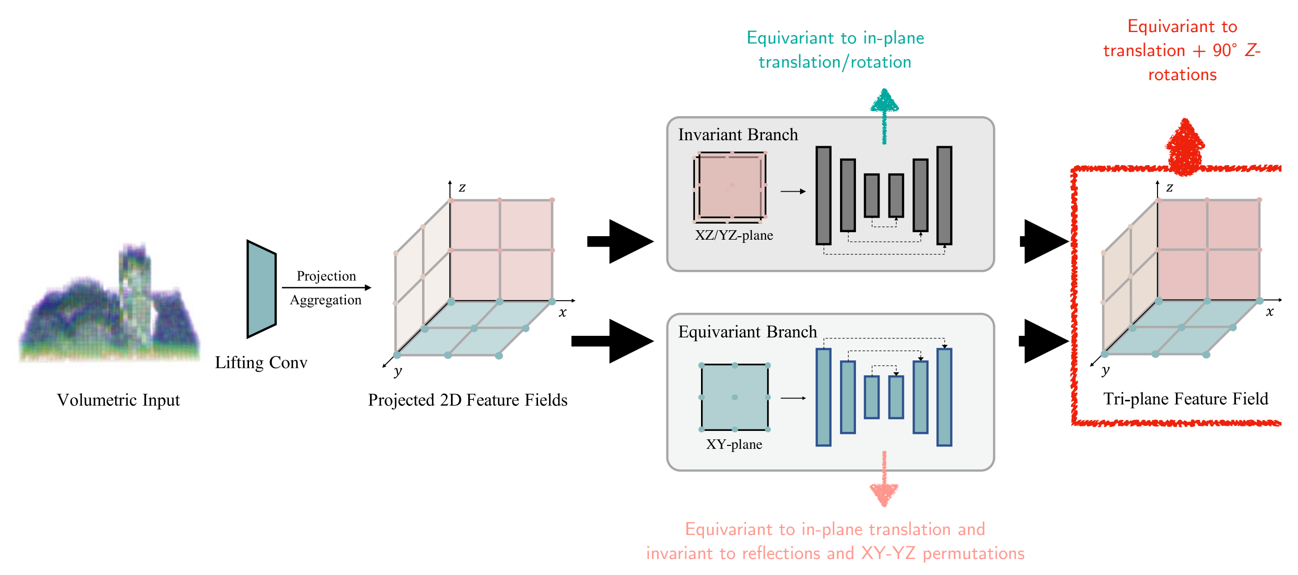
**Implicit grasp diffusion: Bridging the gap between dense prediction and sampling-based grasping**

P. Song, P. Li, and R. Detry, CoRL 2024



**Equivariant volumetric grasping**

P. Song, Y. Hu, P. Li, and R. Detry


Equivariant to in-plane translation/rotation
Equivariant to translation + 90° $Z$-rotations
Invariant Branch
XZ/YZ-plane
Equivariant Branch
XY-plane
Volumetric Input
Lifting Conv
Projection
Aggregation
Projected 2D Feature Fields
Tri-plane Feature Field
Equivariant to in-plane translation and invariant to reflections and XY-YZ permutations