Penn Engineering
University of Pennsylvania

FA26

Special Seminar on
Multi-Modal AI

Jiayuan Mao

Department of CIS, GRASP

https://cis7000-fall26.thelabone.org/



Penn Engineering
University of Pennsylvania

FA26

Lecture 19: Action Representations

Special Seminar on Multi-Modal AI

Jiayuan Mao
Department of CIS, GRASP
https://cis7000-fall26.thelabone.org/

Mid-term Project Check-In

- Pick a 30-min slot here: meet.jiayuanm.com
- Or email me

Lecture Outline

Topic	Concepts	Examples
Robotics	Kinematics, actuation, contact	Arms, legs, hands, soft robots
Actions	Joint, EE, point targets	IK, PDO-IK, URDF
Learning	Chunks and distributions	ACT, diffusion, HT

Robot Foundations:
The Configuration Space

Robot Examples



Mobile manipulator



Human hand



LEAP robot hand (**1.5x** larger in size)

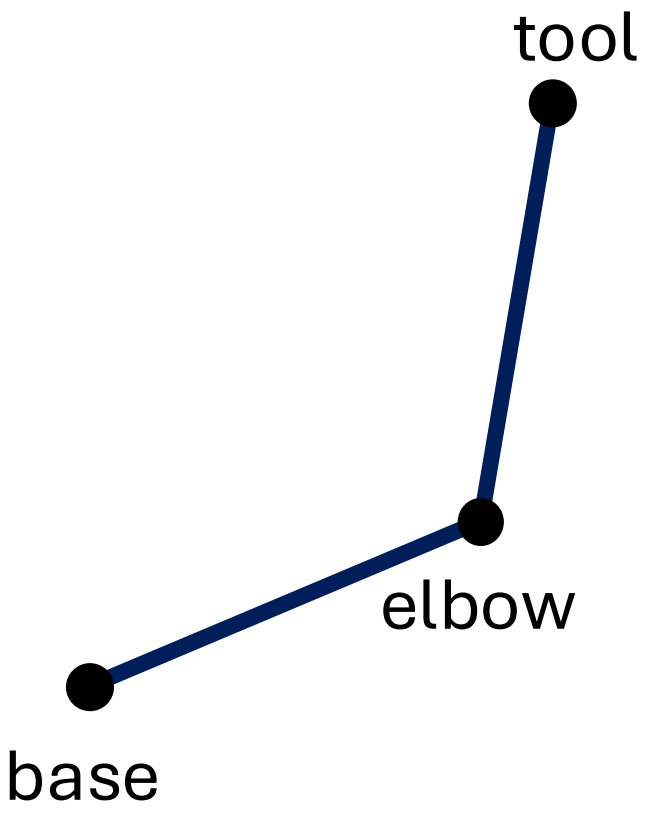
Dexterous hand



Soft suction arm

Reference: Fu et al., Mobile ALOHA • Wang et al., DexCap • Harvard/Wyss, The Tentacle Bot

Configuration Spaces: The Abstract View

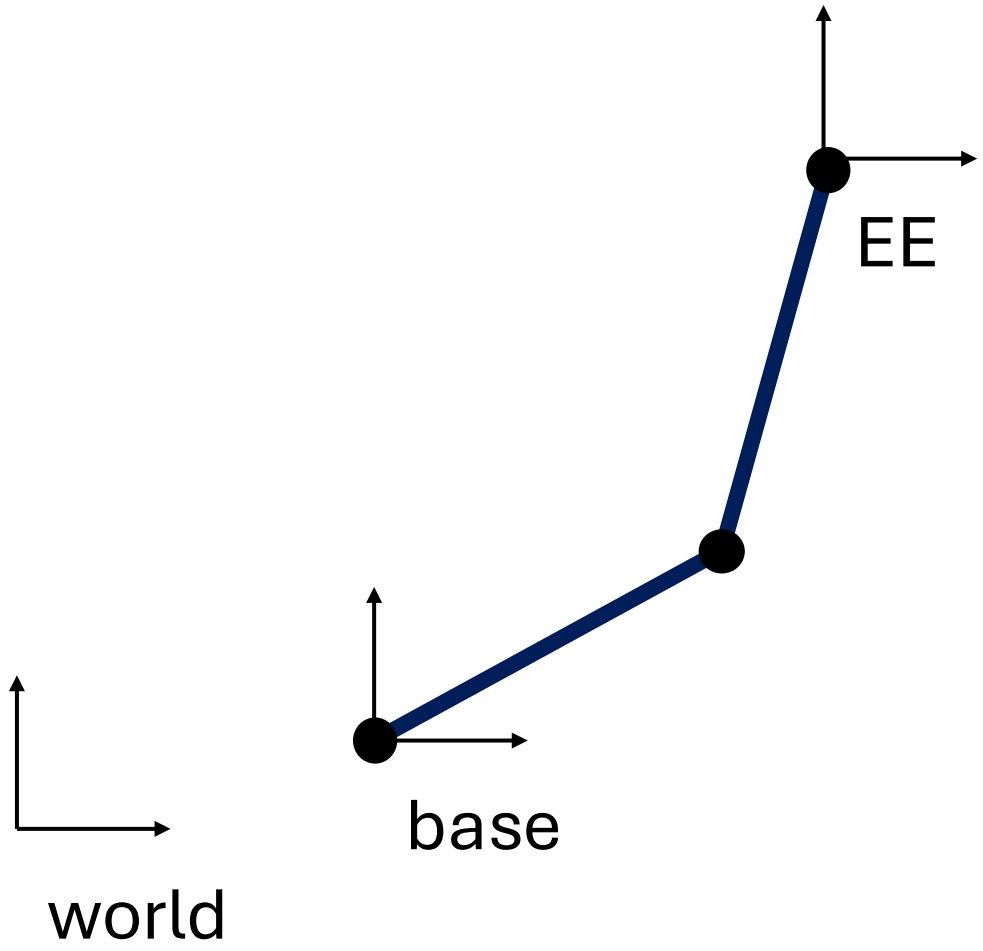


- Link geometry and mass
- Joint motion constraints
- Configuration dimension is usually #actuators

A configuration: $q = (q_1, q_2) \in \mathcal{Q}$

Reference: Hauser, Robotic Systems: Inverse Kinematics

Coordinate Frames

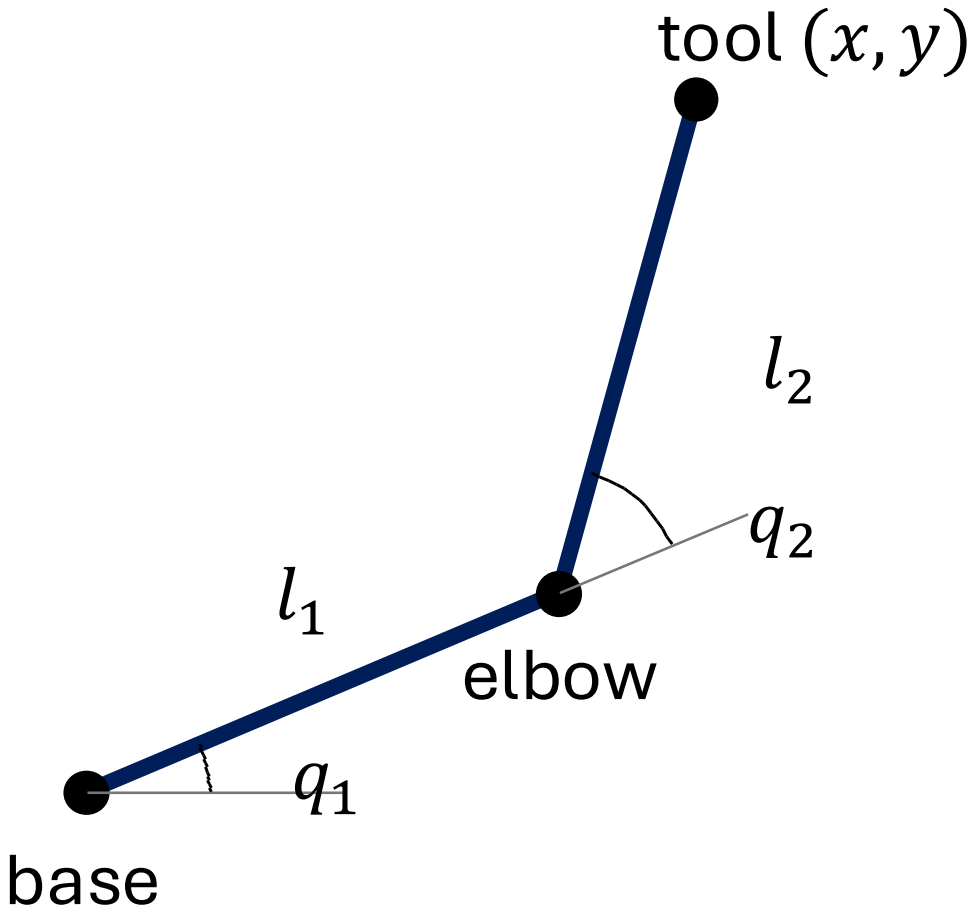


$$^{W}T_{E} = {}^{W}T_{B}\,{}^{B}T_{E}(q)$$

End-effector motion
= Base motion + Arm motion

Reference: Hauser, Robotic Systems: Inverse Kinematics • ROS 2 Documentation: URDF Tutorials

Forward Kinematics: Two-link Example



$$x = l_1 \cos q_1 + l_2 \cos(q_1 + q_2)$$

$$y = l_1 \sin q_1 + l_2 \sin(q_1 + q_2)$$

Reference: Hauser, Robotic Systems: Inverse Kinematics

Two-Link Inverse Kinematics

$$c_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2 l_1 l_2}$$

$$q_2 = \text{atan2}\left(\pm\sqrt{1 - c_2^2}, c_2 \right)$$

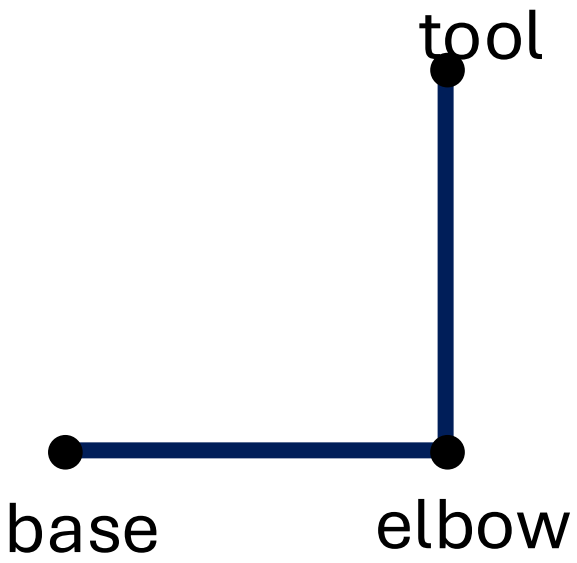
$$q_1 = \text{atan2}(y, x) - \text{atan2}(l_2 \sin q_2, l_1 + l_2 \cos q_2)$$

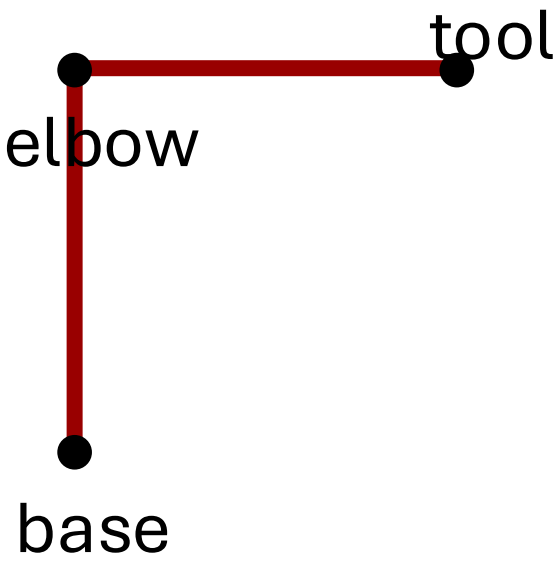
The ± sign gives two IK solutions

Reference: Hauser, Robotic Systems: Inverse Kinematics

Multiple Inverse Kinematics Solutions

Target: (1, 1)



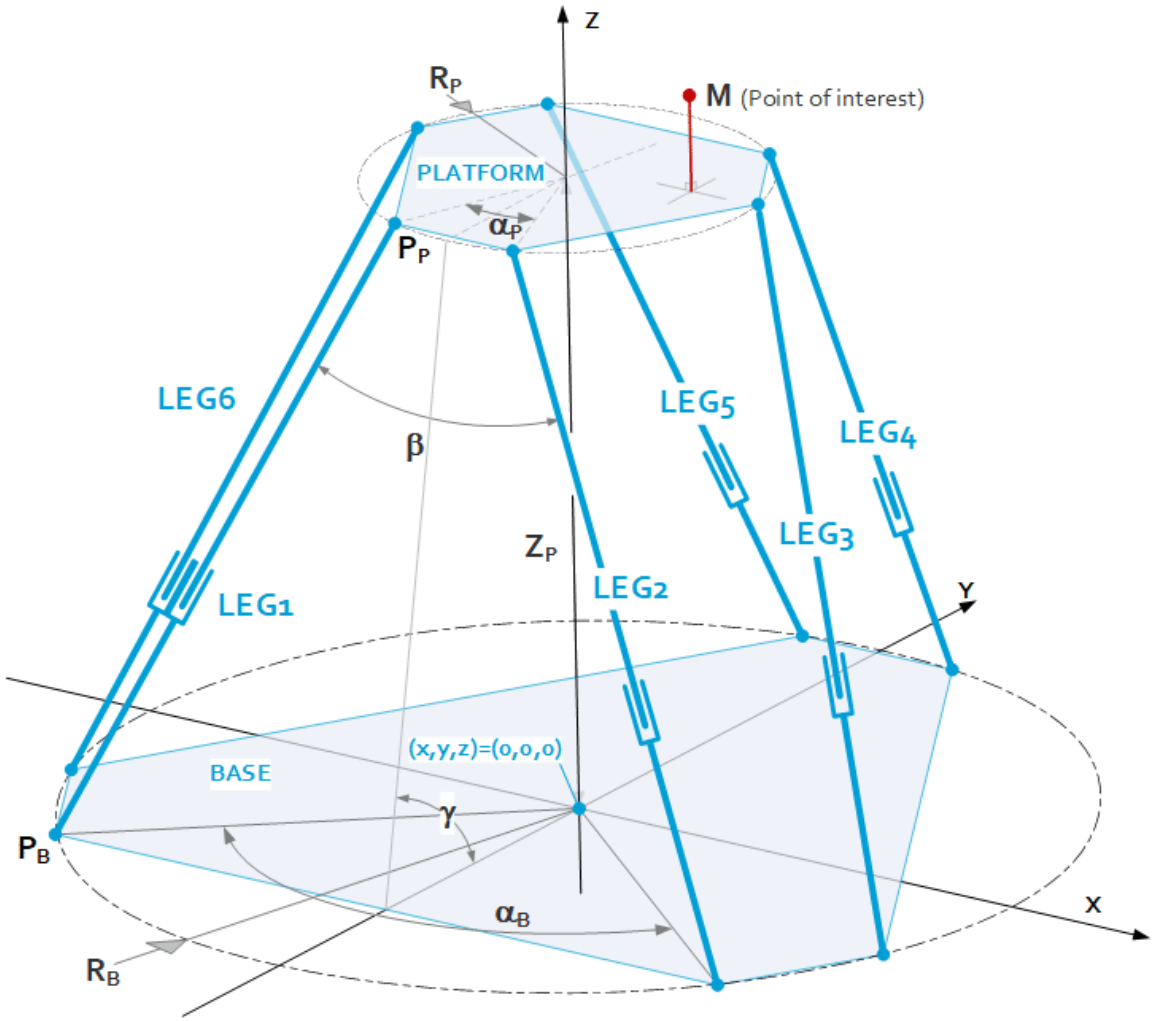


$(q_1, q_2) = (0°, 90°)$

$(q_1, q_2) = (90°, -90°)$

Reference: Hauser, Robotic Systems: Inverse Kinematics

Stewart Platform: Prismatic Joints and IK



Inverse kinematics (IK)

Given the platform pose
Compute each leg length

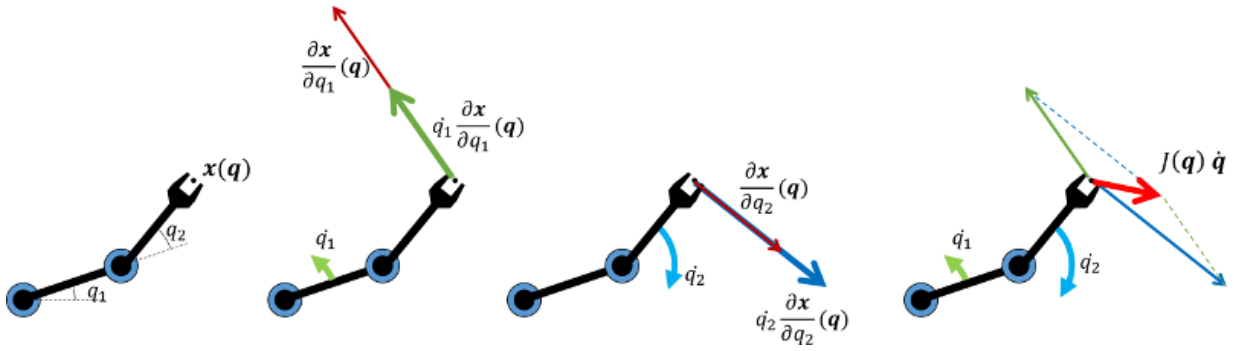
Forward kinematics (FK)

Given six leg lengths
Solve coupled pose equations
Solution may not exist

Reference: JPE, HexaPod: Kinematics • Lynch & Park, Modern Robotics, Ch. 7: Closed Chains

Jacobian

$$\dot{x} = J(q)\dot{q}, \qquad J_i = \frac{\partial x}{\partial q_i}$$



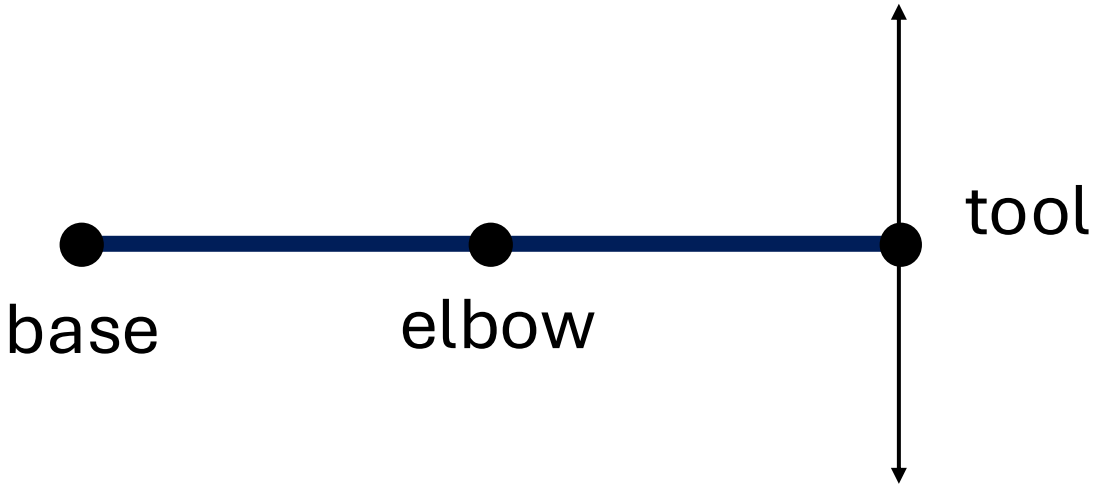
$$J = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix}$$

$$s_{12} = \sin(q_1 + q_2), \quad c_1 = \cos q_1$$

The Jacobian describes local motion: how joint moves give EE moves

Reference: Lynch & Park, Modern Robotics, Sec. 5.3: Singularities • Hauser, Robotic Systems: Inverse Kinematics

Kinematic Singularities



$$\det J = l_1 l_2 \sin q_2$$

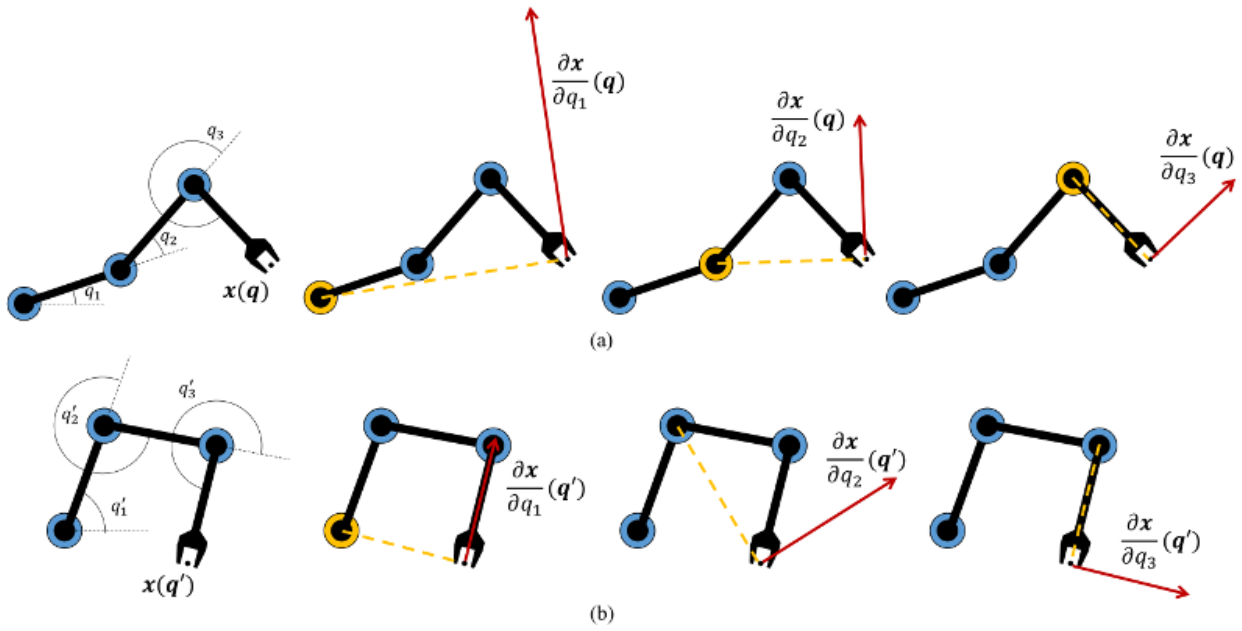
Straight or folded links lose rank

$$\dot{q} = J^{\top}(JJ^{\top} + \lambda^2 I)^{-1}\dot{x}$$

Damping limits joint velocities near singularities

Reference: Lynch & Park, Modern Robotics, Sec. 5.3: Singularities • Hauser, Robotic Systems: Inverse Kinematics

Redundant Manipulators



$$\dot{q} = J^{\dagger}\dot{x} + \left(I - J^{\dagger}J\right)z$$

Different elbow motions can preserve the tool motion

The z can be used for many different purposes:

- Stay close to home
- Avoid singularity

Reference: Hauser, Robotic Systems: Inverse Kinematics

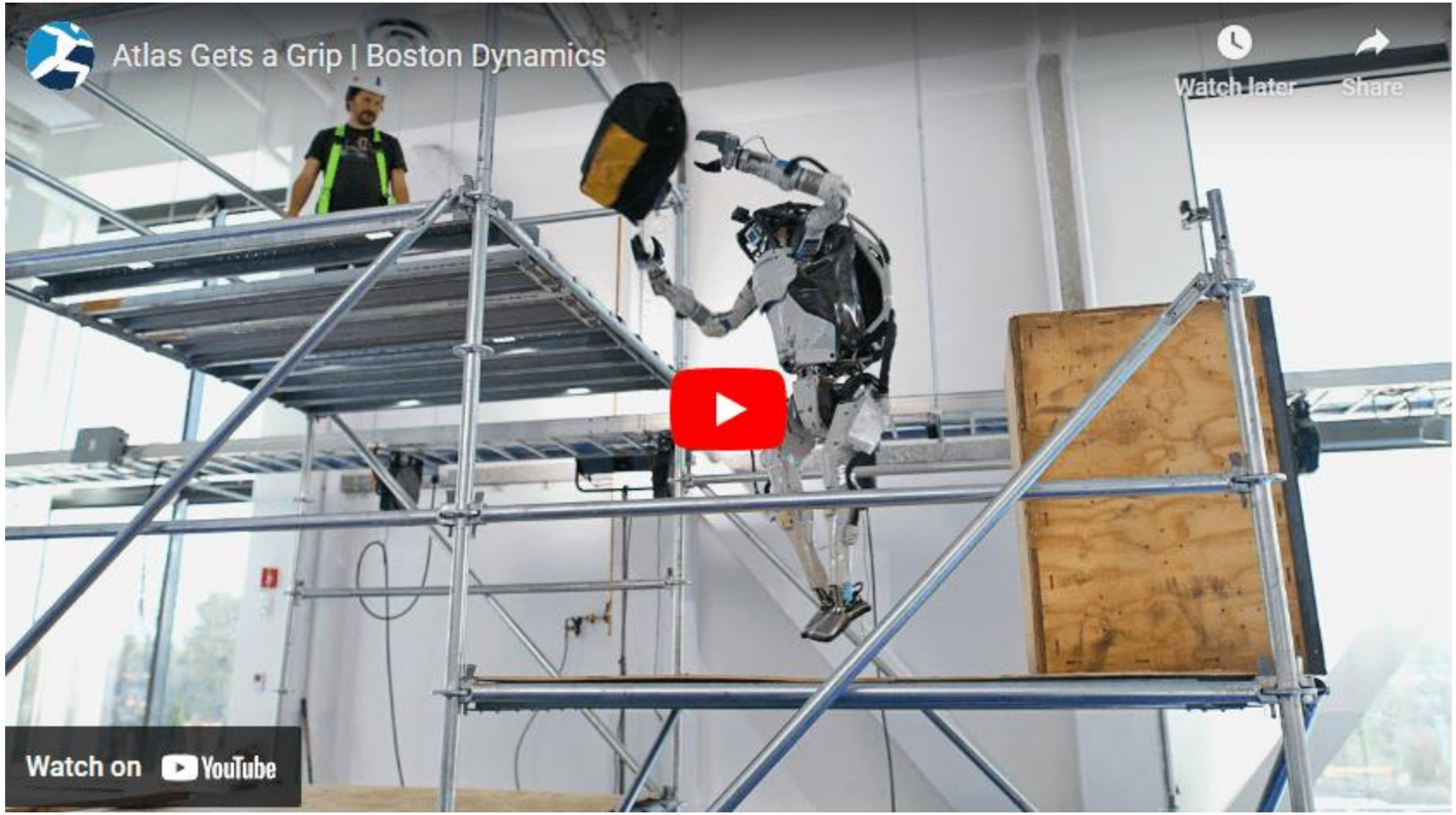
Mobile Manipulators



Wheel constraints:
Holonomic wheels

Reference: Fu et al., Mobile ALOHA

Humanoid Robots



Key challenge:
Balance, friction, etc.
A lot of motion constraints

Cannot be easily simplified as point mass

Reference: Boston Dynamics (2023), Atlas Gets a Grip

Quadruped Robots



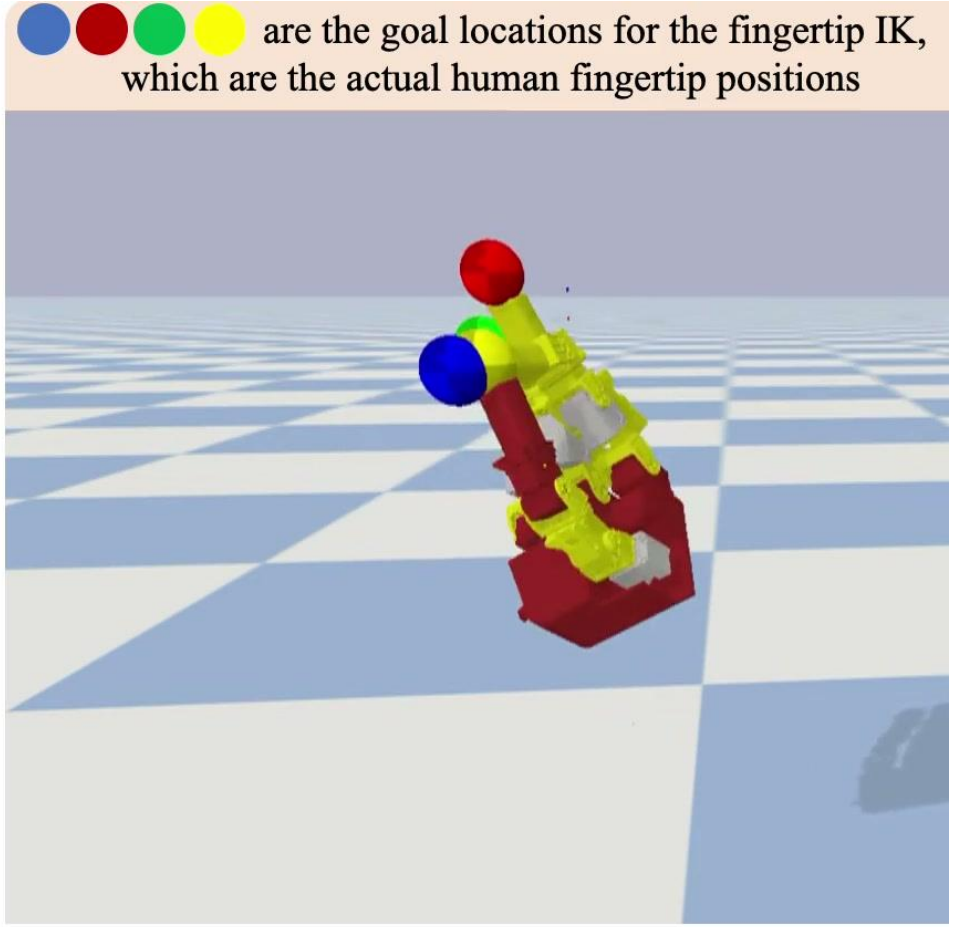
Can sometimes be simplified as point mass

Reference: ETH Zürich Robotic Systems Lab, ANYmal

Dexterous Hands



Human hand



LEAP robot hand (**1.5x** larger in size)

Key challenge:
High DOF: a lot of joint coupling

Contact forces and grasp stability

Reference: Wang et al., DexCap

Quadrotors



Key challenge: Underactuated

Thrust controls translation

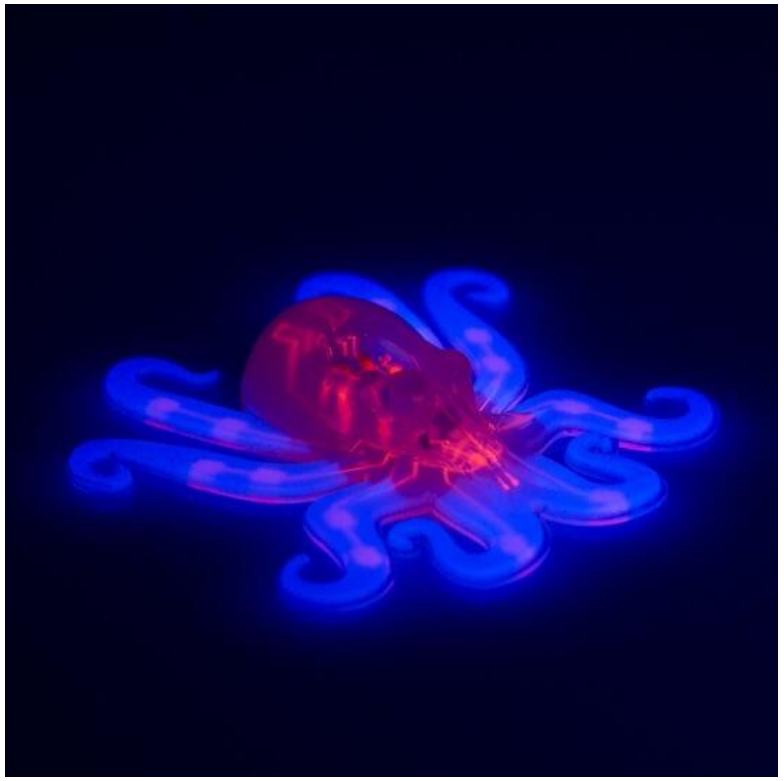
Attitude controls thrust direction

Reference: UZH Robotics and Perception Group, Drone Racing / Agilicious

Soft Robots



Entanglement gripper



Octobot

Shape depends on commands, material properties, and load

Reference: Harvard SEAS (2022), Jellyfish-Inspired Robotic Gripper • Harvard SEAS (2016), Octobot

Continuum Robot Representations



$$r(s) \in \mathbb{R}^3, \qquad s \in [0, L]$$

Sample points along the centerline

Approximate shape with piecewise curvature

Reference: Harvard/Wyss, The Tentacle Bot

Octopus-Inspired Robots



Reference: Harvard/Wyss, The Tentacle Bot

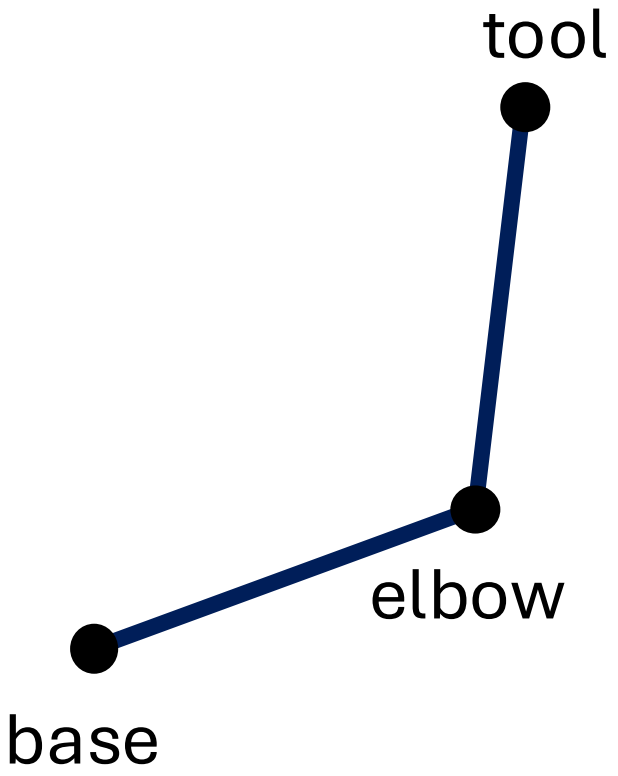
Summary: Robotics Action Foundations

Constraint	Effect	Example
Geometry	Reachable configurations	Link lengths, joint limits
Actuation	Available motion	Drone thrust, wheeled base
Contact and dynamics	Feasible motion and force	Balance, friction, deformation

Reference: Hauser, Robotic Systems: Inverse Kinematics • ROS 2 Documentation: URDF Tutorials

Robot Foundations:
The Action Representations

Joint-Space Actions



$$a_t = q_t^* \quad \text{or} \quad a_t = \Delta q_t$$

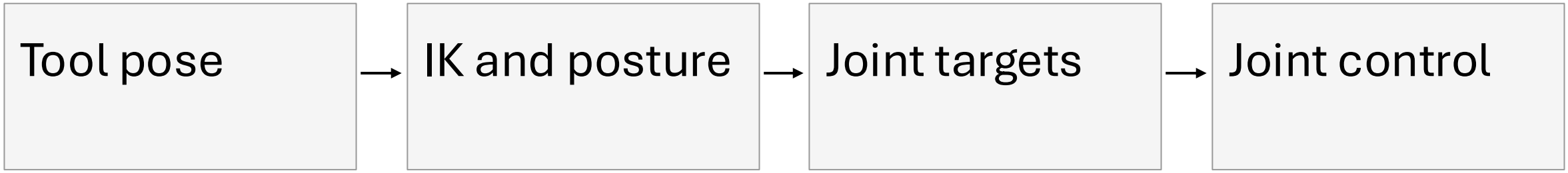
- Joint order and units
- Joint limits and actuator conventions

Position, increment, velocity, or torque

Reference: Zhao et al. (2023), Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT)

End-Effector Actions

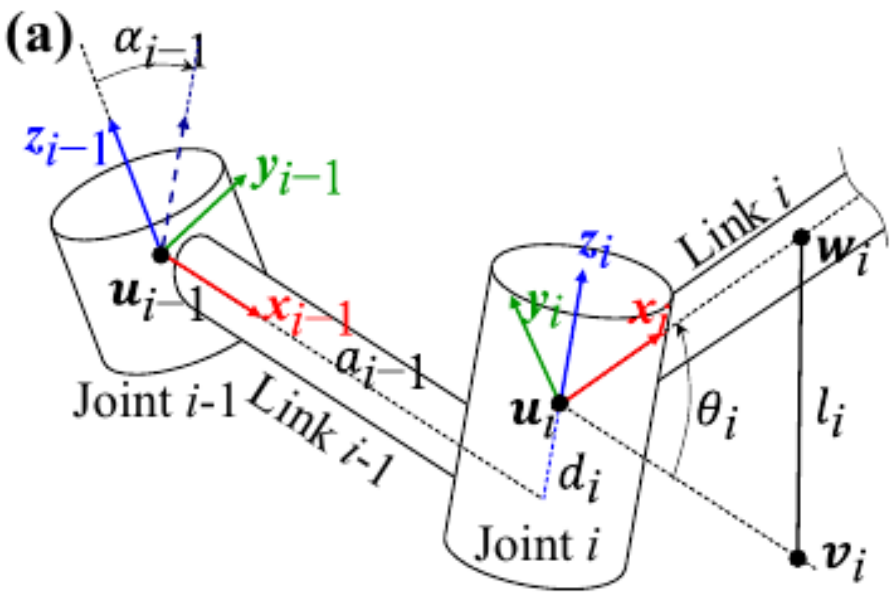
$$a_t = ({}^{W}T_E^*), \qquad T_E^* \in SE(3)$$

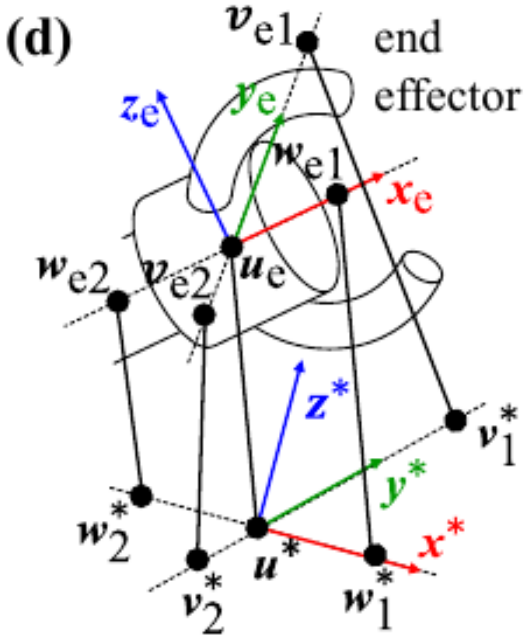


Redundancy and singularities still matter

Reference: Hauser, Robotic Systems: Inverse Kinematics

Point-Based Representation: PDO-IK





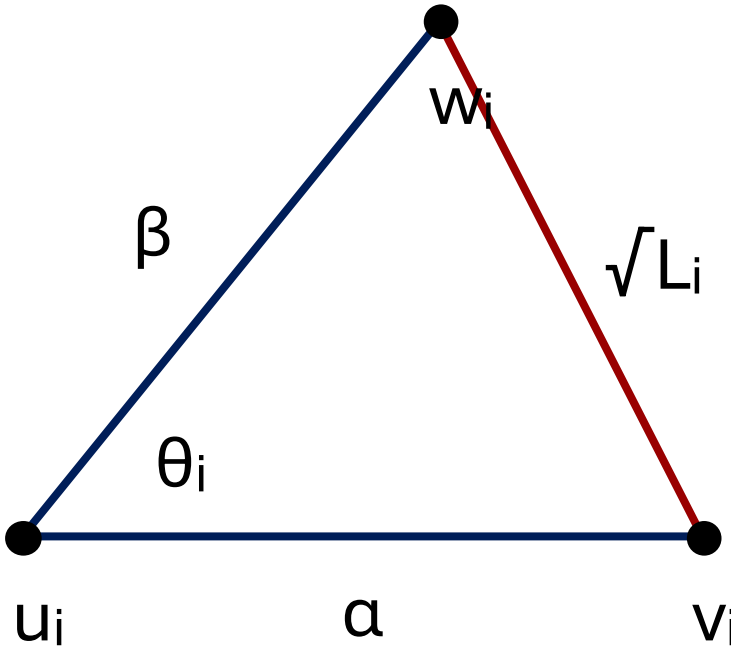
Robot points **Tool targets**

- Labeled body points
- Distances encode geometry
- IK recovers joint targets

Target points are converted to joint commands

Reference: Yu Chen et al. (WAFR 2024), Propagative Distance Optimization for Constrained Inverse Kinematics; Fig. 1

PDO-IK: Distance Representation



$$L_i = \| v_i - w_i \|^2 = \alpha^2 + \beta^2 - 2\alpha\beta\cos\theta_i$$

Normalized lengths: $\alpha = \beta = 1/\sqrt{2}$

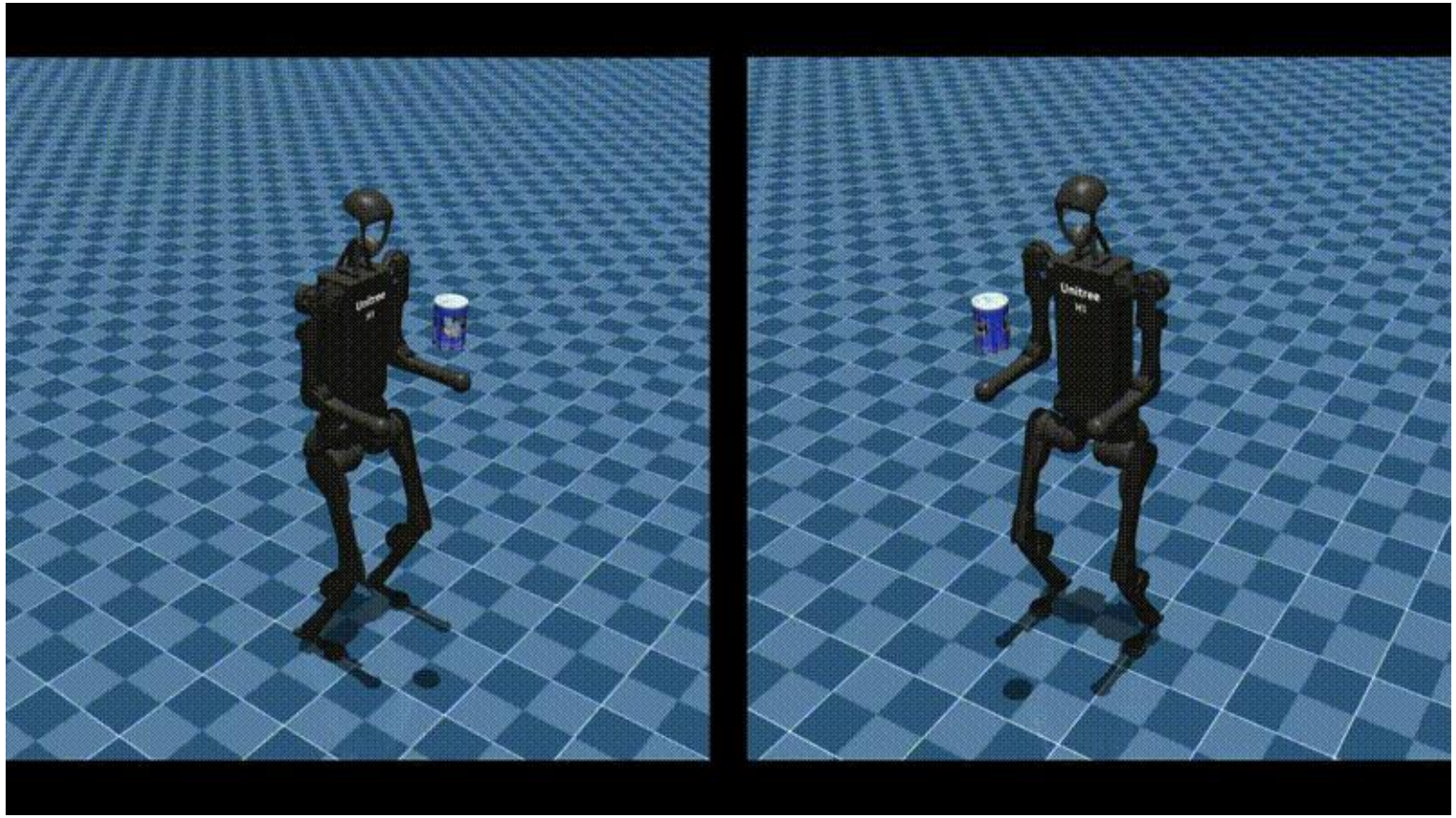
$$\cos\theta_i = 1 - L_i, \quad \sin\theta_i = \sqrt{2L_i - L_i^2}$$

$$0 \le \theta_i \le \pi, \qquad 0 \le L_i \le 2$$

Decompose larger angle ranges into bounded subangles

Reference: Yu Chen et al. (WAFR 2024), Propagative Distance Optimization for Constrained Inverse Kinematics; Sec. 3.1, Eqs. 1–2

PDO-IK: Experiments and Limitations



H1 simulation: fixed feet and a CoM constraint

Reference: Yu Chen et al. (WAFR 2024), Propagative Distance Optimization for Constrained Inverse Kinematics

Joint, End-Effector, and Point Representations

Representation	Output	Execution
Joint	Joint targets or increments	“Joint tracking”
End effector	Tool pose	IK / Jacobian
Point based	Labeled target points	IK

Reference: Hauser, Robotic Systems: Inverse Kinematics • Yu Chen et al. (2024), PDO-IK

Discussion: Why Use Point Targets?

Task instructions and perception often use spatial coordinates

Perception output

3D object points
(camera frame)

Camera to world

$$p_{\text{obj}}^{W} = R_{wc} p_{\text{obj}}^{C} + t_{wc}$$

“Move 3 m right”
(world +x direction)

$$p_{\text{target}}^{W} = p_{\text{obj}}^{W} + \begin{bmatrix} 3 \\ 0 \\ 0 \end{bmatrix} \text{m}$$

Point inputs and outputs can simplify what the network must learn

Reference: Lynch & Park, Modern Robotics, Sec. 3.3.1 • Yu Chen et al. (2024), PDO-IK

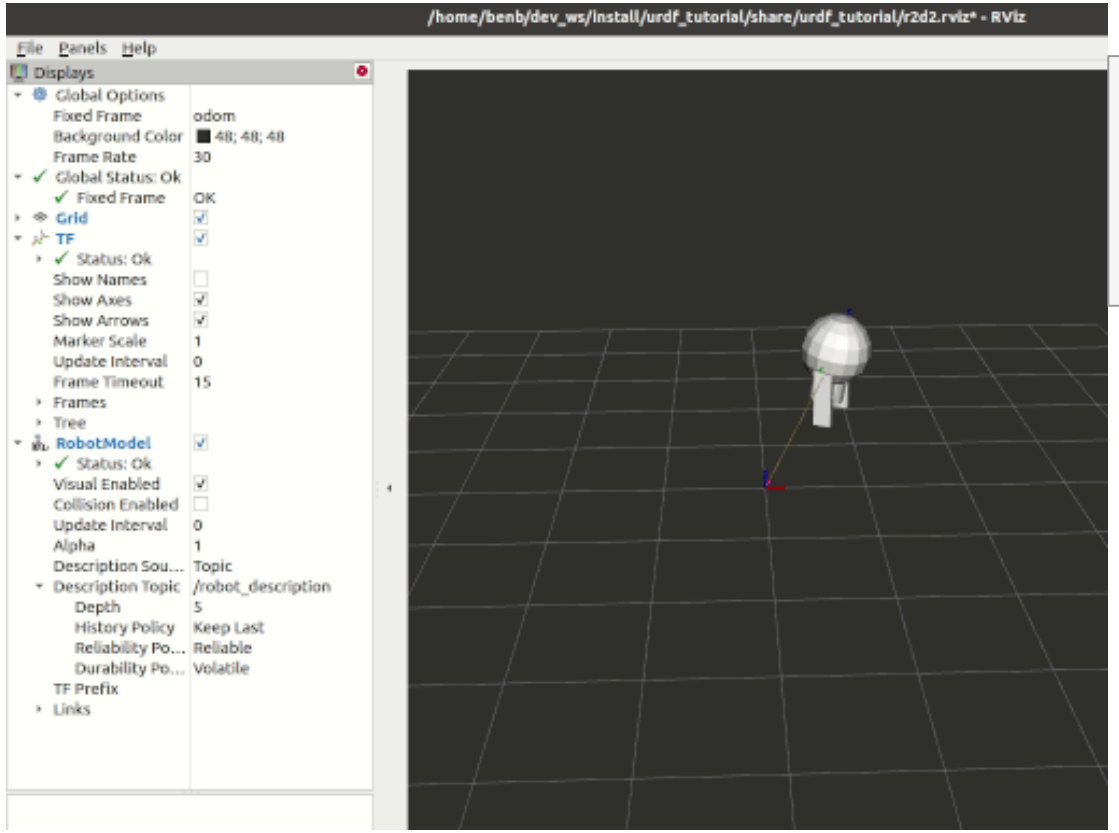
Discrete and Continuous Actions

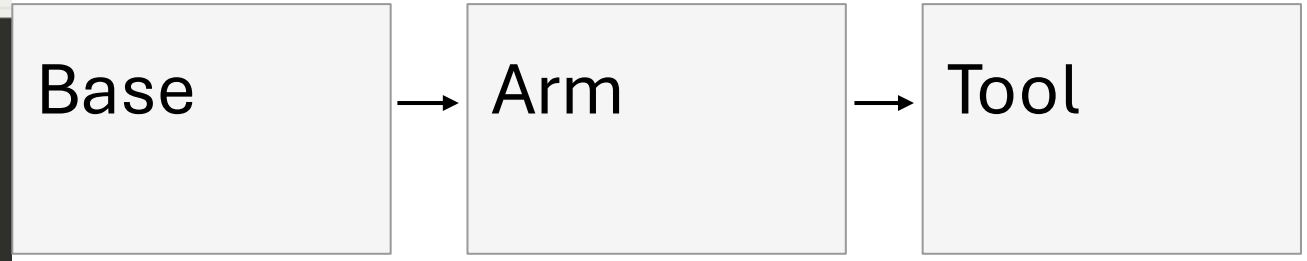
Representation	Continuous	Discrete / hybrid
Joint	Angles or velocities	Joint bins or primitives
End effector	Pose or twist	Pose bins and gripper state
Point based	3D target coordinates	Point bins or tokens

Reference: Zhao et al. (2023), ACT • Chi et al. (2023/2024), Diffusion Policy

Robot Foundations:
The Robot Structure Representation

URDF: Robot Description





Joints connect the link frames

Link properties

Allowed joint motion

A tree of coordinate frames

Reference: ROS 2 Documentation: URDF Tutorials

URDF: Links, Joints, and Frames

```
<joint name="elbow" type="revolute">
 <parent link="upper_arm"/>
 <child link="forearm"/>
 <origin xyz=".30 0 0" rpy="0 0 0"/>
 <axis xyz="0 0 1"/>
 <limit lower="-2.6" upper="2.6"
      effort="20" velocity="2"/>
</joint>
```

- Parent and child links
- Joint frame
- Motion axis and limits

Reference: ROS 2 Documentation: URDF Tutorials

Actions Should be Conditioned on A Lot Of Things…

Information	Example	Source
Morphology	Links, axes, limits, inertia	Robot description
State	Joint motion and contacts	Sensors
Control interface	Position, velocity, effort	Driver and controller

$$\pi_\theta(a_t \mid o_{\leq t}, \mathcal{M}, \mathcal{C})$$

Reference: ROS 2 Documentation: URDF Tutorials

URDF: Pros and Cons

- A kinematic tree gives the policy explicit body structure
- Inertias help, but actuators and controller usually have more complex dynamics properties
- Soft material behavior requires richer models

Reference: ROS 2 Documentation: URDF Tutorials

Machine Learning for Actions

Behavior Cloning

$$\mathcal{D} = \{(o_t, a_t^*)\}, \qquad \min_\theta \mathbb{E}_{\mathcal{D}}[\ell(\pi_\theta(o_t), a_t^*)]$$

Demonstrations → Policy training → Policy rollout

Reference: Zhao et al. (2023), ACT • Chi et al. (2023/2024), Diffusion Policy

Action Chunking

$$A_t = (a_t, \ldots, a_{t+H-1}), \qquad 1 \leq h \leq H$$



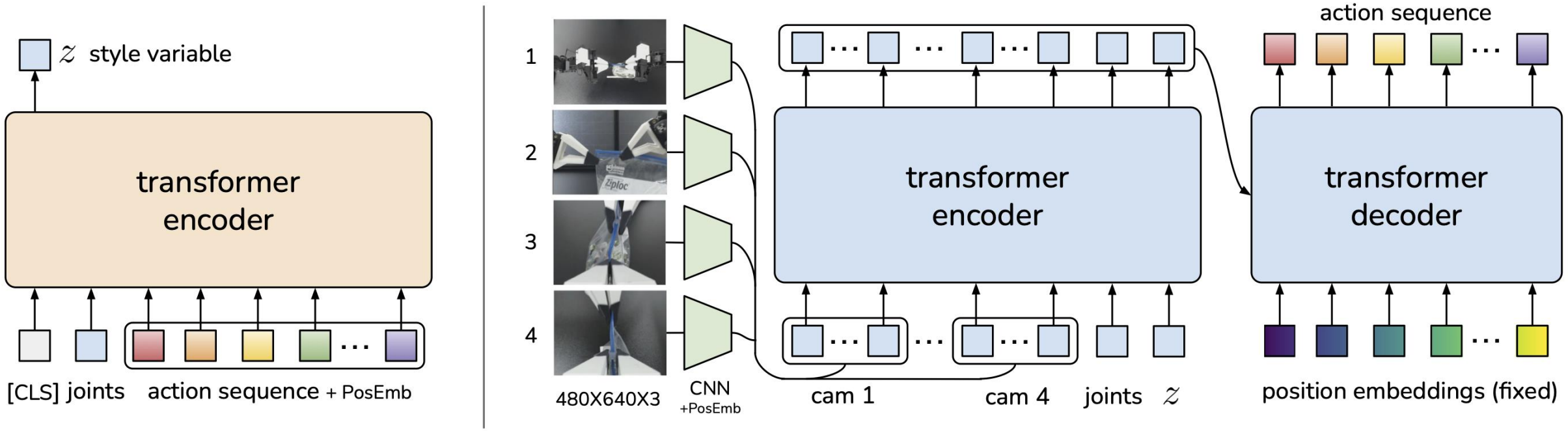
Execute first 3

Predict 8 actions

Prediction length and feedback delay are different choices

Reference: Zhao et al. (2023), ACT • Chi et al. (2023/2024), Diffusion Policy

Action Chunking Transformer (ACT)



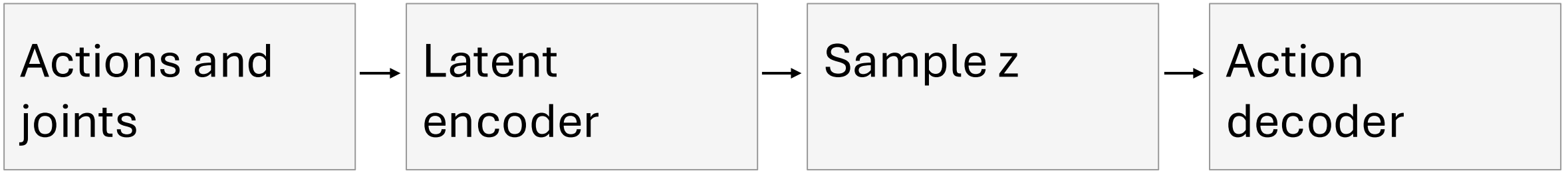
Predict an action sequence from images and joint state

Output: a chunk of joint position targets

Reference: Zhao et al. (2023), Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT)

ACT: Training with a CVAE

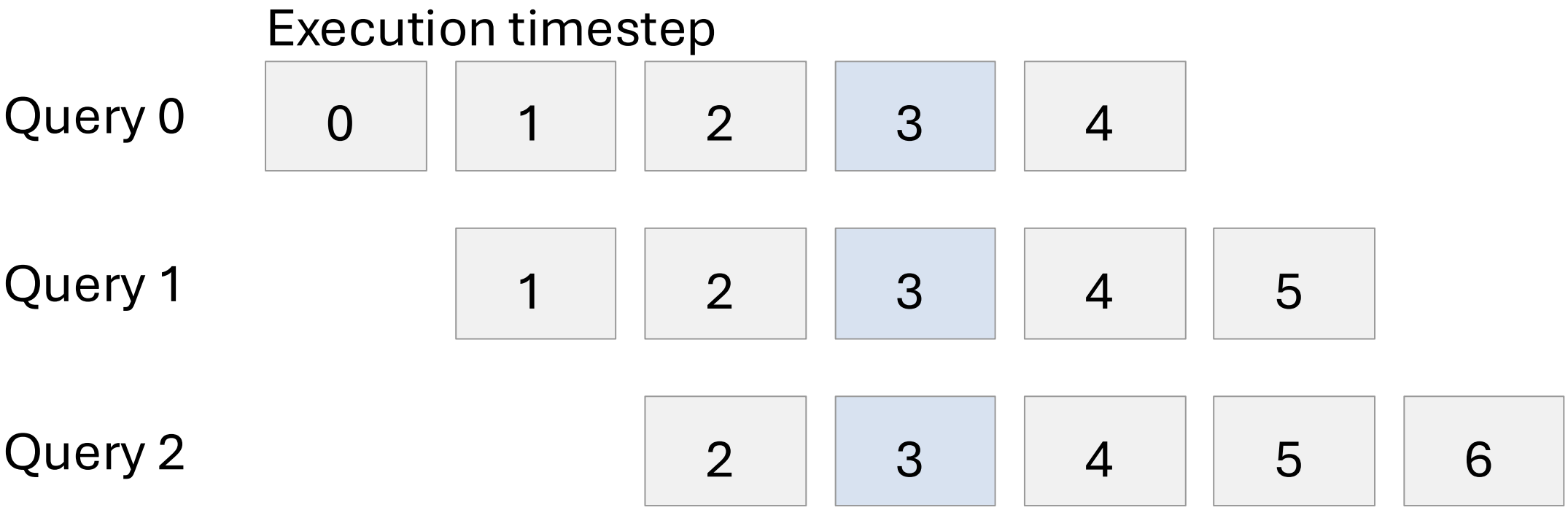
$$\mathcal{L} = \| A - \hat{A}_\theta(o, z) \|_1 + \beta D_{\mathrm{KL}}\left(q_\phi(z \mid A, q) \,\|\, \mathcal{N}(0, I)\right)$$



At inference, use images and joint state with z = 0

Reference: Zhao et al. (2023), Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT)

ACT: Temporal Ensembling



$$a_3 = \sum_i \bar{w}_i \, \hat{a}_{3|i}, \quad \sum_i \bar{w}_i = 1$$

Combine estimates at one time step

Reference: Zhao et al. (2023), Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT)

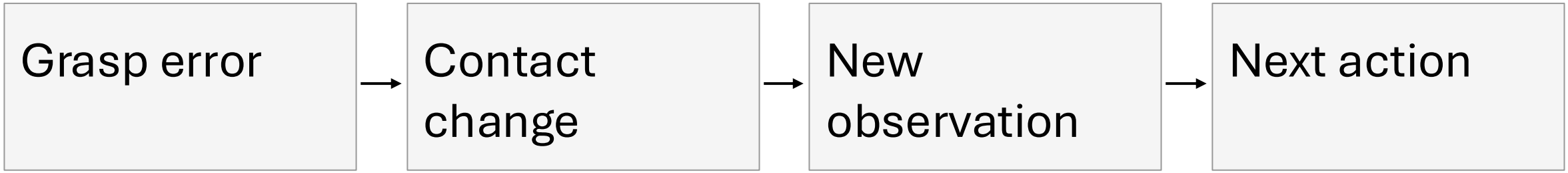
ACT: Experiments



Slot Battery: 96% success in 25 trials

Reference: Zhao et al. (2023), Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware (ACT)

Error Accumulation in Behavior Cloning



$$x_{t+1} = F\left(x_t, \pi(x_t)\right)$$

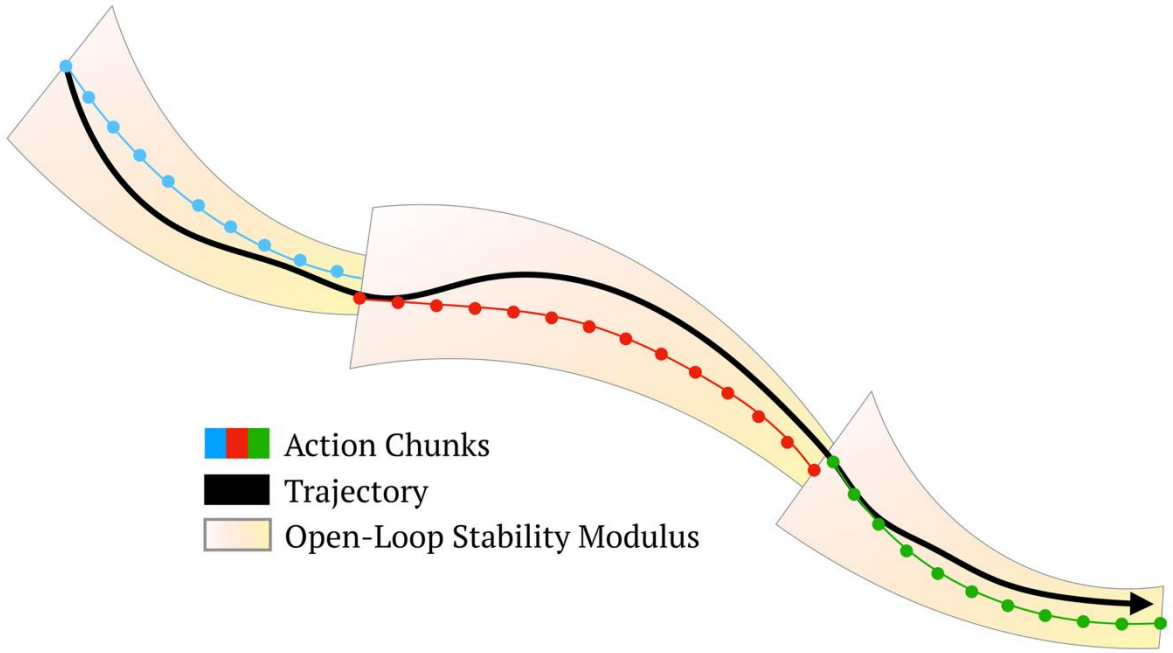
“Stability”: whether errors shrink or grow

Reference: Zhang et al. (2025), Action Chunking and Exploratory Data Collection, arXiv:2507.09061

Why Action Chunking Works?

Reference: Zhang et al. (2025), Action Chunking and Exploratory Data Collection, arXiv:2507.09061

Policy Sensitivity

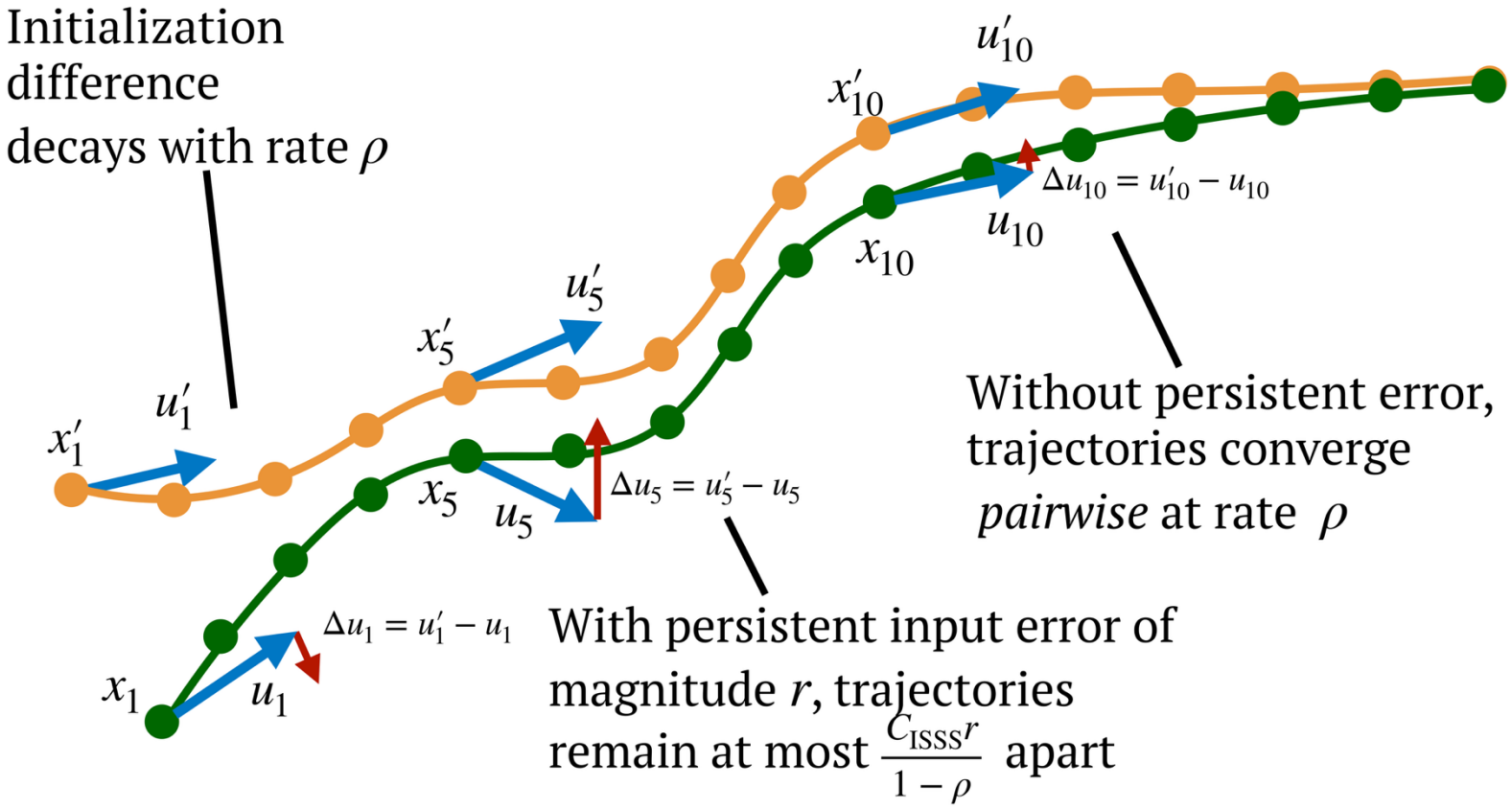


Chunking reduces repeated queries to a sensitive policy

The dynamics must satisfy stability assumptions

Reference: Zhang et al. (2025), Action Chunking and Exploratory Data Collection, arXiv:2507.09061

Action Chunking: Stability Analysis



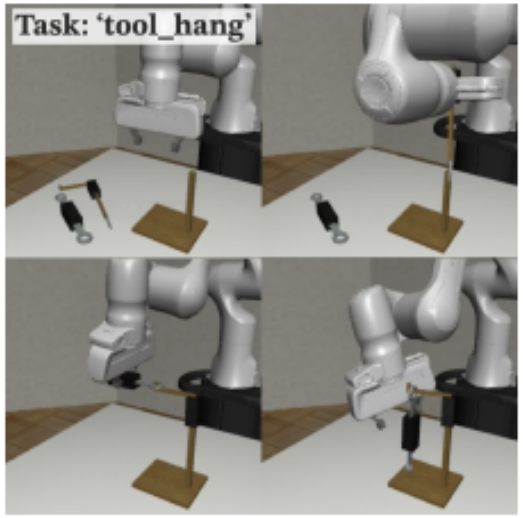
"Incrementally stable dynamics"

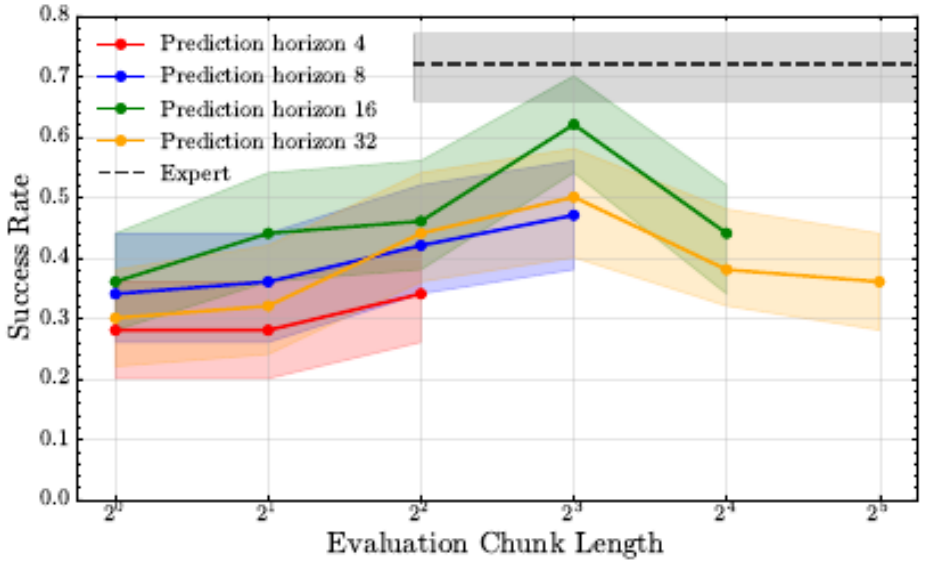
Each system have a rate at which action errors will decay

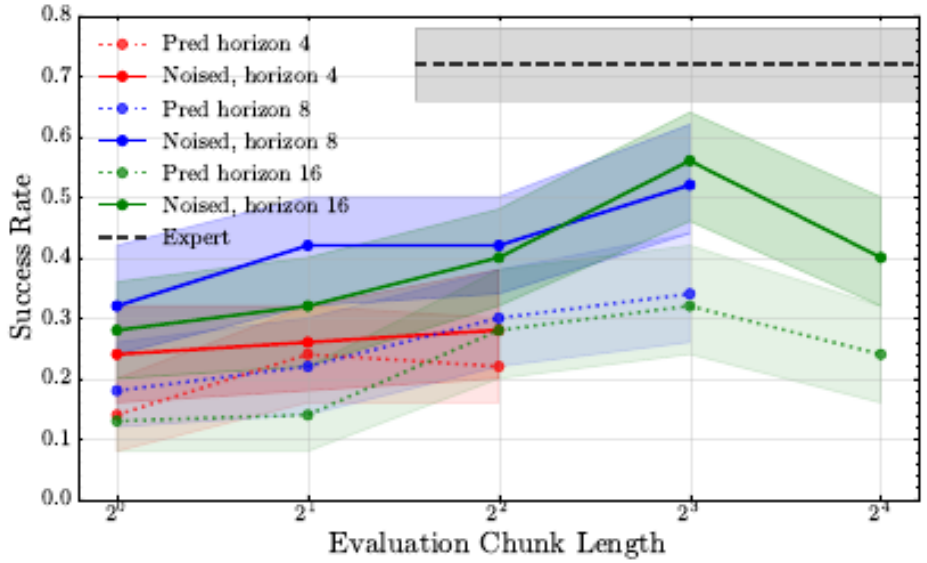
We need to balance error decay and error accumulation (by calling pi too much)

Reference: Zhang et al. (2025), Action Chunking and Exploratory Data Collection, arXiv:2507.09061; Proposition 3.1

Prediction and Execution Horizons







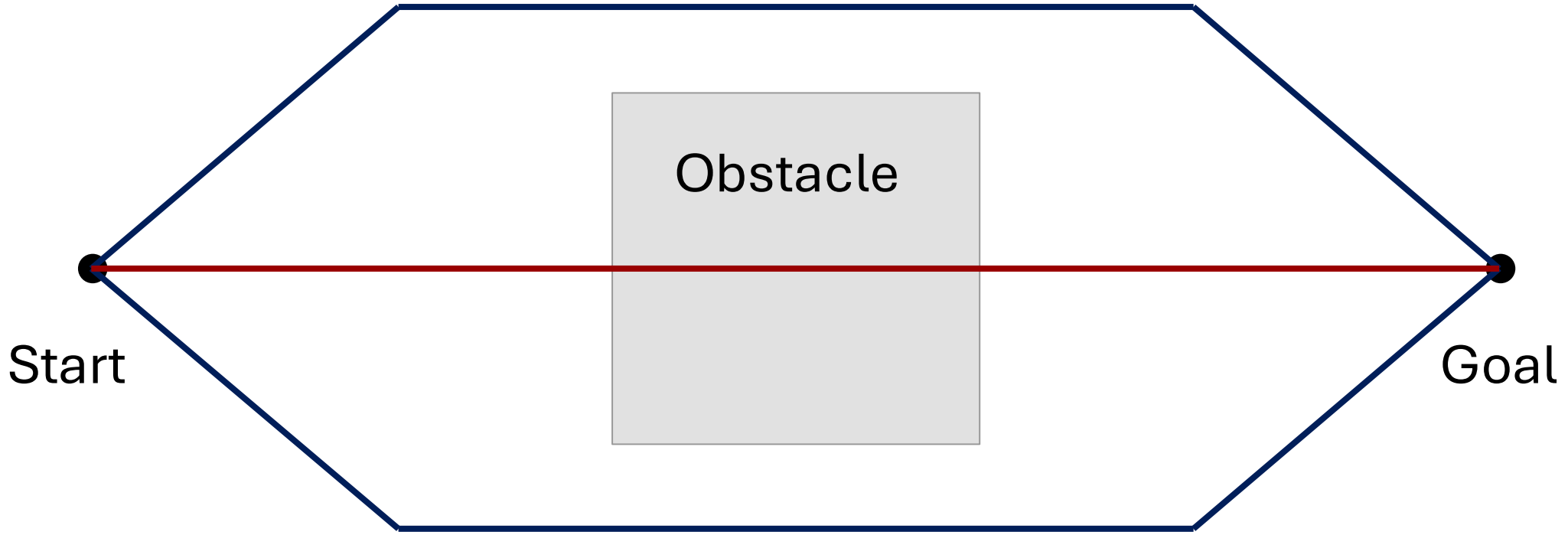
Key idea: Execution length!!! matters (not prediction horizons)

Choose a balance between stability and feedback delay

Reference: Zhang et al. (2025), Action Chunking and Exploratory Data Collection, arXiv:2507.09061; Fig. 5

Multimodal Action Distributions

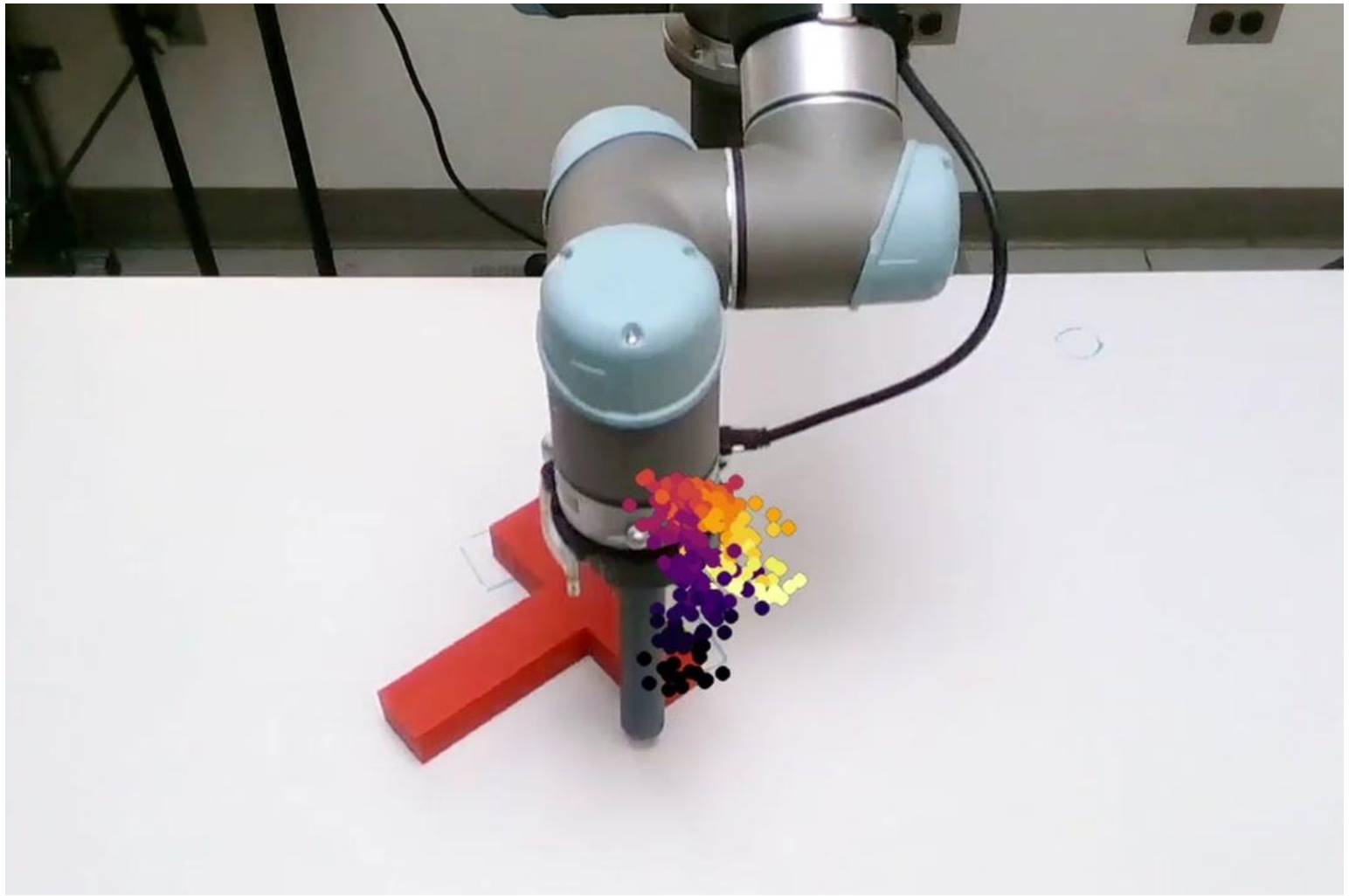
Two routes around the same obstacle



Mean path collides

Reference: Chi et al. (2023/2024), Diffusion Policy: Visuomotor Policy Learning via Action Diffusion

Diffusion Policy: Inference



Sample a noisy action chunk

Denoise using observations

Execute a prefix and replan

Reference: Chi et al. (2023/2024), Diffusion Policy: Visuomotor Policy Learning via Action Diffusion

Diffusion Policy: Training Objective

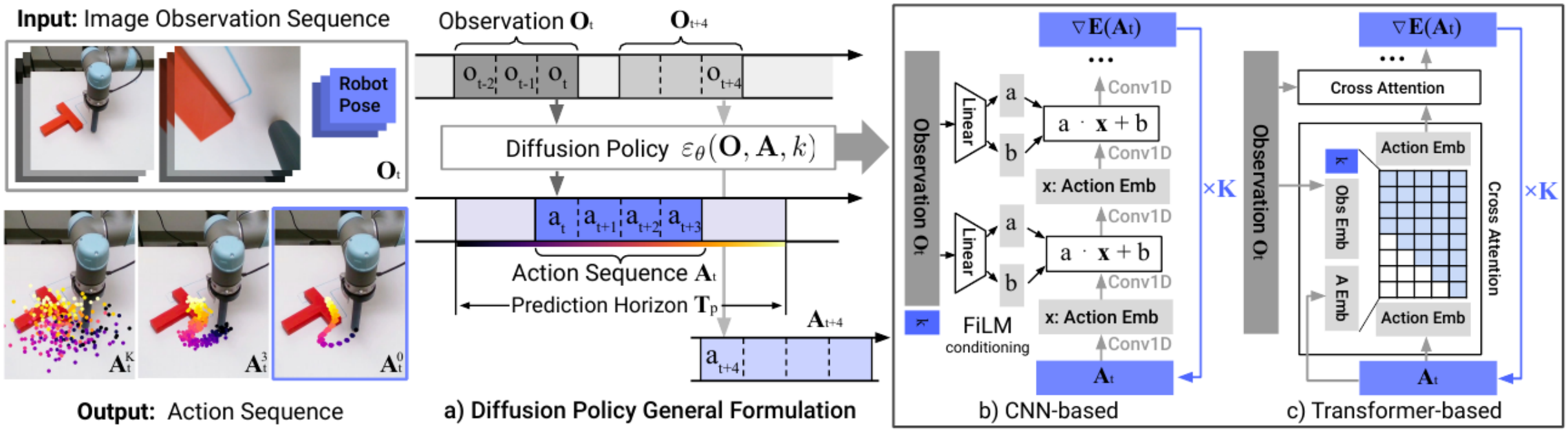
$$A^{(k)} = \sqrt{\bar{\alpha}_k} A^{(0)} + \sqrt{1 - \bar{\alpha}_k}\, \epsilon, \quad \epsilon \sim \mathcal{N}(0, I)$$

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{A,o,k,\epsilon}\left[\| \epsilon - \epsilon_\theta\left(A^{(k)}, k, o\right) \|^2\right]$$

Predict the noise added to an observation-conditioned action chunk

Reference: Chi et al. (2023/2024), Diffusion Policy: Visuomotor Policy Learning via Action Diffusion

Diffusion Policy: Control Loop

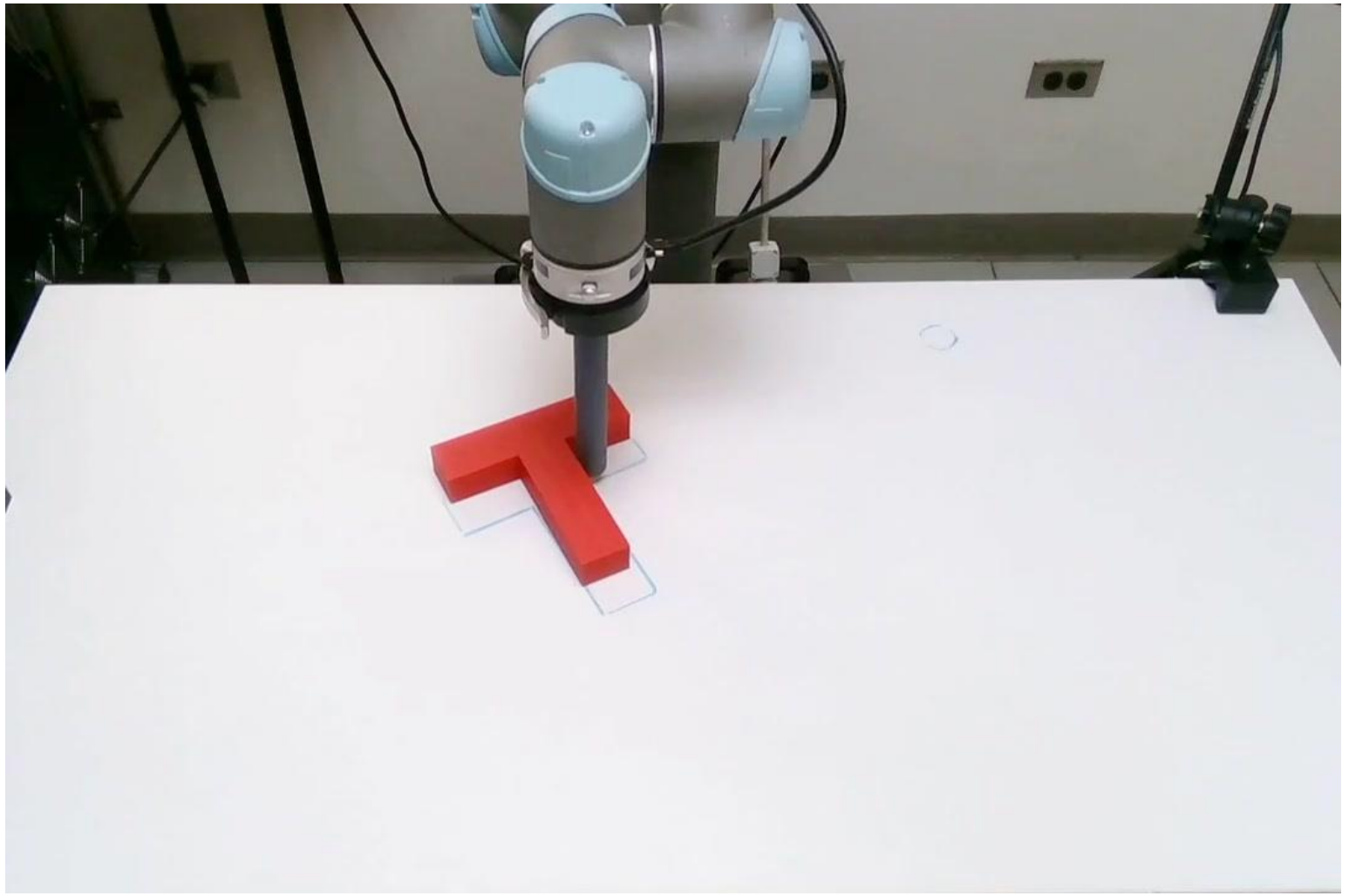


Images condition the action denoiser

Replan to balance computation, smoothness, and response time

Reference: Chi et al. (2023/2024), Diffusion Policy: Visuomotor Policy Learning via Action Diffusion

Diffusion Policy: Experiments



Intuition: Samples represent multiple strategies

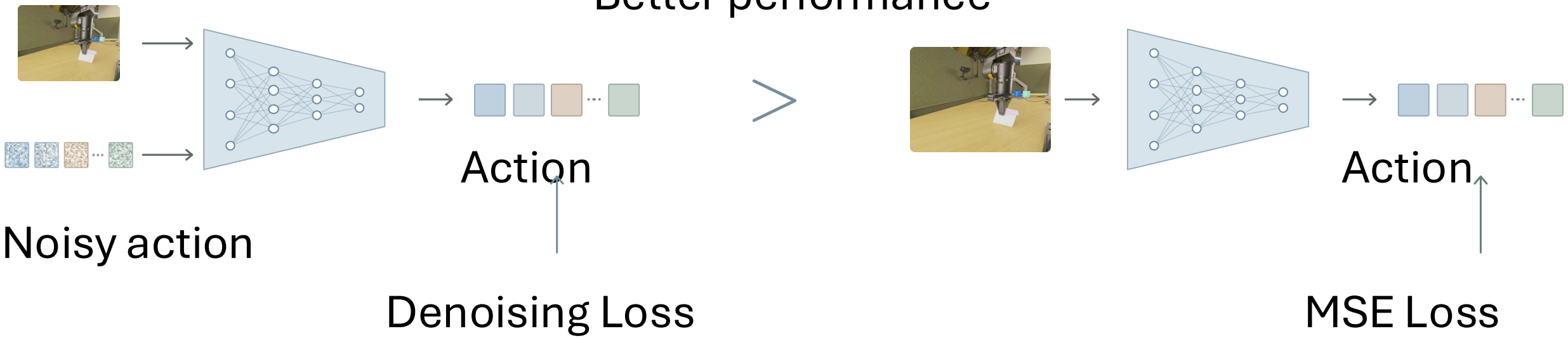
Reference: Chi et al. (2023/2024), Diffusion Policy: Visuomotor Policy Learning via Action Diffusion

Why Diffusion Works?

Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Generative Policies vs Regression Policies

Why are generative policies better than regression policies?

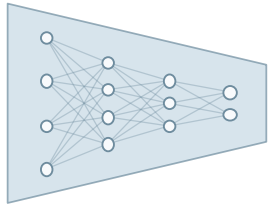


Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Action and Residual Distributions



Image

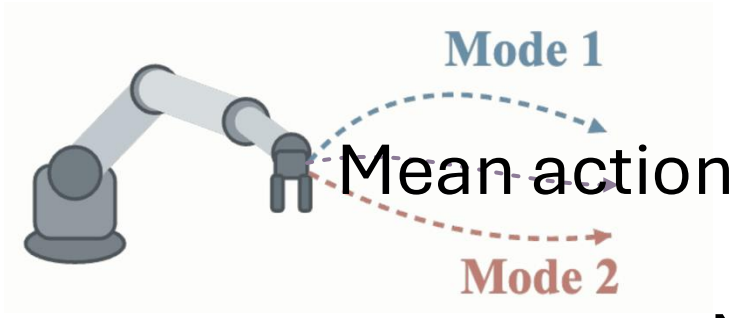


Regression policy



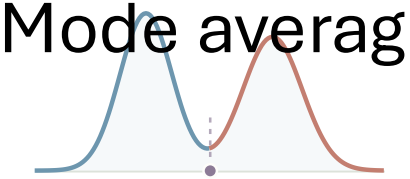
Action

Multi-modal action distribution



Mean action

Mode averaging

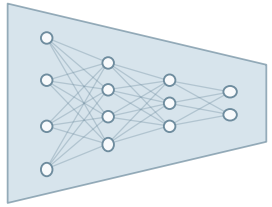


Mean prediction

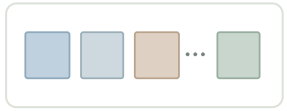
Our explanation: MSE assumes the wrong residual distribution



Image

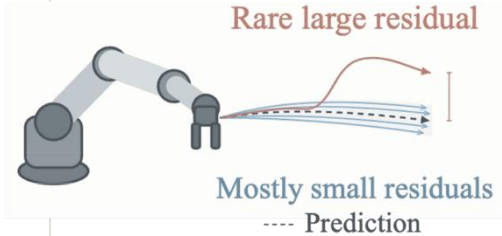


Regression policy

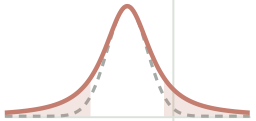


Action

Heavy-tailed residual distribution



Gaussian



Action residual

Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

MSE and Gaussian Residuals

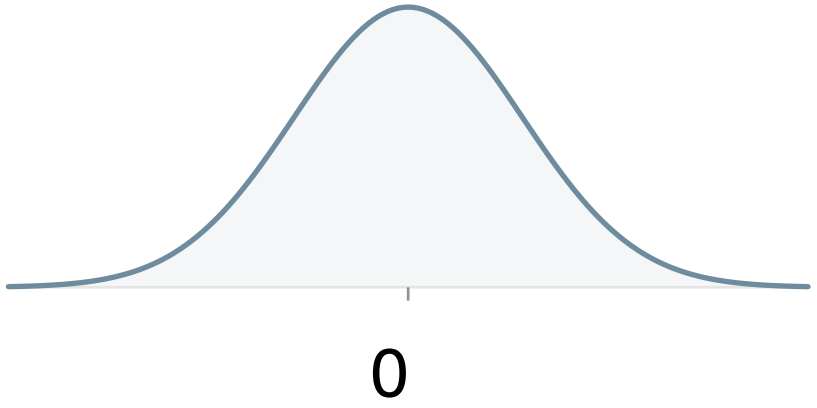
MSE loss

residual = pred - target

loss = residual.square().mean()

assumes

Fixed-scale Gaussian



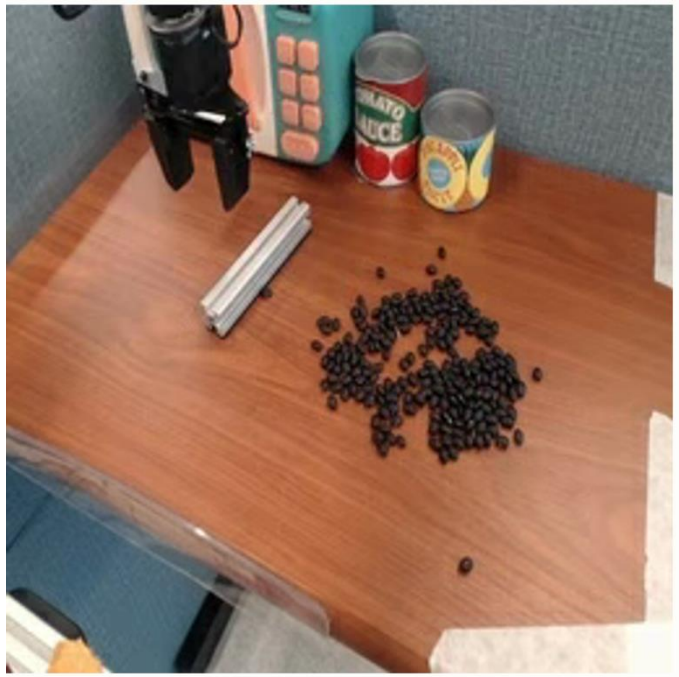
Action residual

Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Residual Scale Changes During a Task

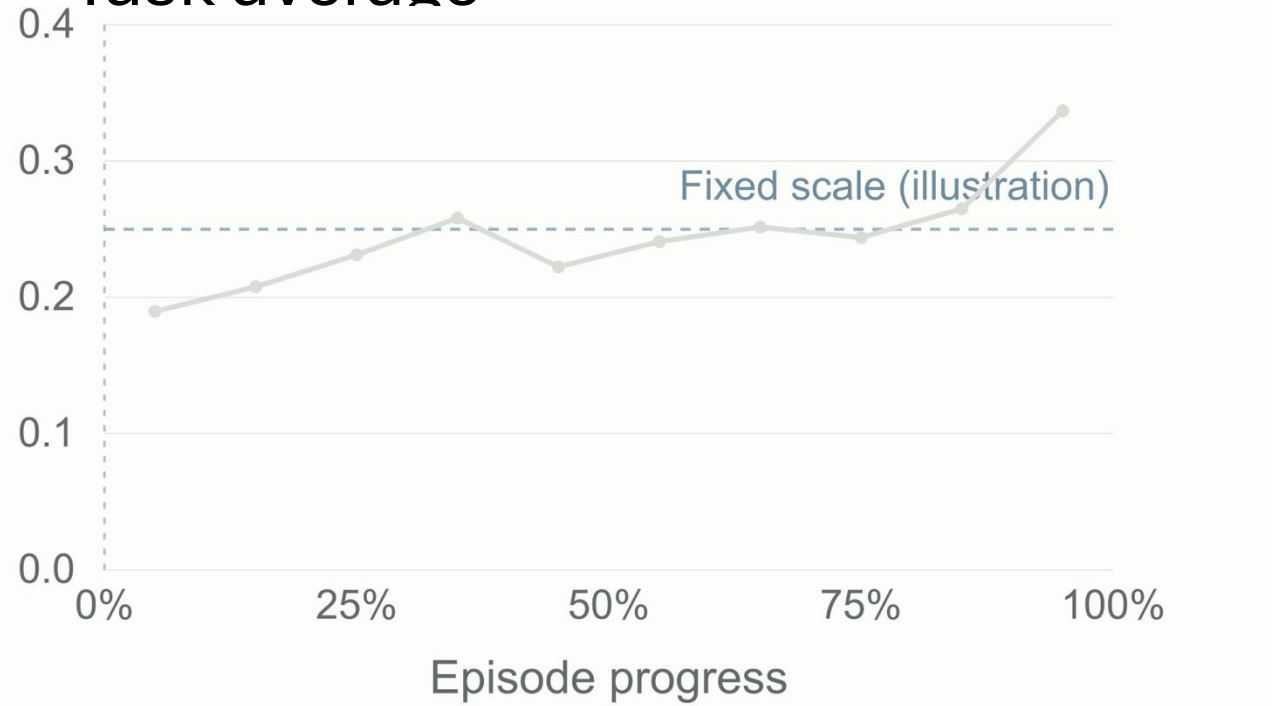
Sweep into Pile

Bridge · Example episode



Residual RMS

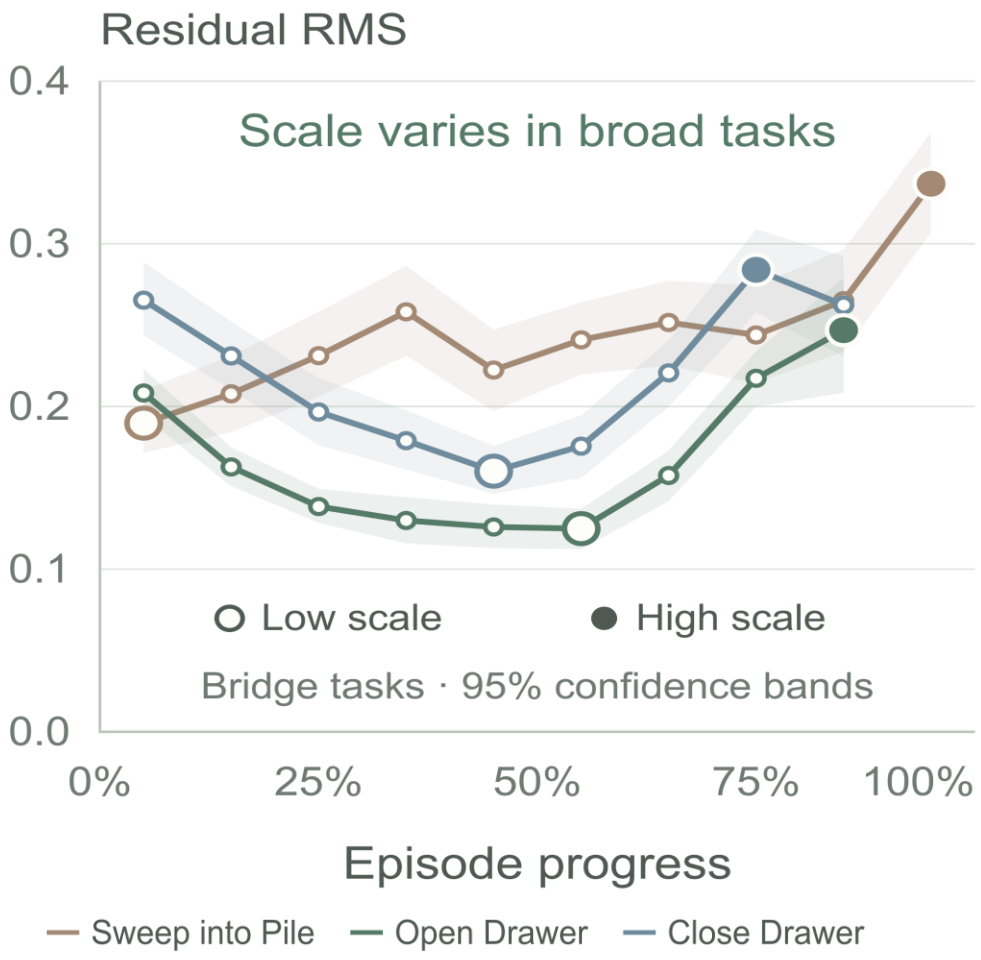
Task average



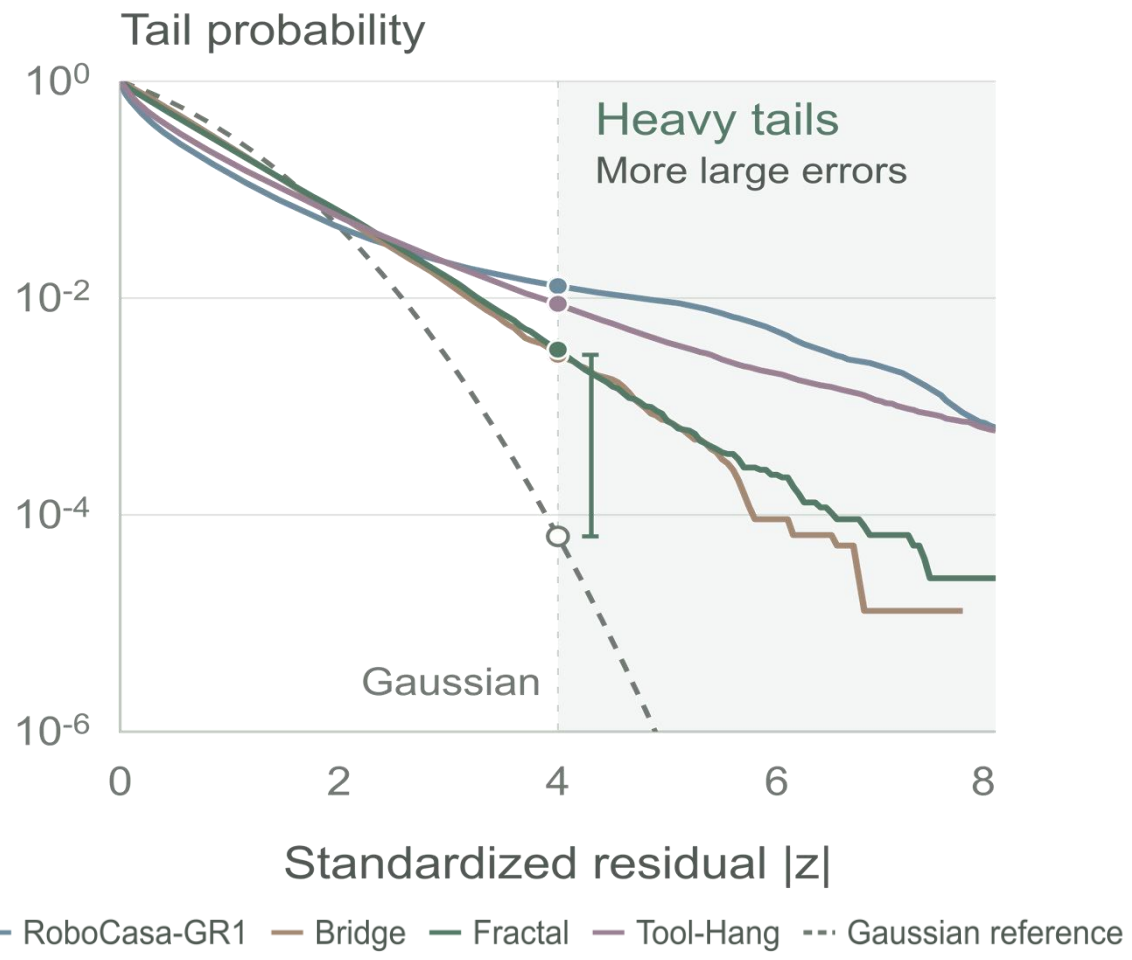
Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Residual Distributions Across Tasks

Task progress



MSE-Policy



Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Modeling the Residual Distribution



Image

Regression policy

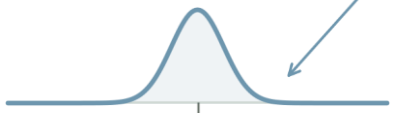
Action

Scale modeling



$\sigma_\theta(o)$





Small scale



Large scale

Explicit modeling with Student-t

Gaussian

Student-t



Heavy-tailed residuals

Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

HT-Policy: Training

$$\ell_{\mathrm{HT}}(o, a) = \frac{d}{2} \log \sigma_\theta^2(o) + \frac{\nu + d}{2} \log\left(1 + \frac{\|r\|_2^2}{\nu\, \sigma_\theta^2(o)}\right)$$

$r = f_\theta(o) - a$ d : action-chunk dimension ν : degrees of freedom

MSE-Policy

```
optimizer.zero_grad()
pred = policy(obs)
loss = mse_loss(pred, target)

loss.backward()
optimizer.step()
```

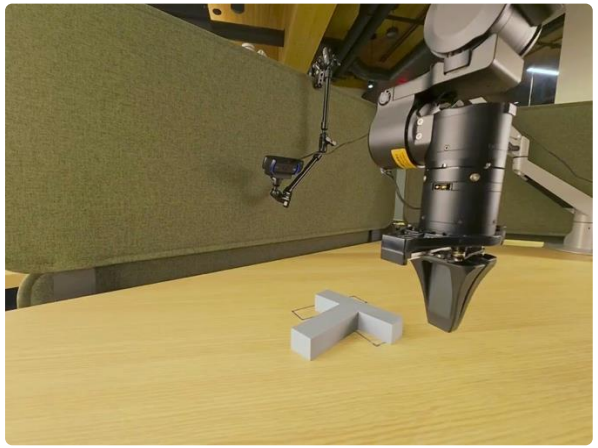
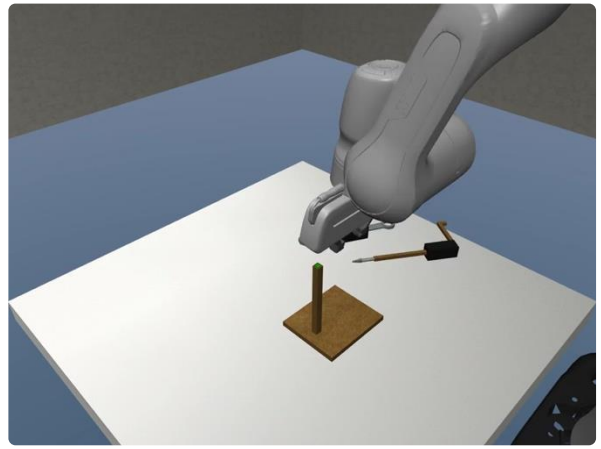
HT-Policy

```
optimizer.zero_grad()
pred, scale = policy(obs)
loss = ht_loss(
    pred, target, scale)
loss.backward()
optimizer.step()
```

Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

HT-Policy: Task Demonstrations

Visualization of HT-Policies across real-world and simulation tasks

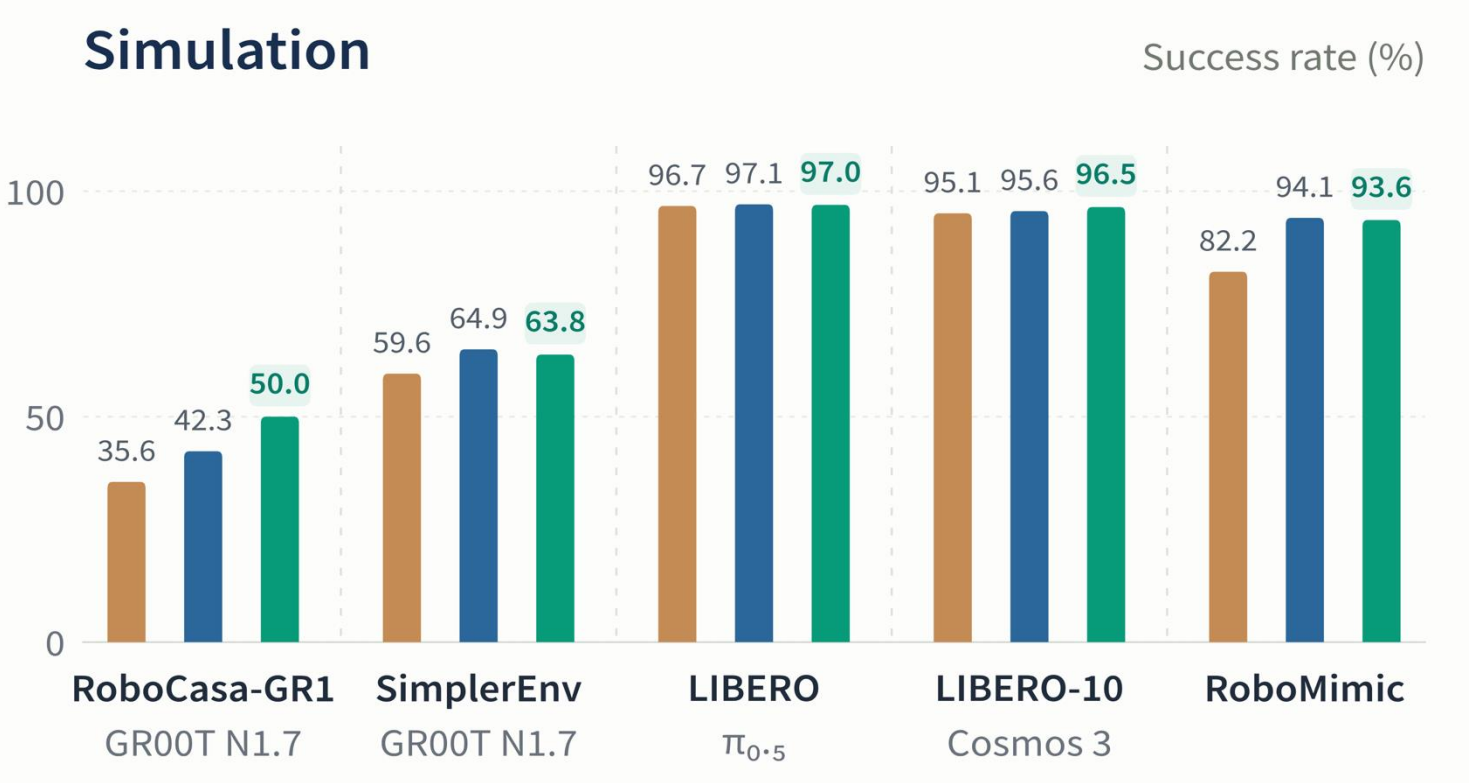
Peel Note	Push T	Store Bottle	Tool Hang
			
52.0% success	88.0% success	58.3% success	97.0% success
1.3× speed	3.8× speed	1.2× speed	1.7× speed

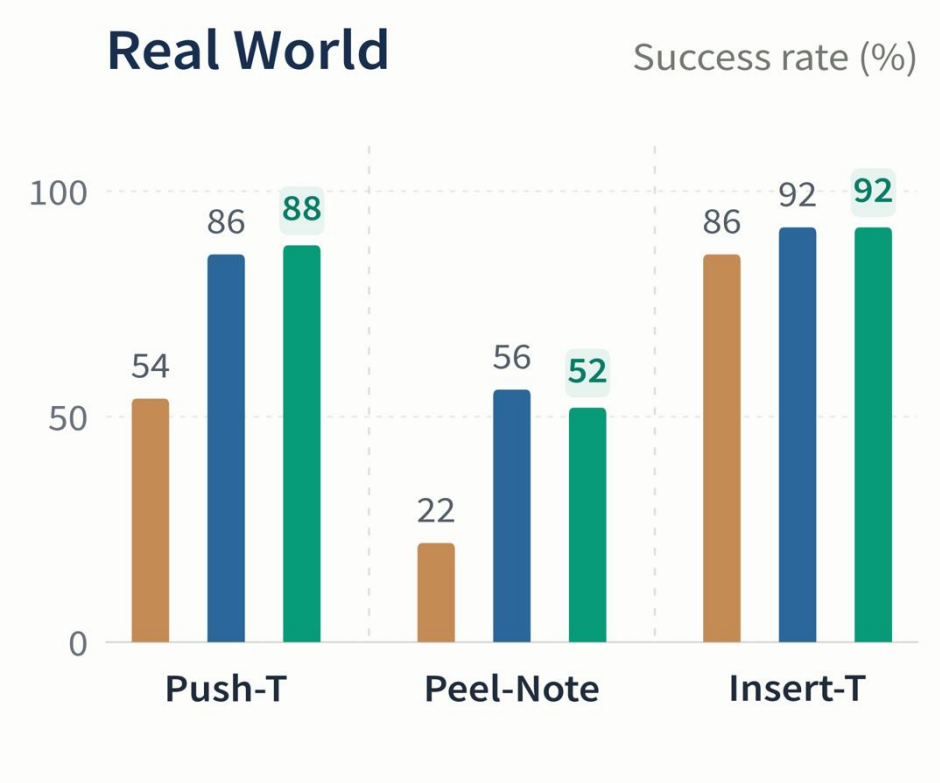
Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

HT-Policy: Performance

Performance

MSE-Policy Flow-Policy / DP HT-Policy (ours)

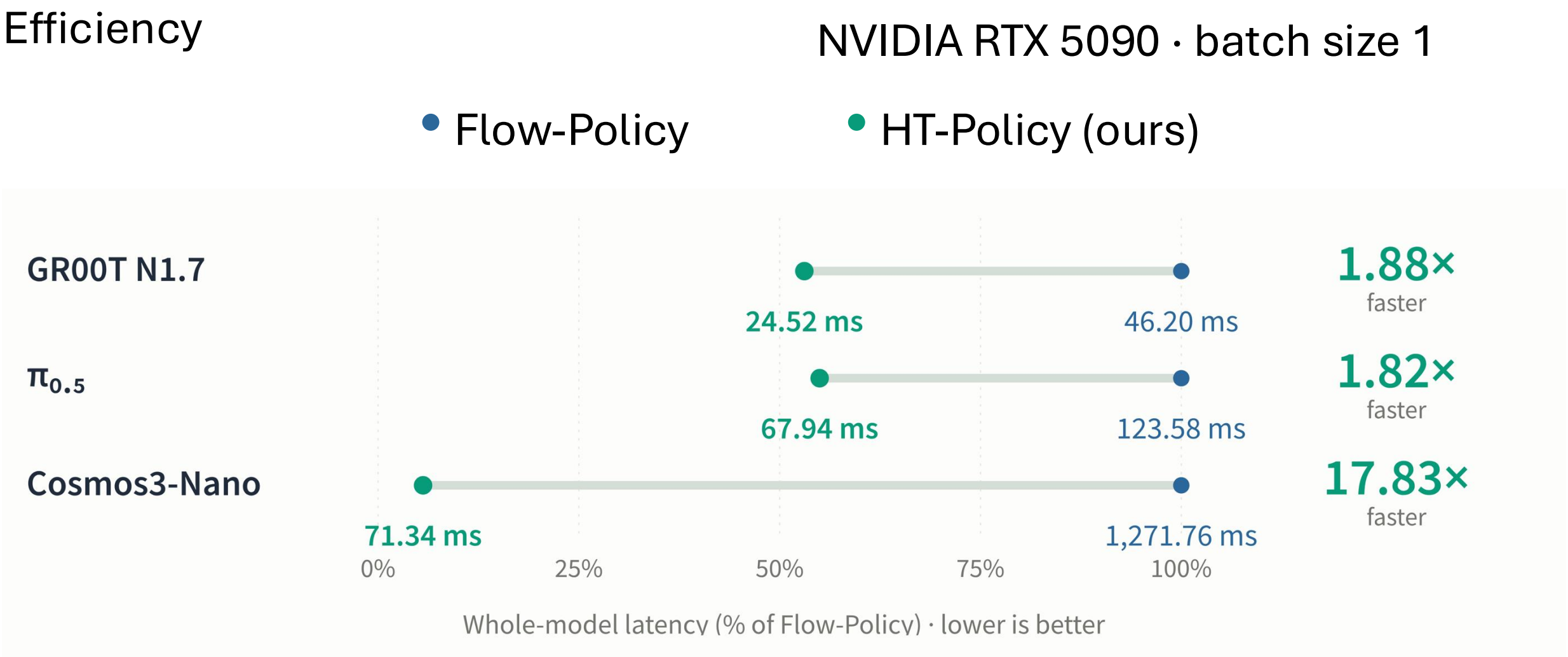




Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

HT-Policy: Inference Time

Efficiency



Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Why Diffusion Denoising Also Works?

Recall that a flow-matching network takes the policy inputs o, a timestep t, and a noisy action $a_t = ta + (1-t)\epsilon$, where $\epsilon \sim \mathcal{N}(0, I_d)$, and minimizes the squared velocity residual $v_\theta(a_t, t, o) - (a - \epsilon)$. For any fixed $t > 0$, consider a naïve baseline predictor $\tilde{v}(a_t, t, o) = a_t/t$. Its velocity residual is

$$\tilde{v}(a_t, t, o) - (a - \epsilon) = \frac{a_t}{t} - a + \epsilon = \frac{\epsilon}{t}.$$

For this baseline model, its network prediction residual is distributed as $\mathcal{N}(0, I_d/t^2)$, regardless of the overall action distribution. Therefore, the flow-matching objective admits a predictor whose residual scale depends only on the injected noise and the timestep.

Empirically, we found that a well-trained predictor that is conditioned on o also inherits this behavior in the sense that, at low-noise timesteps, the learned predictor has tails no heavier than this reference:

$$\Pr[\|v_\theta(a_t, t, o) - (a - \epsilon)\|_2 > \delta \mid t] \leq \Pr[\|\epsilon\|_2 > t\delta] \quad \text{for sufficiently large } \delta.$$

Reference: Zhou et al. (2026), Residual Modeling Closes the Regression and Generative Policy Gap in Robot Learning

Questions for Discussion

- What do you think are the pros and cons for FAST, based on our course material?
- What do you think are the pros and cons for RT-2 style text-based action representations?
- What do you think are the pros and cons for using EE-based representations for “cross-embodiment generalization”?
- What do you think are the pros and cons for using demonstrations to specify a learning target for the robot?