Penn Engineering
University of Pennsylvania

GMLR

CIS 6280 · FALL 2026

WORLD MODELS

LECTURE 17

World Models for Robotics I
Planning and Policy Learning

Jiatao Gu

University of Pennsylvania · November 10, 2026

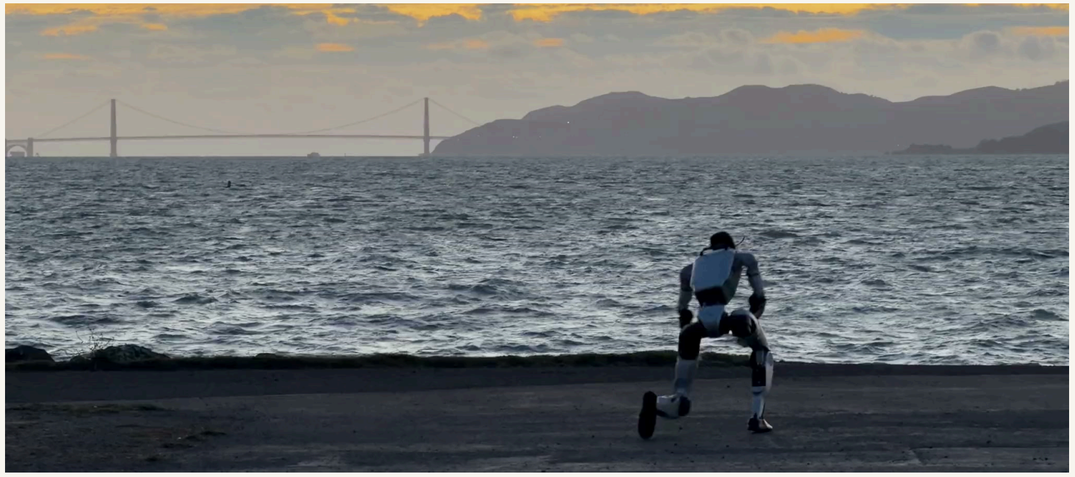
Course announcements

Assignment 3	Due November 18
Final project	Data are available. See Canvas for instructions and updates.
Cloud compute	US$100 per team More details to come.

Robots acting in the physical world



Manipulation: move an object toward a goal.



Whole-body control: coordinate motion and contact.

Actions change both the robot and what it will observe next.

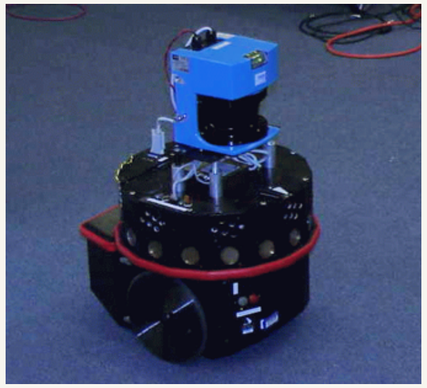
Huang et al., PointWorld (2026) · Liao et al., SuperMimic (2026)

What is a robot?

A robot is a programmable physical system that senses and acts in the world to perform tasks.



Manipulator



Mobile robot



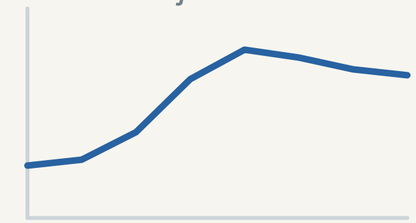
Humanoid

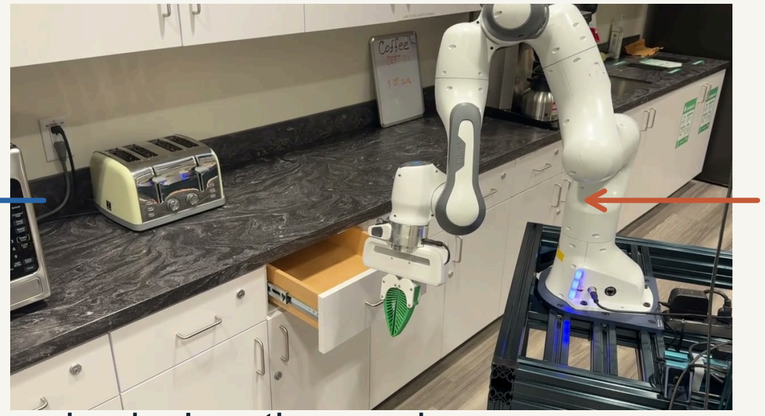
Body carries the system. **Sensors** measure. **Software** computes. **Actuators** apply forces.

A robot in a closed loop

Sensor signals

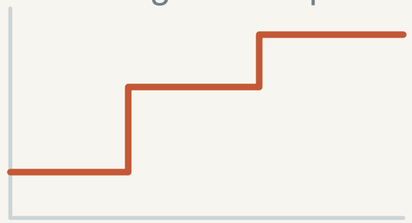
measured joint motion





Command

target or torque



A new physical motion produces a new measurement.

The next input depends on the action we just took.

Two uses of a world model

Planning

choose actions for the current situation

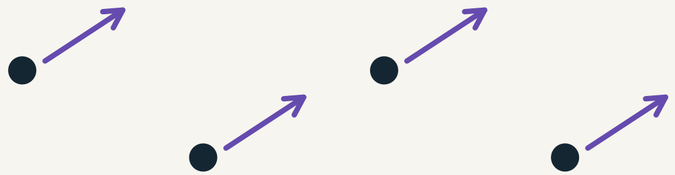


Optimize the current command sequence.

Policy learning

improve actions across situations

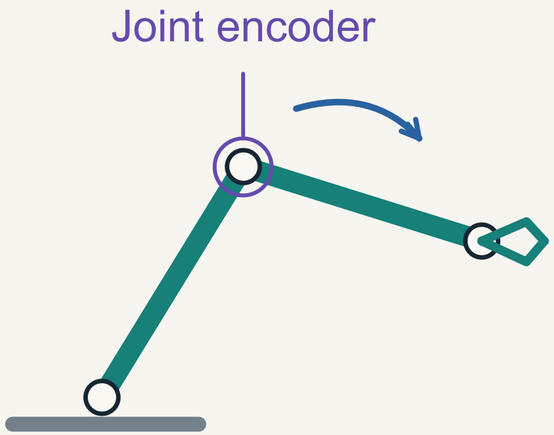
π_θ(observation, goal)



Fit one function shared by those decisions.

Planning optimizes a decision now. Policy learning improves decisions across situations.

Proprioception: sensing the robot’s own body



Joint angles q

how the joints are positioned

Joint velocities $\dot{q}$

how fast the angles are changing

These signals remain available when the hand is out of the camera’s view.

Proprioception: sensor measurements of the robot’s own configuration and motion.

Actual angles and velocities are state variables. Their measured values are observations.

Humanoids sense more than joint angles



Humanoid example: SuperMimic.
Sensor categories vary by robot.

Joint encoders

Joint angles; velocities from their changes.

IMU: inertial measurement unit

Gyroscope: body rotation rate.
Accelerometer: acceleration readings used with gravity to estimate body motion.

Contact / force sensing

Foot pressure or force sensors, when available, provide evidence about ground support.

Body tilt and velocity must be estimated from these signals; an IMU does not directly measure position.

SuperMimic: humanoid example · ROS REP 145: IMU conventions · Hartley et al., Contact-Aided State Estimation

Dexterous hands: joint state and touch



Shadow Dexterous Hand.
Tactile sensing depends on the configuration.

Proprioception: finger motion

Actual joint angles q and velocities $\dot{q}$.
Encoders provide measurements of the hand's motion.

Touch: hand–object contact

Pressure readings or tactile images help infer where contact occurs, how the object is loaded and whether it slips.

Object state

Object position, orientation and motion also matter.
Vision and touch help estimate whether it is held or slipping.

Action: desired finger positions or motor torque commands, depending on the controller.

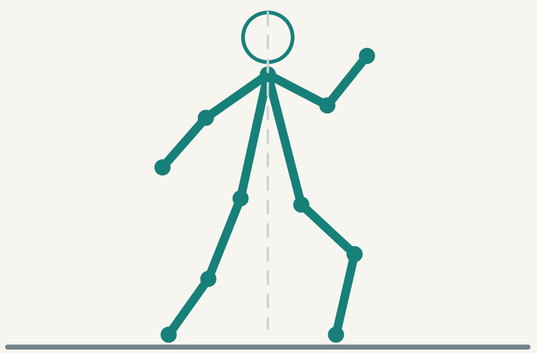
The world model predicts how the hand and the grasped object respond to a command.

Shadow Robot · Dexterous Hand and sensing options · Shadow Robot · joint and tactile measurements · DIGIT · tactile observations

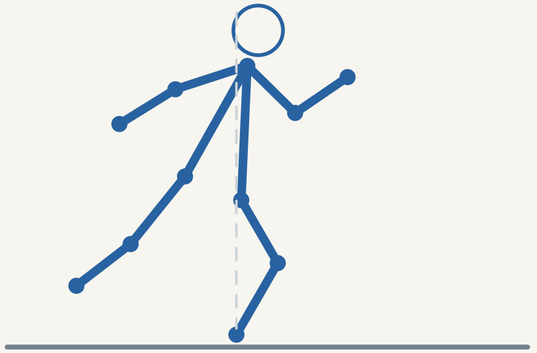
State estimation: what is the body doing now?

Same joint angles

different orientation of the whole body



Encoders describe the joints.



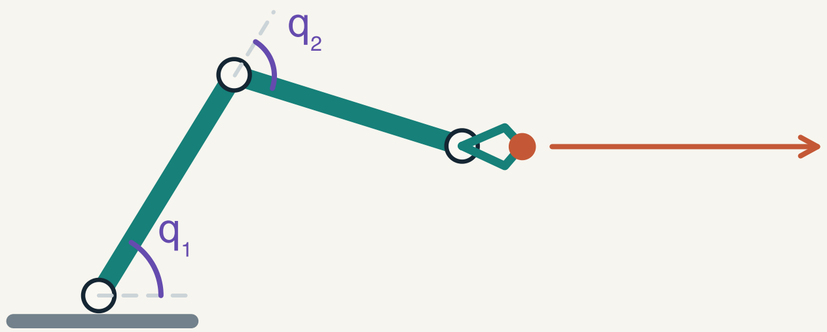
IMU readings help distinguish body tilt.

Combine joint measurements, IMU readings and contact information over time.

Estimated state: joint configuration and motion, plus body orientation and velocity.

This estimated current state can initialize the world model’s next prediction.

Forward kinematics: where is the gripper now?



Known link lengths + current joint angles

Gripper pose now

position and orientation

Computed from the same configuration.

No future motion is predicted here.

$$\text{gripper pose now} = k(q; \text{robot geometry})$$

q = current joint angles. Pose = position and orientation. k = the geometric calculation.

Kinematics locates the robot’s parts from its current joint configuration.

Dynamics: what happens after a command?



A commanded target is not the same as the resulting joint position.

$$\underbrace{(q_{t+1}, \dot{q}_{t+1})}_{\text{next actual joint state}} = f\left(\underbrace{(q_t, \dot{q}_t)}_{\text{current actual joint state}}, \underbrace{q_t^{\text{target}}}_{\text{one action interface: } a_t}\right)$$

The world model predicts the resulting motion, including the controller's response.

What exactly does the action command?

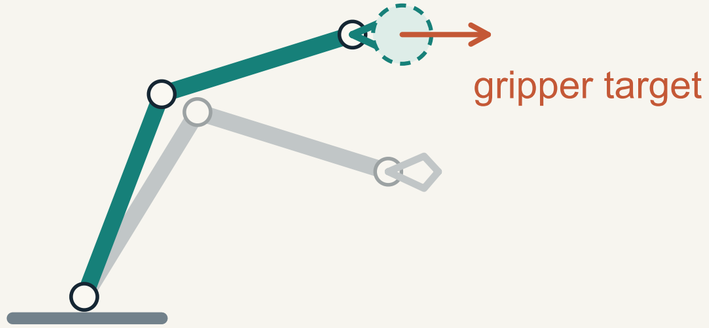
Action chosen by the agent	What executes it	Result to predict
Joint-angle target	A joint controller tracks the target	Actual joint angles and velocities
Joint-velocity target	A controller tracks the requested velocity	Actual joint angles and velocities
Joint torque	Actuators apply the commanded torque	Actual joint angles and velocities
Gripper pose target	Typically: inverse kinematics (IK), then joint control	Actual gripper and joint motion

The action is the input at the interface we choose to model.

From a gripper target to physical motion

1. Inverse kinematics (IK)

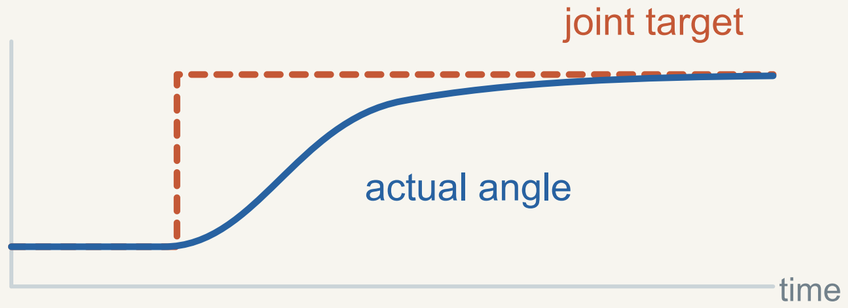
find joint angles for the desired gripper pose



A geometric solution, before any motion.

2. The controller moves the robot

actual joint motion follows over time



Motors, inertia and contact affect the response.

$$\underbrace{k(q^{\text{target}})}_{\text{forward kinematics}} = \underbrace{\text{desired gripper pose}}_{\text{given target}} \qquad \text{IK solves for } q^{\text{target}}$$

IK may have several solutions, or none. Joint limits and collisions constrain the choice.

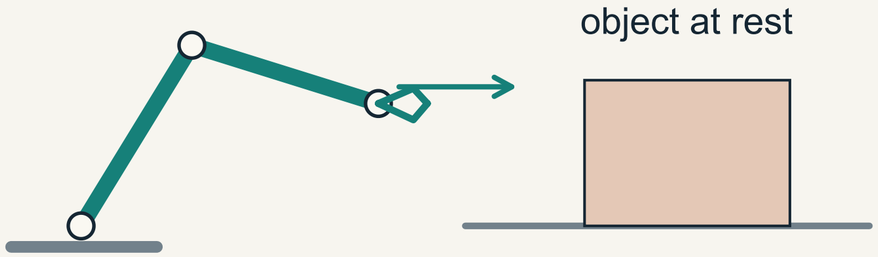
A world model receiving gripper targets must capture the downstream execution and physical response.

Hauser, Inverse Kinematics · Hauser, Robot Control

Contact changes the consequences of an action

Before contact

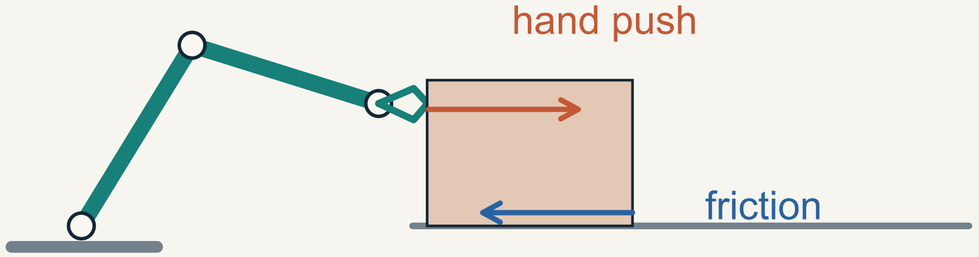
the command moves the arm



No hand force reaches the object.

During contact

forces determine the response



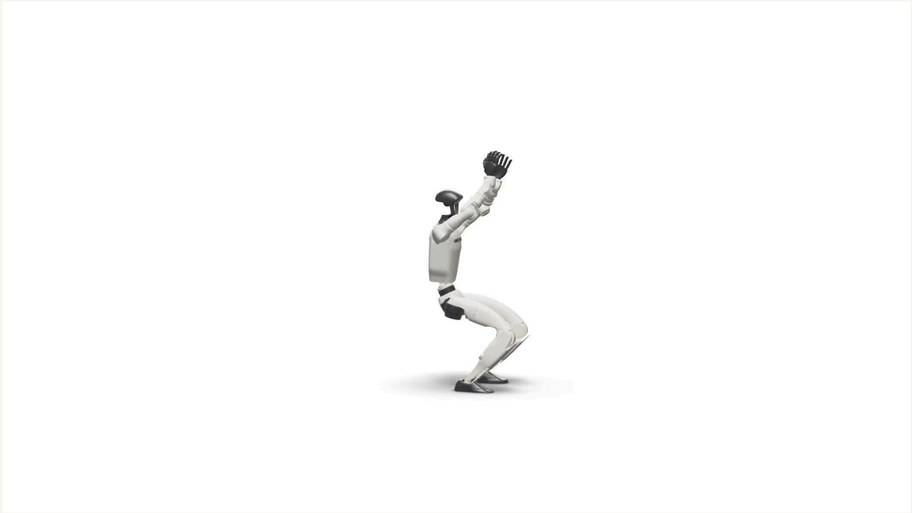
The object may stay still or start sliding.

$$\underbrace{m\,\dot{v}}_{\text{mass} \times \text{horizontal acceleration}} = \underbrace{F_{\text{hand}}}_{\text{push from the robot}} + \underbrace{F_{\text{friction}}}_{\text{force from the table}}$$

For the object on a level table; forces have signs. Friction can balance the push.

A manipulation world model must predict the object's response as well as the robot's motion.

A simulator is a world model



SuperMimic: simulated robot motion.

Physical state

Body pose, joints, velocities and contacts.

Step

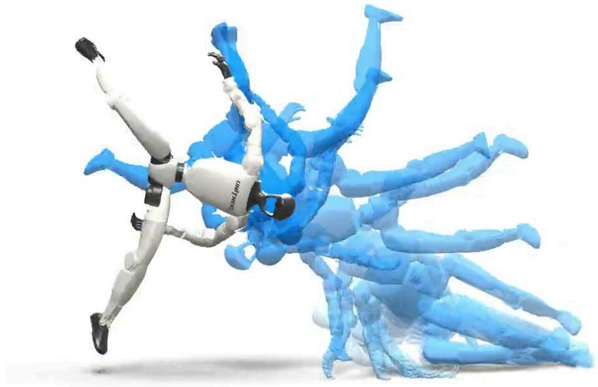
Apply a command and advance the state.

Render

Produce the camera view shown here.

The simulation step and its camera rendering are different operations.

Learning a dynamics model



Policy switched off: simulator motion and predicted continuation.

Gray / white: simulator Blue: world-model predictions

Blue silhouettes show predicted poses at successive times.

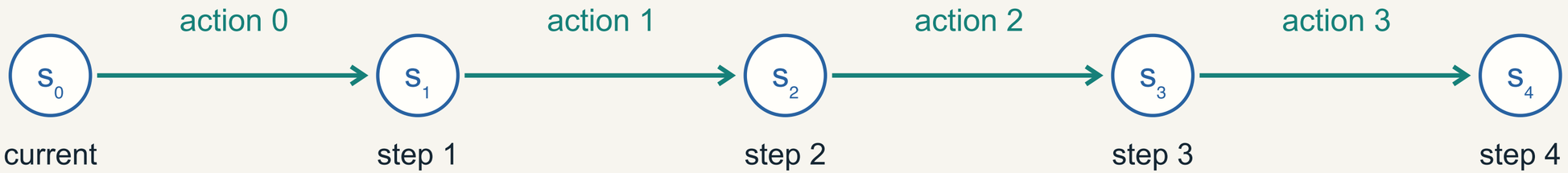
Fit what happens after a command.

$$\hat{s}_{t+1} = f_\omega(s_t, a_t)$$

ω: model weights. A hat marks a prediction.

Predicting a fall is a dynamics task; preventing it is a control task.

What is one training example?



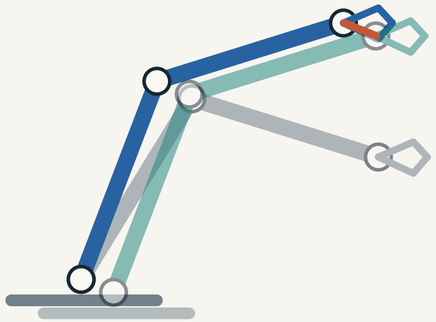
Input before the transition	Input during the transition	Supervision after the transition
State or observation history	Executed action and its duration	Next state or next observation

Learning one-step predictions

$$\hat{s}_{t+1} = f_\omega(s_t, a_t)$$

Same recorded transition

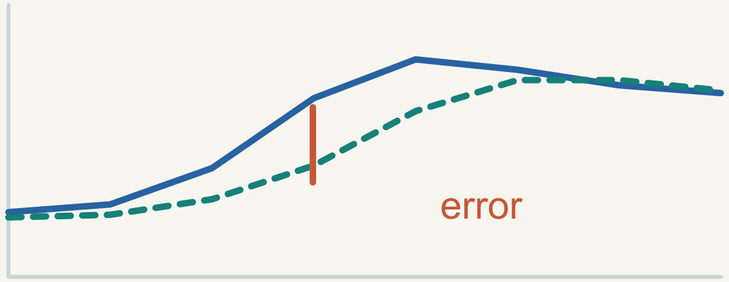
compare corresponding quantities



Pose error is measured at the same time.

Motion must match too

measured and predicted velocity

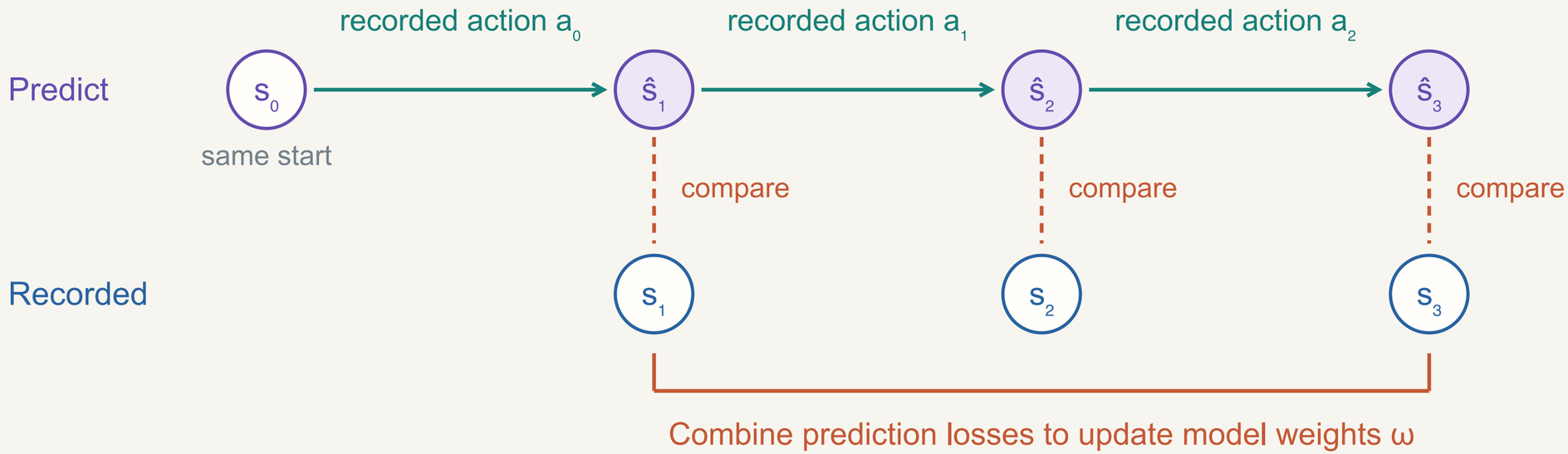


Solid: recorded. Dashed / pale: predicted.

$$\mathcal{L}_{\text{model}} = \mathbb{E}\left[\|\hat{s}_{t+1} - s_{t+1}\|_W^2\right]$$

W sets the relative scales of the state components.

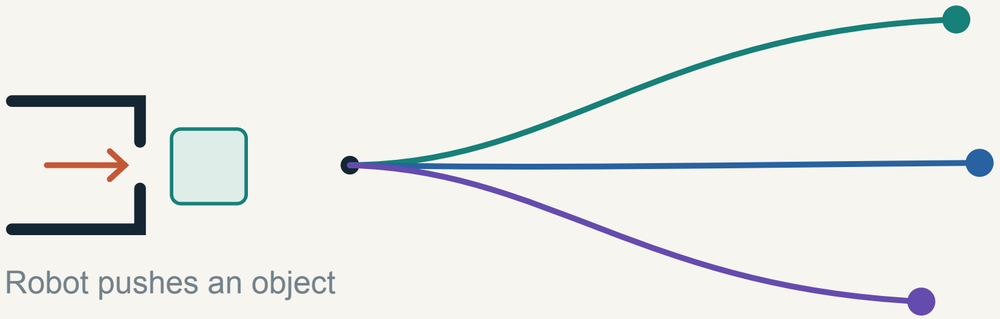
Fitting a deterministic multi-step rollout



Replay the recorded commands, compare every predicted outcome, and update the model.

A world model can predict a distribution

Same observed start. Same push command.



Robot pushes an object

Possible object paths — qualitative illustration

Unobserved contact details and friction affect the outcome.

One mean prediction can hide several plausible futures.

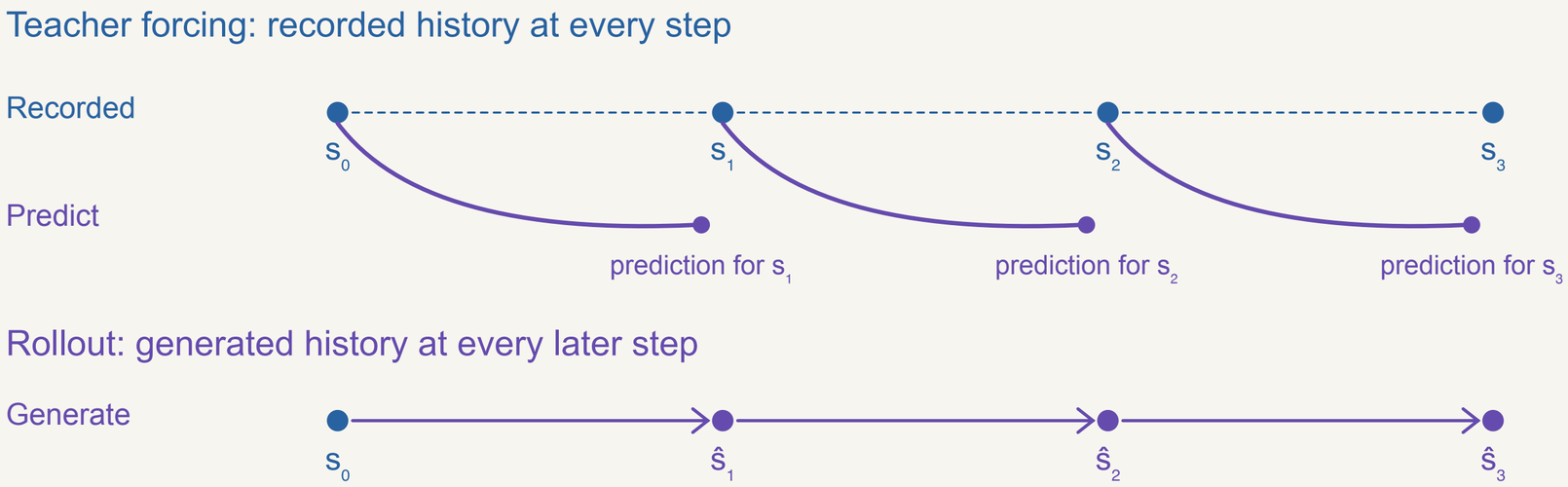
$$\hat{s}_{t+1} \sim p_\omega\left(\cdot \mid s_{\le t}, a_{\le t} \right)$$

pω: learned next-state distribution. ≤ t: available history. ~: sample a possible outcome.

Uncertainty is relative to what the robot knows about the current world.

Chua et al., PETS (2018)

Teacher forcing: learn from recorded histories



$$\mathcal{L}_{\mathrm{TF}} = -\sum_{k=0}^{H-1} \log p_\omega\left(s_{k+1} \mid s_{\le k}, a_{\le k}\right)$$

All s values are recorded; H = sequence length. Diffusion models use conditional denoising instead.

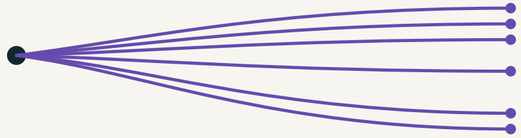
Many supervised steps in one sequence; each prediction still receives the recorded past.

Huang et al., Self Forcing (2025)

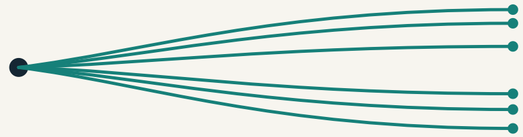
DMD can train the distribution of rollouts

Hold the starting information and action sequence fixed

World-model rollout samples



Teacher target distribution



Match the distribution of whole trajectories — not one paired path.

Generate	Feed each sampled prediction into the next step: self-rollout.
Compare distributions	Noise the trajectory. Evaluate teacher and generated-distribution scores.
Update the world model	Their score difference supplies a gradient through the generated trajectory.

Score: ∇ log density. The generated-distribution score is fitted to current model samples.

For robotics, the teacher must model the consequences of the same commands.

Huang et al., Self Forcing (2025) · Yin et al., DMD (2024)

Physics, learned models, and residual models

Construction	Prediction	What is fitted
Physics model	$\hat{s}_{t+1} = f_{\text{physics}}(s_t, a_t)$	Physical parameters or nothing
Learned model	$\hat{s}_{t+1} = f_\omega(s_t, a_t)$	The state-update function
Residual model	$\hat{s}_{t+1} = f_{\text{physics}}(s_t, a_t) + \Delta_\omega(s_t, a_t)$	A correction to the baseline

The additive form assumes compatible state coordinates. Rotations need a suitable update rule.

So far, the output has been physical state. What else could we predict?

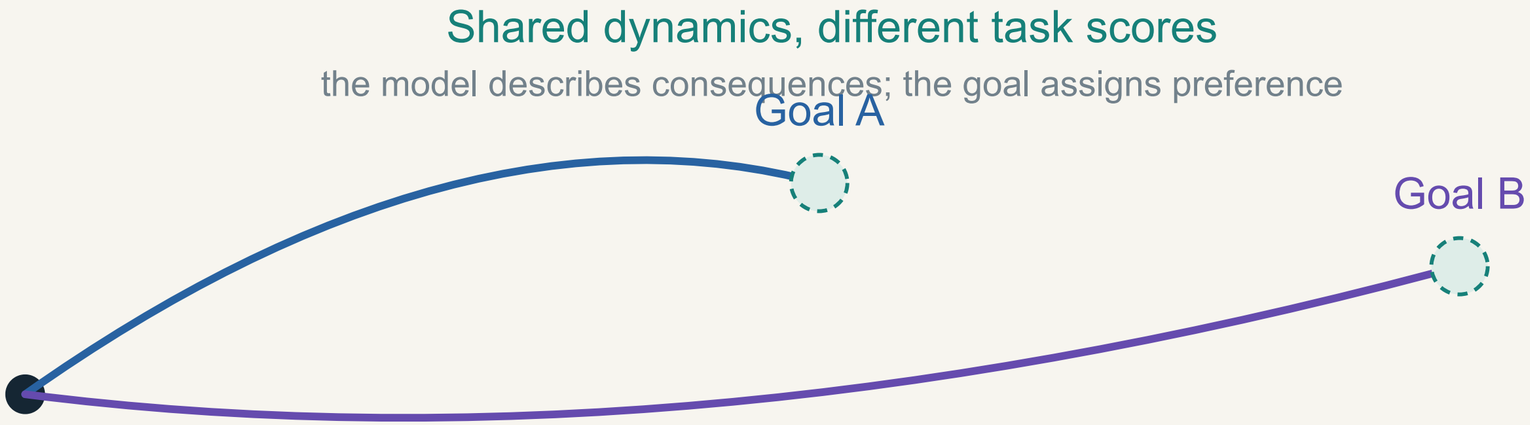
Robot interaction in different representations

The models from earlier lectures can predict the same robot interaction.

Representation	Predicted output after the command
Physical state	Next joint angles, velocities and object pose
Latent features	Learned features of the next observation
3D points	Next positions of points on the robot and objects
Images / video	Next views of the robot and scene

Each representation must capture the consequences of the robot's command.

The same dynamics can serve different goals



Changing the goal need not change the physical transition rule.

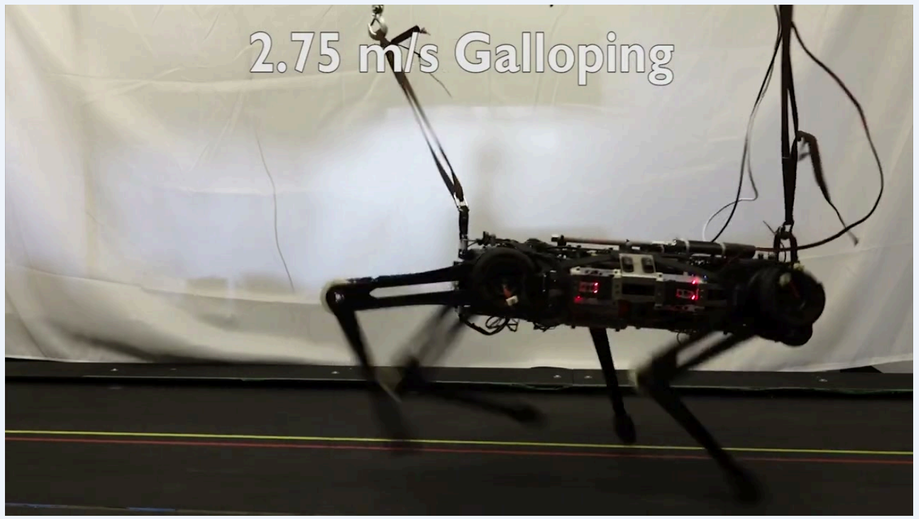
Predict what happens

How will the body and environment change?

Decide what we want

Which predicted outcome helps the task?

MIT Cheetah: choose how the feet push



MIT Cheetah 3 · official MPC experiment

Given

Desired body motion and which feet will touch the ground.

Predict

How the body moves under different foot forces.

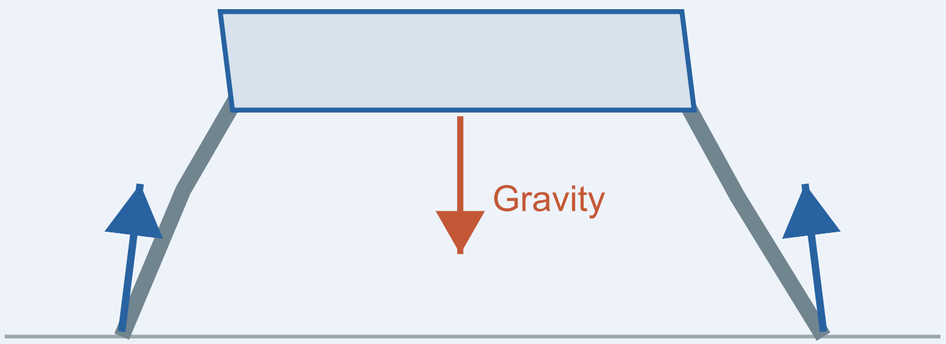
Choose

Forces that follow the command and keep the body balanced.

The world model predicts body motion; MPC chooses the forces.

The model predicts how pushes move the body

Predict the motion of the body



The ground pushes back on supporting feet

Side view; two supporting feet shown

Current body state

Position, tilt, and motion, estimated from sensors.

Candidate forces

How each supporting foot pushes against the ground.

Predicted body motion

Where the body moves and how it tilts next.

Different pushes change both the body’s movement and its balance.

Optimize a sequence of foot forces

Use the model to predict body motion over the next few control intervals.

$$\min_{\text{future foot forces}} \Big[\text{body-motion error} + \lambda\,\text{force penalty}\Big]$$

Body-motion error

How far are the predicted height, tilt, and velocity from the command?

Force penalty

Discourage unnecessarily large forces.

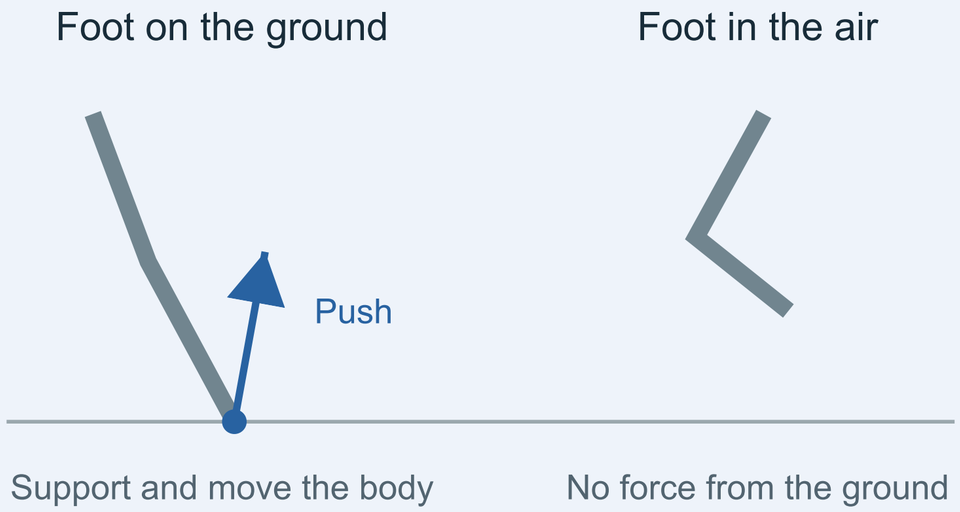
λ sets the trade-off between the two terms.

Add these costs over the prediction horizon. Allow only feasible foot forces.

Change the proposed forces, predict the resulting motion, and minimize the total cost.

Di Carlo et al., Convex MPC for MIT Cheetah 3 (IROS 2018)

The feet cannot produce arbitrary forces



A supporting foot can push

Ordinary contact cannot pull on the ground.

Sideways push is limited

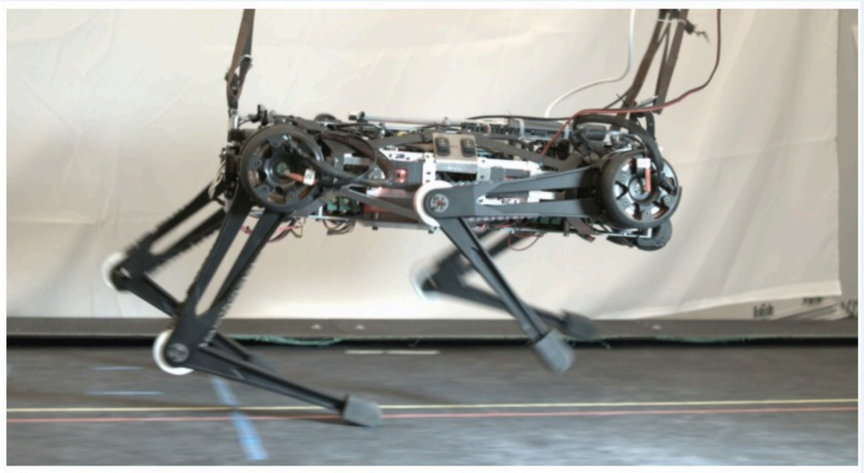
Too much sideways force would make the foot slip.

A lifted foot cannot support the body

Its ground contact force must be zero.

The gait schedule tells MPC which feet can exert forces at each future step.

The leg controller turns forces into motor commands



MIT Cheetah 3 · Di Carlo et al. (2018)

Feet in the air follow a separate swing trajectory.

MPC chooses foot forces

Take only the forces for the next control interval.

The leg controller computes joint torques

Use the current leg configuration to realize those forces.

The motors apply torque

Sensor feedback corrects the actual motion.

Foot force is the MPC decision; joint torque is the actuator command.

Execute briefly, measure, and replan

Plan from the current measured state



Measure again; update the plan



Only the orange interval is executed.

Measure

Estimate the body’s current position, tilt, and motion.

Plan

Predict future motion and optimize the foot forces.

Execute

Apply only the first short part of the plan.

Replan

Measure the actual result and solve again.

Every new plan starts from what the robot actually did.

Di Carlo et al., Convex MPC for MIT Cheetah 3 (IROS 2018)

A learned model can fill the same prediction role



PointWorld · real robot closing a drawer

Model input

Scene points and candidate robot-point motion.

Predicted outcome

How scene and object points move in response.

Task objective

Score predicted object motion against the desired outcome.

Optimize commands, execute a short prefix, and observe again.

Huang et al., PointWorld (2026)

Physical and learned models support the same loop

	MIT Cheetah 3	PointWorld-style manipulation
Model	Simplified body physics	Learned scene-point dynamics
Candidate action	Forces at the feet	Robot motion commands
Prediction	Body position, tilt and motion	How scene and object points move
Goal	Follow the desired body motion	Move the object to its target
Execution	Apply briefly, then observe again	Apply briefly, then observe again

Predict consequences, choose actions, execute, and use new feedback.

Di Carlo et al., Convex MPC for MIT Cheetah 3 (IROS 2018) · Huang et al., PointWorld (2026)

Can action selection be learned ahead of time?

MPC on the robot

At each new state, solve for a new sequence of actions.

Optimize online; apply the next action.

The predictive model is used during online action selection.

o_{now}: available observations; g: task goal; θ: shared policy parameters.

A learned policy on the robot

Use a trained function to choose the next action.

$$a_{\text{now}} = \pi_\theta(o_{\text{now}}, g)$$

World-model rollouts can be used to train that function.

Next: use predicted consequences to improve the policy itself.

Di Carlo et al., Convex MPC for MIT Cheetah 3 (IROS 2018) · Hafner et al., Dreamer

World Models for Learning Policies



SuperMimic: imagined body motion under policy actions.

$$a_t = \pi_\theta(o_t, g_t)$$

Fit a reusable mapping from observations and a goal to commands.

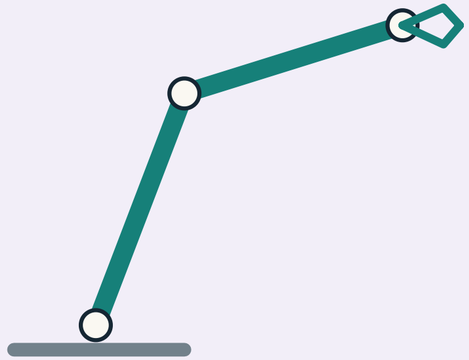
The model supplies experience or derivatives during training.

The same policy parameters are reused across states and control steps.

From predicted states to policy experience

Predicted physical state

configuration and motion



Two quantities for policy learning

both obtained from the predicted state

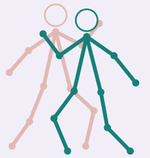
Observation

joint and inertial readings



Reward

match the reference goal



Goal information defines the task comparison.

The policy needs observations to act and rewards to evaluate its behavior.

Short rollouts and long-term return

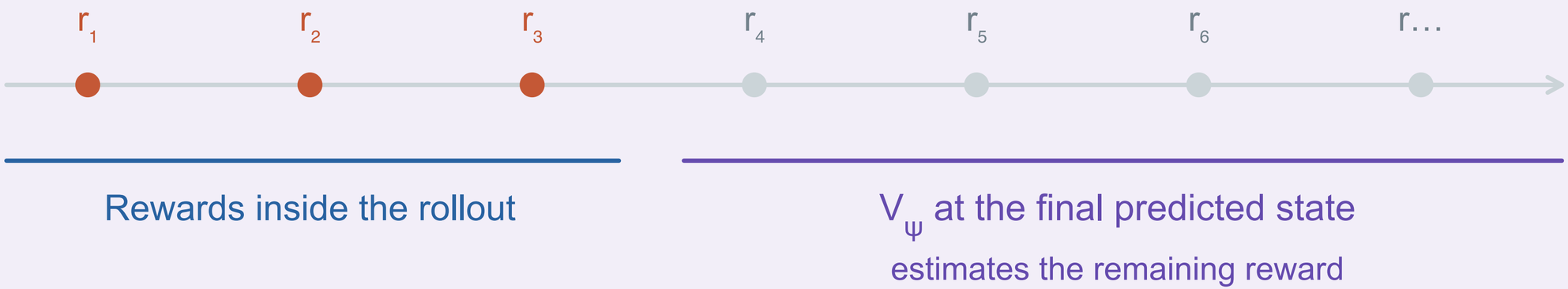


$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots$$

Return G is accumulated future reward. γ discounts rewards farther in the future.

Hafner et al., Dreamer (2020)

A value model estimates the remaining return



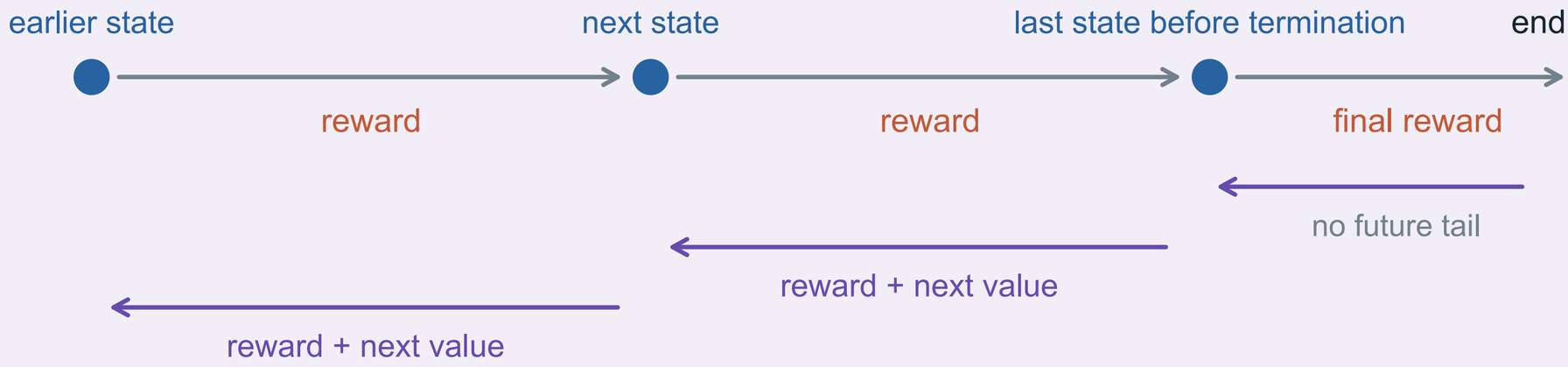
$$V_\psi(s, g) \approx \mathbb{E}[G \mid s, g, \text{follow the current policy}]$$

The critic is the value model. Its learned parameters are ψ.

Hafner et al., Dreamer (2020)

Where does the critic's supervision come from?

Each target combines a reward with the return that remains.



$$y_t = r_{t+1} + \gamma \, \mathrm{stopgrad}[V_{\bar{\psi}}(s_{t+1}, g_{t+1})], \qquad \mathcal{L}_V = \mathbb{E}[(V_\psi(s_t, g_t) - y_t)^2]$$

Shown for a nonterminal transition. At termination, the target is just the immediate reward.

Hafner et al., Dreamer (2020)

Two routes to a policy gradient

Score-function update

No derivatives through dynamics

A stochastic policy supplies an action distribution.

$$a_t \sim \pi_\theta(\cdot \mid o_t, g_t)$$

$$\mathbb{E}[\nabla_\theta \log \pi_\theta(a_t \mid o_t, g_t)\, \hat{A}_t]$$

Â compares return with a baseline. Increase the probability of better actions.

Differentiate the prediction

Uses derivatives through dynamics

A differentiable policy produces actions inside a differentiable model.

$$a_t = \pi_\theta(o_t, g_t)$$

$$\nabla_\theta \hat{J}(\theta)$$

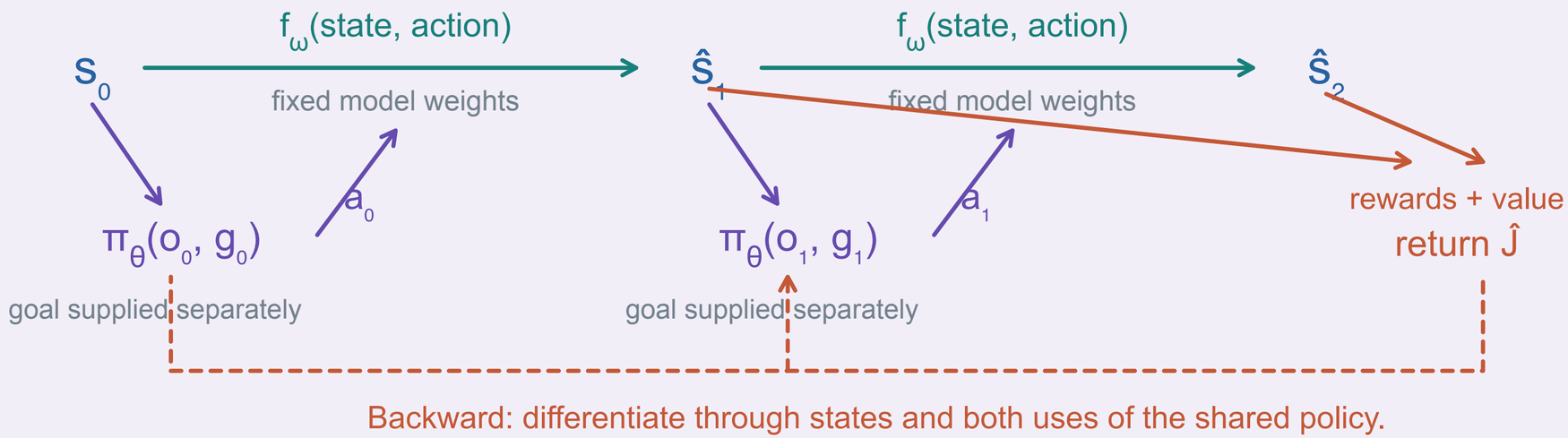
Ĵ is predicted return. Trace its dependence on policy parameters.

Both routes can sample rollouts. The distinction is how the update uses the dynamics.

Hafner et al., Dreamer (2020) · SuperMimic, §3–4

How future return reaches the policy

Forward: predict the outcome of repeated policy decisions.



$$\hat{J}(\theta) = \mathbb{E}[\sum_{k=0}^{H-1} \gamma^k \hat{r}_{k+1} + \gamma^H V_\psi(\hat{s}_H, g_H)]$$

Fix model and value weights. Differentiate through their inputs. The same θ appears at every policy call.

Hafner et al., Dreamer (2020)

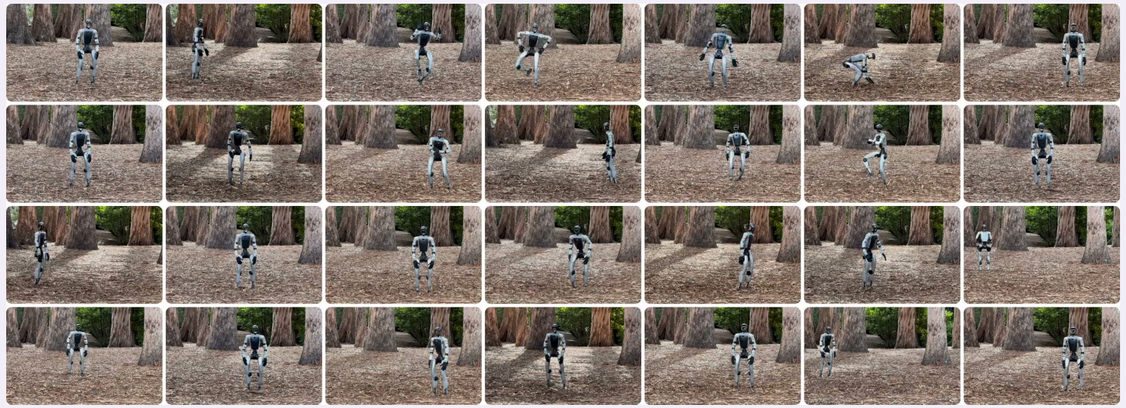
Planning updates actions; learning updates a policy

	Action planning	Policy learning
Decision variable	$A = (a_0, \ldots, a_{H-1})$	θ in π_θ
Prediction starts from	The current estimate	A distribution of training states
Optimization result	A candidate command sequence	A reusable action-selection function
World-model weights	Fixed during action search	Fixed during each actor update

Fitting the world model is a separate update to ω.

Tedrake, Trajectory Optimization · Hafner et al., Dreamer (2020)

From imagined training to robot control



Real Unitree G1 demonstrations from the SuperMimic project.

During training

Model predictions improve the actor.

On the robot

The actor reads proprioception and the motion goal.

Optional online model

A separate design choice: combine the actor with planning.

A learned world model can be central to training and absent from the deployed controller.

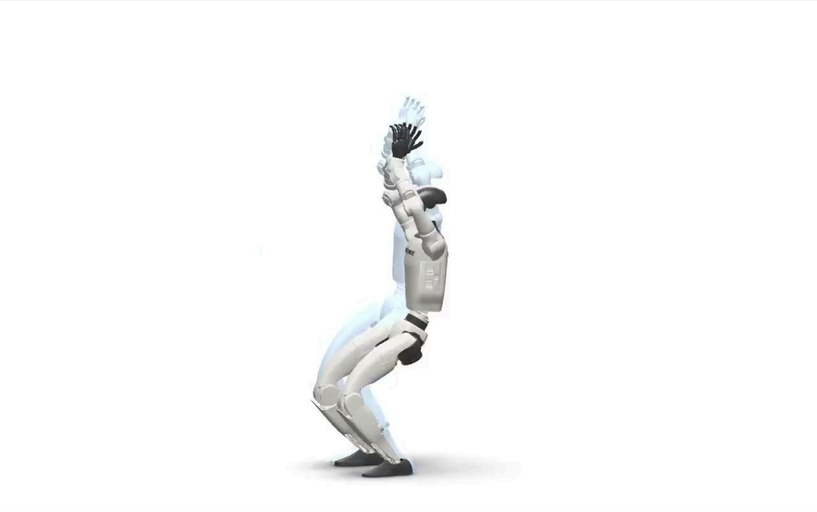
SuperMimic



A world-model-trained policy on a physical Unitree G1.

A complete example of learning robot control through predicted dynamics.

Motion tracking



SuperMimic tracking a reference motion in simulation.

Pale blue: reference motion. Gray / white: simulated robot.

Reference

A time-varying desired body configuration.

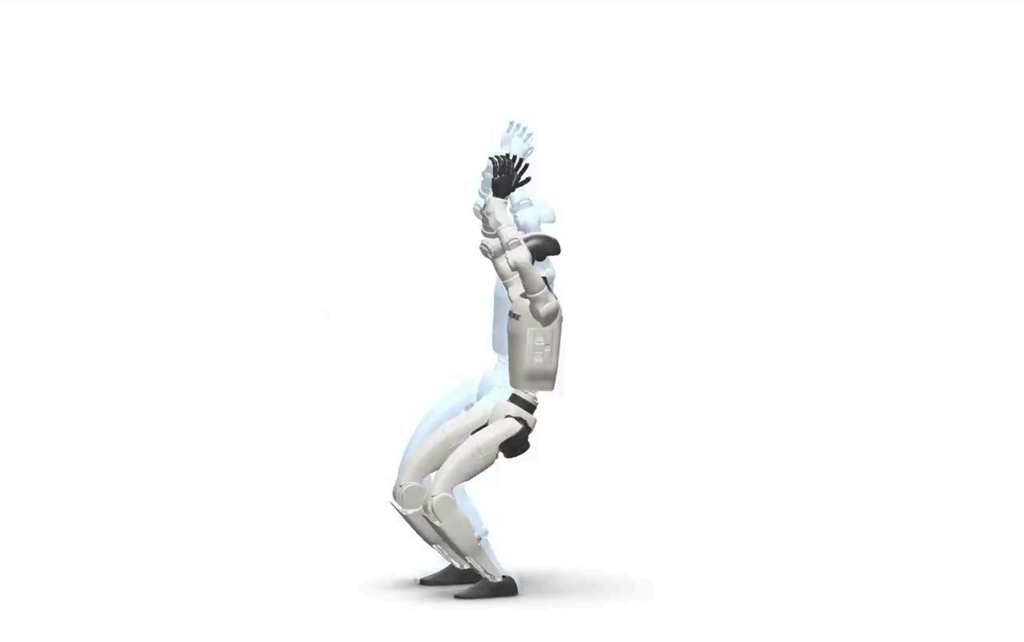
Robot

The body realized under the controller.

Tracking reward

Compare pose and motion at the same time.

A reference motion is not a motor command



Reference and simulated robot at one instant.

$$a_t = \pi_\theta(o_t, g_t)$$

Goal g_t

Desired body motion from the reference.

Observation o_t

Current joint and body measurements.

Command a_t

Normalized joint setpoints.

The actor compensates for the difference between desired and actual motion.

Many behaviors supply one dynamics dataset



A shared controller tracks many motions in simulation. Official project visualization.

$$(s_t, a_t, s_{t+1}, \text{physical context}) \quad \text{from many behaviors}$$

The task changes which transitions are collected; the dynamics model learns their physical consequences.

Why add a learned model to the simulator?



Model rollouts provide predicted robot states.

Fast rollout evaluation

Batch many predicted transitions on the GPU.

Differentiable dynamics

Use sensitivities of predicted motion to improve the actor.

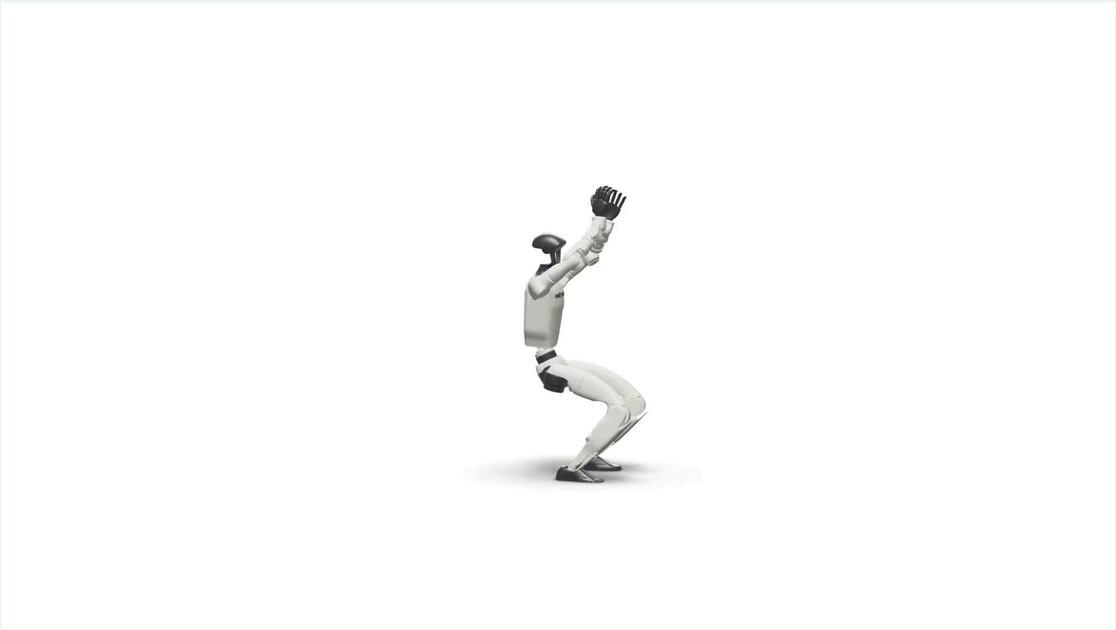
Simulator evidence

Keep the simulator as the source of recorded transition targets.

The learned model is a computational surrogate for policy training.

SuperMimic, §3–4

Collecting transitions in the simulator



The policy acts in the simulator during data collection.

Apply the actual command

Include any exploration noise in the recorded action.

Record the successor

The simulator supplies the next state.

Record physical context

Preserve the settings used for that transition.

$$\mathcal{D} = \{(s_t, a_t, s_{t+1}, \xi)\}$$

ξ denotes physical context, such as friction and actuator settings.

Fitting predictions to simulator evidence



Blue predicted poses are compared with recorded simulator outcomes during model fitting.

$$\hat{s}_{k+1} = f_\omega(\hat{s}_k, a_k, \xi)$$

Start

Use a recorded initial state.

Replay

Supply the recorded action sequence.

Fit ω

Compare predictions with recorded states at matching times.

The image illustrates predicted motion; the training targets come from the recorded trajectory.

Starting imagination from recorded states



The current policy generates a new predicted continuation.

Keep the seed

A recorded starting state and physical context.

Change the commands

Let the current policy choose new actions.

Evaluate the result

Compute observations, rewards and a terminal value.

Model fitting follows recorded actions. Policy improvement evaluates new actions.

The policy acts on predicted observations



A simulator trajectory with a sequence of world-model predictions.

Gray / white: simulator Blue: world-model predictions

Blue silhouettes show predicted poses at successive times.

Observe

Construct sensor-like inputs from the predicted state.

Act

Combine these inputs with the reference goal.

Predict

Advance the state under the policy’s new command.

The robot dynamics supplies the actor gradient



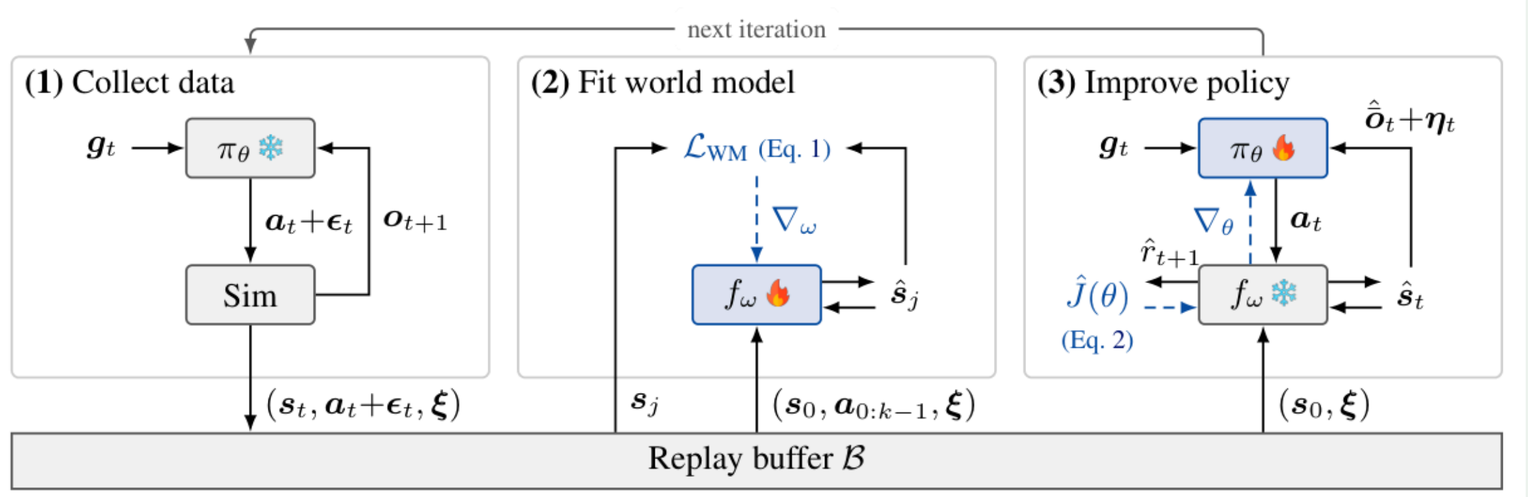
Predicted body motion provides the middle sensitivity in the actor update.

$$\frac{\partial \hat{J}}{\partial \hat{s}} \qquad \underbrace{\frac{\partial \hat{s}}{\partial a}}_{\text{learned robot dynamics}} \qquad \frac{\partial a}{\partial \theta}$$

One derivative path: command changes body motion; body motion changes return.

Model weights stay fixed during this actor update. The simulator is outside this derivative path.

Putting the training stages together



Original SuperMimic training figure: simulator collection, model fitting, policy improvement.

ε: action noise. η: sensor noise. ξ: physical context. Snowflake: fixed weights in that update.

The improved policy collects the next round of simulator experience.

What information reaches the deployed policy?



The deployed actor controls the physical Unitree G1.

Proprioception: measurements of the robot's own joints and body.

Available to the actor

Joint motion, inertial signals, observation history and the goal.

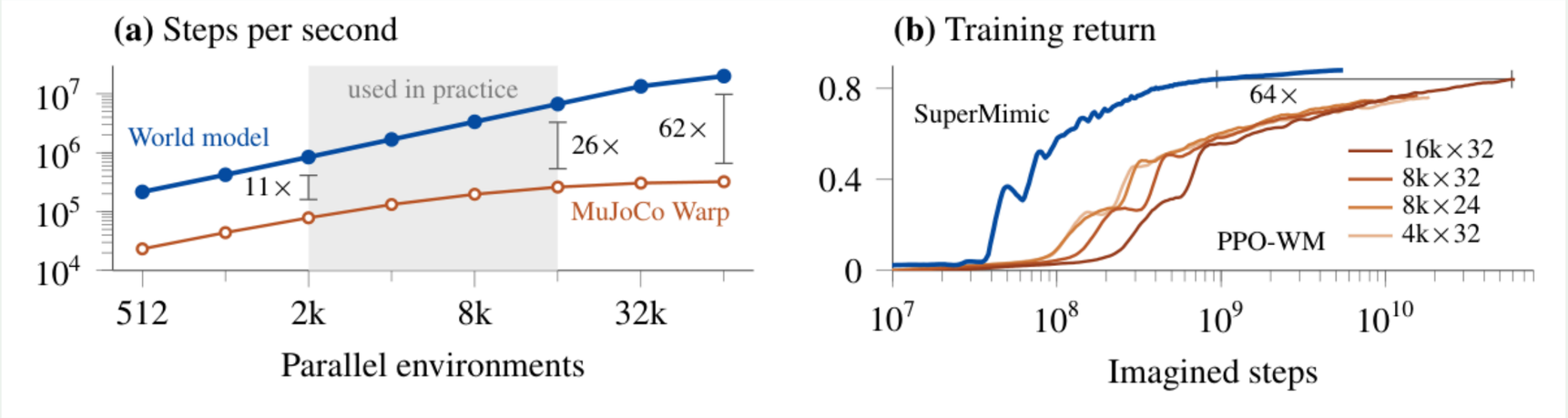
Available during training

The model and critic may also use simulator state and physical context.

Produced by the actor

Normalized joint setpoint commands.

Two benefits, measured separately



Left: transitions per second. Right: imagined transitions needed for learning.

Rollout throughput

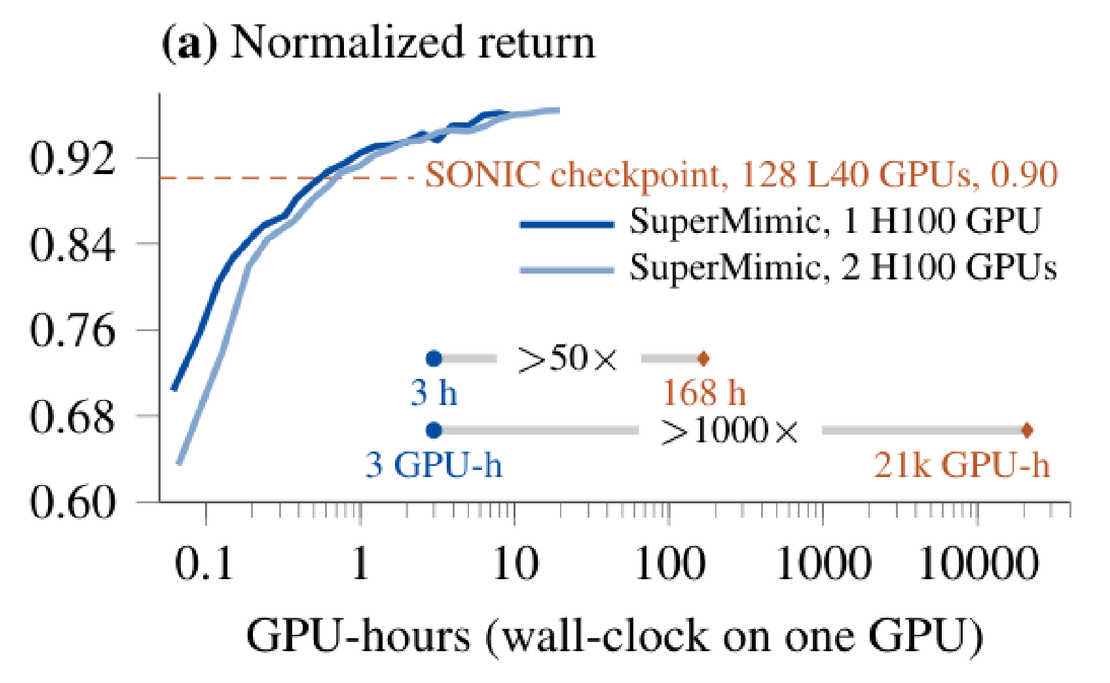
World model versus MuJoCo Warp on an H100.
One step = one control interval.

Policy-update efficiency

Blue: gradients through the model.
Orange PPO-WM: PPO using model rollouts.

Right-panel legend = parallel rollouts × steps per rollout. The 64× annotation concerns imagined steps to comparable return.

Reading the return and compute comparison



Original return-versus-compute comparison.

Baseline

SONIC is a PPO-based humanoid motion tracker.

Vertical axis

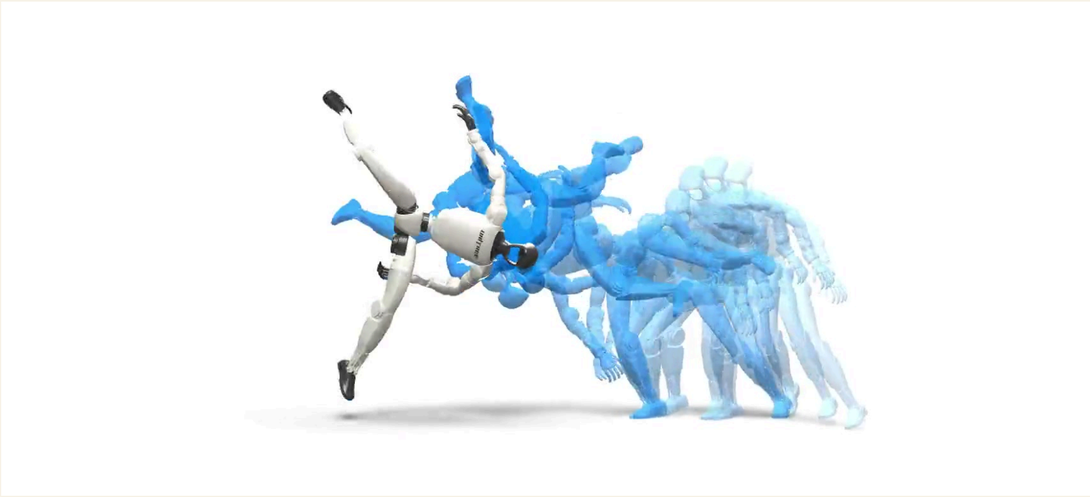
Episode reward divided by its maximum possible total.

Horizontal axis

GPU-hours = GPU count × elapsed hours. The plotted axis aggregates device time.

The compared runs use different GPU hardware and training data. The plot does not establish performance on arbitrary new motions.

Two gaps separate prediction from hardware

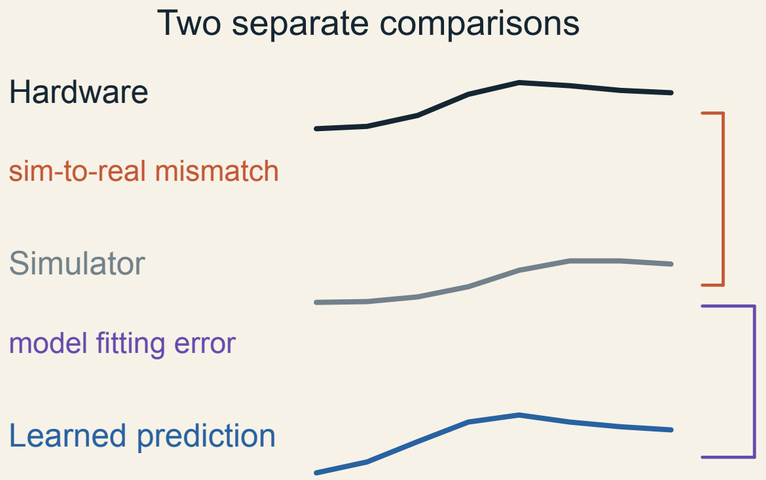


Simulator and model under a supplied slippery-floor condition.

Gray / white: simulator Blue: world-model predictions

Blue silhouettes show predicted poses at successive times.

Friction is supplied to this prediction. This clip compares a model with its simulator.

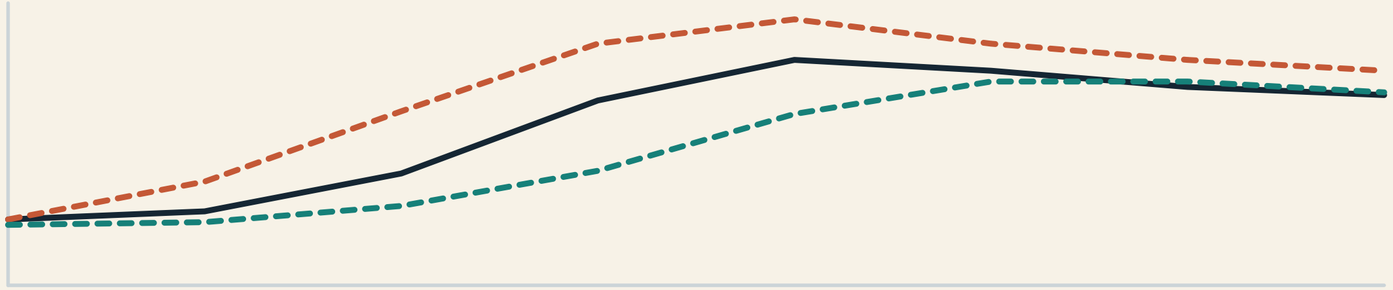


Schematic responses under the same command.

Identifying physical parameters

Replay the same recorded commands

adjust physical parameters to match the measured response



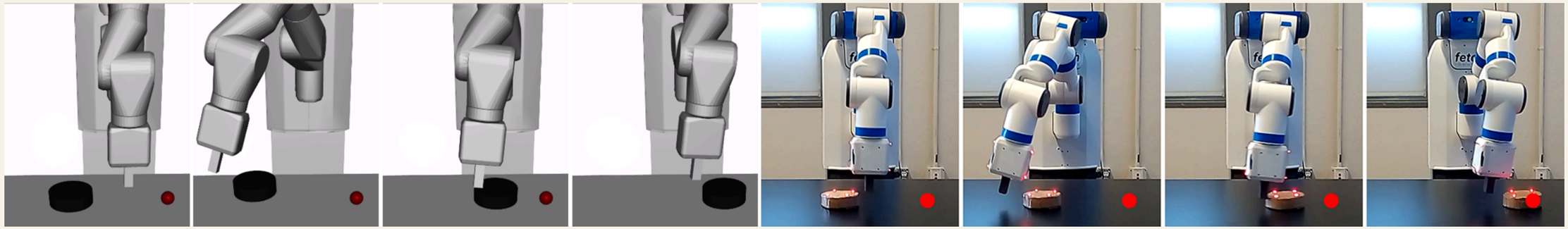
Schematic illustration; validate on additional recorded trajectories.

Possible parameters Mass, inertia, friction, actuator response, and delay.

Data requirement The recorded motions must reveal the parameters we want to estimate.

Hauser, Robotic Systems · Tedrake, Trajectory Optimization

Domain randomization



Dynamics randomization: train in simulation (left), execute the learned policy on a real arm (right).

Randomize	What changes in the response
Mass, inertia and friction	Acceleration, balance, sticking and sliding
Actuation and sensing	Motor response, command delay, calibration and noise

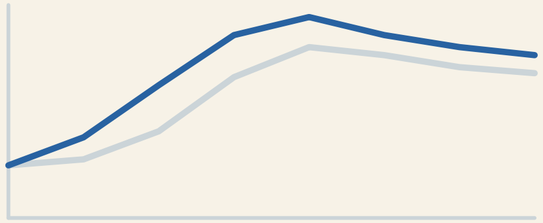
Train one policy across plausible variations of the physical system.

Training across randomized dynamics

One shared policy encounters different physical responses.

Mass / inertia

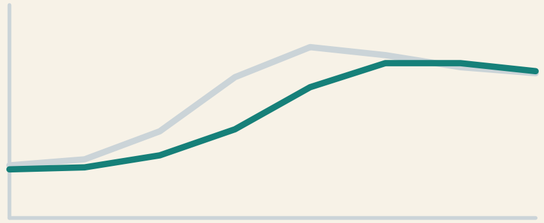
schematic response over time



changes acceleration

Friction / contact

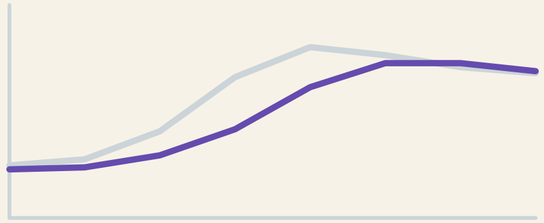
schematic response over time



changes sticking and sliding

Actuator delay

schematic response over time



changes when motion responds

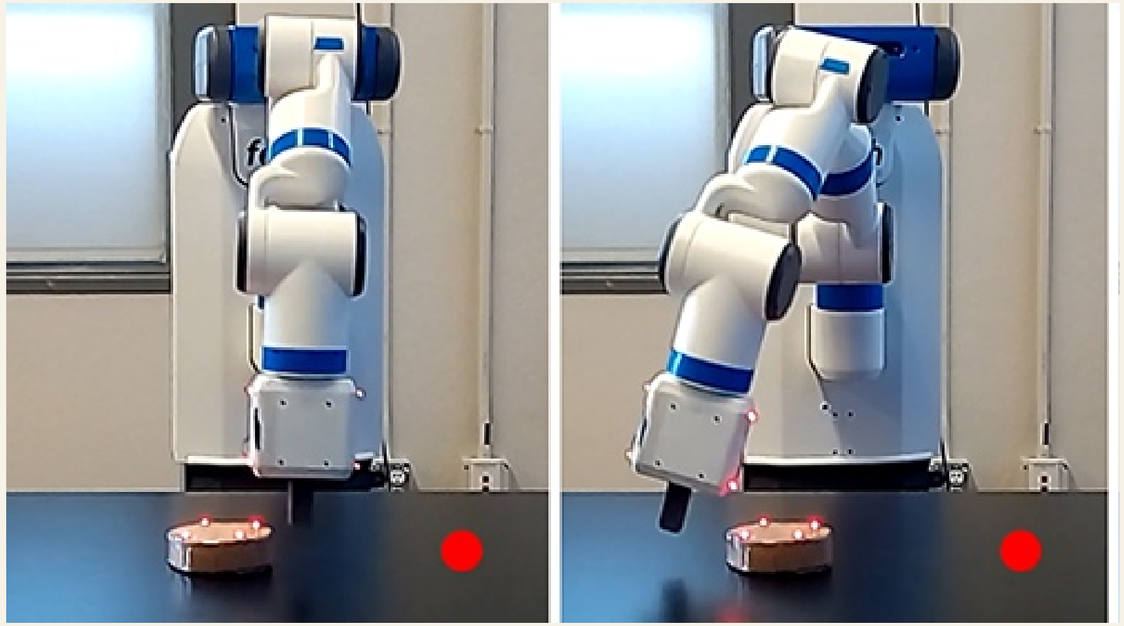
$$\xi \sim p_{\text{train}}(\xi), \qquad \max_{\theta} \ \mathbb{E}_{\xi}[J(\theta;\xi)]$$

ξ: physical conditions. p_train: their training distribution. J: return in those conditions.

Sample physical conditions, collect rollouts, and update the same policy θ.

Peng et al., Dynamics Randomization (2018) · SuperMimic, §3–4

The real robot does not know its exact parameters



The deployed controller must work with information available on the real robot.

Known in simulation

The simulator knows the sampled mass, friction and delay.

Hidden at deployment

The actor usually receives observations, not exact parameter values.

Use feedback or history

Observed responses can reveal how the current system behaves.

Robustness across conditions and adaptation from experience are different capabilities.

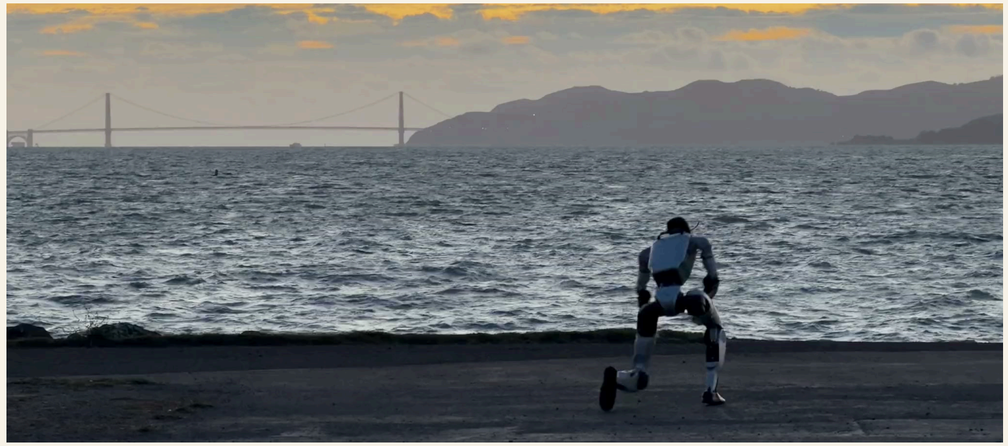
Calibration and randomization work together

	System identification	Domain randomization
Question	Which dynamics fit the real system?	Which variations should the policy tolerate?
Evidence / setup	Recorded real commands and responses	A chosen range of simulator parameters
Outcome	A fitted model or parameter estimate	A policy trained across possible dynamics

Calibrate the center of the range; randomize the remaining uncertainty.

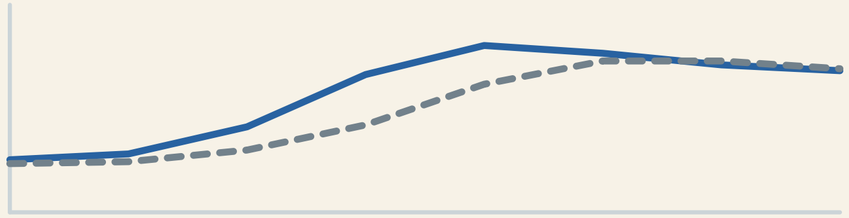
Check unseen parameter combinations and real trajectories. A wide training range is not a guarantee.

Closing the loop on the real robot



Evaluate the actual response

sensor feedback closes the loop



Measured motion versus intended behavior

Prediction quality	Control quality
Error on relevant transitions	Task completion and recovery
Behavior under new conditions	Command feasibility and control latency

Two uses of world models

	For planning	For learning policies
Optimize	A current action sequence	A reusable policy
Model provides	Predicted candidate outcomes	Experience or policy gradients
Typical deployment	Online prediction and search	Policy inference, optionally planning
Connection	Policies can propose actions	Values can extend a planning horizon

Next: controlling robots from visual and language inputs with action-conditioned models.