

PHYSICAL AI ESSENTIALS

A Practical Book for Engineering
the Next AI Era



Dr. Barak Or, Ph.D.



Physical AI Essentials

A Practical Book for Engineering the Next AI Era

Copyright © 2026 ArtificialGate Ltd.

Haifa, Israel.

All rights reserved.

Published by ArtificialGate Ltd.



*This book is dedicated to
Alma and Aime.*



Introduction

Welcome.

The phrase “Physical AI” is now used for almost everything that joins a learned model to a machine. I use it more narrowly. AI becomes physical when its output enters a causal loop with the world: a prediction informs an action, the action changes the scene, and the changed scene becomes the model’s next input. At that point a delayed camera exposure, a slipping wheel, a compliant gripper, or a person stepping into an aisle is no longer an implementation detail. It is part of the intelligence problem.

My own route into this subject came through aerospace engineering, inertial navigation, deep learning, and autonomous systems. Those fields teach the same lesson in different ways: a strong model cannot rescue a weak physical contract. An unknown coordinate frame can dominate a sophisticated network; a stale state estimate can invalidate an otherwise sensible action; a beautiful predicted video can still violate the dynamics of the platform that is meant to realize it.

I wrote this book for practicing engineers, research engineers, and graduate students who want to build the complete loop. The order is intentional. We begin with implementation, develop the mathematics needed to understand it, and then build the systems argument required for deployment. In practice these views reinforce one another: implementation reveals which mathematics matters, mathematics makes failure legible, and systems engineering determines what may reach the machine.

What the book enables

By the end, a reader should be able to:

- ▶ distinguish a vision-language model (VLM), vision-language-action model (VLA), world model, world foundation model, world-action model (WAM), joint-embedding predictive architecture (JEPA), robot foundation model, simulator, and digital twin;

- ▶ prepare robot-learning datasets and train or retrain policy families with PyTorch, TensorFlow, Hugging Face, and LeRobot workflows;
- ▶ derive the principal learning objectives used for contrastive perception, behavioral cloning, latent dynamics, diffusion, flow matching, and predictive representation learning;
- ▶ design evaluation that measures closed-loop task behavior instead of reporting only offline loss;
- ▶ identify Physical AI hallucinations across perception, grounding, affordance, dynamics, action, time, and safety;
- ▶ place learned models inside a monitored system with uncertainty estimation, constraint checks, fallback behavior, observability, and release gates.

Module	Engineering question
I. Foundations	What exactly is Physical AI, and how do states, sensors, actions, data, geometry, dynamics, and demonstrations fit together?
II. Perception to action	How do VLMs become VLAs, and how are robot foundation policies trained and adapted with modern open tooling?
III. Predictive intelligence	How do world models, JEPA systems, WAMs, and planners represent and choose possible futures?
IV. Reliable systems	How do simulation, hallucination analysis, runtime authorization, and deployment engineering keep learned action useful and bounded?

Each module ends with a companion Google Colab activity. The complete exercise brief and the solution are intentionally separate from the book: the bundle contains one learner notebook and one solution notebook per module. The book gives the objective, interfaces, expected output, compute budget, and software requirements so the laboratory work remains part of a coherent course.

Where this book sits

No responsible Physical AI text should pretend to replace the books that established robotics, estimation, control, deep learning, or reinforcement learning. *Modern Robotics* remains the better place to study screw theory in depth [1]; *Probabilistic Robotics* is still a foundational treatment of belief and localization [2]; *Modern Deep Learning Foundations* offers an engineering-first preparation path from fundamentals to deployment [3]; and the comprehensive deep-learning and reinforcement-learning texts deserve to be read on their own terms [4, 5].

The missing piece is the bridge between those shelves and the model stack now arriving in laboratories: VLMs, VLAs, world models, JEPA systems, WAMs, open



robot-learning tools, and runtime checks for physical hallucinations. I have written this book around that bridge.

How to read it

Do not read every page at the same speed. If you are building your first robot-learning system, follow the lessons in order and run the four laboratories. If you already work in robotics, begin with Lesson 4 and return to the data material when an interface becomes ambiguous. If your background is deep learning, resist the temptation to skip Lessons 2 and 3: frames, timestamps, action semantics, and closed-loop shift are where otherwise impressive models usually become fragile.

Prerequisites and compact bridge

The intended reader can program in Python and has working knowledge of linear algebra, probability, optimization, and deep learning. Prior control theory and robotics are useful but not required. This bridge fixes the notation used in the book and points underprepared readers toward the minimum concepts to review. Readers wanting an engineering-first deep-learning bridge should use *Modern Deep Learning Foundations* [3]; readers needing a comprehensive theoretical treatment should use a dedicated deep-learning reference [4], a probabilistic machine-learning reference [6], and a robotics mechanics reference [1] in parallel.

Linear algebra

A vector $\mathbf{x} \in \mathbb{R}^n$ is an ordered point in an n -dimensional space; a matrix $A \in \mathbb{R}^{m \times n}$ maps it to $A\mathbf{x} \in \mathbb{R}^m$. The Euclidean norm, inner product, and quadratic form are

$$\|\mathbf{x}\|_2 = \sqrt{\sum_{i=1}^n x_i^2}, \quad \langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^\top \mathbf{y}, \quad q(\mathbf{x}) = \mathbf{x}^\top Q \mathbf{x}. \quad (1)$$

Terms: x_i is component i ; $Q \geq 0$ is positive semidefinite when $q(\mathbf{x}) \geq 0$ for every $\mathbf{x}$.

The singular value decomposition $A = U\Sigma V^\top$ separates orthonormal directions from nonnegative gains. It explains low-rank adaptation, conditioning, latent compression, and observability. Review eigenvalues, projections, Jacobians, and matrix calculus if these are unfamiliar.

Probability and estimation

A random variable X is described by a density or mass function $p(x)$. Conditional probability and Bayes' rule are

$$p(x | y) = \frac{p(y | x)p(x)}{p(y)}, \quad p(y) = \int p(y | x)p(x) dx. \quad (2)$$

Terms: $p(x)$ is the prior; $p(y | x)$ the likelihood; $p(x | y)$ the posterior; $p(y)$ the evidence.

Expectation is $\mathbb{E}[f(X)] = \int f(x)p(x)dx$. For vector $\mathbf{x}$, mean $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ and covariance $\Sigma = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top]$ measure location and coupled spread. Physical AI repeatedly estimates a hidden state from incomplete, delayed, and noisy observations. Readers who need the complete probability bridge can consult Appendix A, which covers conditioning, Gaussian inference, likelihood, information measures, Monte Carlo estimation, stochastic processes, calibration, and risk. Appendix B provides the full symbol reference.

Optimization and learning

Training usually solves empirical risk minimization:

$$\boldsymbol{\theta}^* = \arg \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\mathbf{x}_i), \mathbf{y}_i) + \lambda \Omega(\boldsymbol{\theta}). \quad (3)$$

Terms: N is the sample count; $f_{\boldsymbol{\theta}}$ a parameterized model; ℓ the data loss; Ω a regularizer; $\lambda \geq 0$ its weight; $\boldsymbol{\theta}^*$ an optimizer.

Gradient descent uses $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_k)$, where α_k is a learning rate. Readers should understand backpropagation, minibatches, validation, regularization, normalization, attention, and transfer learning. PyTorch examples dominate current robot-learning infrastructure; TensorFlow equivalents are supplied where they clarify framework-independent design.

The robotics bridge

Four concepts are enough to begin:

- ▶ **State** $\mathbf{x}_t$: variables sufficient for the chosen dynamical model at time t .
- ▶ **Observation** $\mathbf{o}_t$: sensor information available to the agent; it need not be the state.
- ▶ **Action** $\mathbf{a}_t$: a command represented at the learning interface, such as joint velocity, an incremental Cartesian end-effector pose command in a declared frame, torque, or a discrete skill.
- ▶ **Policy** $\pi_{\boldsymbol{\theta}}(\mathbf{a}_t | \mathbf{o}_{\leq t}, \mathbf{g})$: a distribution over actions conditioned on observation history and goal $\mathbf{g}$.

The simplest controlled dynamical system is

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{o}_t = h(\mathbf{x}_t) + \mathbf{v}_t. \quad (4)$$

Terms: f is the transition matrix; h the observation model; $\mathbf{u}_t$ the physical control input; $\mathbf{w}_t$ process disturbance; $\mathbf{v}_t$ measurement noise. The learned action $\mathbf{a}_t$ may be translated into $\mathbf{u}_t$ by a controller.

This distinction prevents a common error: treating the token predicted by a VLA as though it were already a safe actuator command.

Recommended external preparation

Underprepared readers should use Or's *Modern Deep Learning Foundations* for engineering-oriented preparation from fundamentals through training and deployment; focused portions of Goodfellow, Bengio, and Courville for deeper theoretical coverage; Sutton and Barto for reinforcement learning; Lynch and Park for robot motion; Thrun, Burgard, and Fox or Barfoot for estimation; and the official PyTorch, TensorFlow, Hugging Face, and LeRobot tutorials for software fluency. Full mastery is not required before Lesson 1. The objective is to recognize the notation, implement differentiable models, and reason about uncertainty and time.

LEARNING OBJECTIVES

You are ready when you can trace tensor shapes through a neural network, derive a gradient for mean-squared error, interpret a conditional probability, explain why observation is not state, and identify the units and coordinate frame of an action.



Reader guide

How claims are marked

This edition separates four kinds of statement. [F] denotes a mathematical or physical foundation; [R] a result supported by a cited research source; [E] an engineering recommendation derived from practice; and [H] a frontier hypothesis that still needs broader validation. The tags prevent a plausible design pattern from being mistaken for an established theorem.

Four landmark results

Four navy LANDMARK RESULT boxes isolate the theoretical results that carry most directly into Physical AI engineering: belief-state sufficiency under partial observability, the finite-horizon simulation bound for learned dynamics, split-conformal marginal coverage, and control-barrier forward invariance. Each box states assumptions, mathematics, engineering meaning, and the boundary of the guarantee. These are established foundations applied to Physical AI, not claims that the book invented new universal laws.

Three passes through a lesson

1. **Implementation pass:** read the interface contracts, worked exercise, diagrams, and release checklist.
2. **Mathematics pass:** reconstruct each objective from the defined symbols and check units, frames, and conditioning variables.
3. **Assurance pass:** identify the distribution boundary, observable failure signal, authorization rule, and fallback.



Worked studies

Numbered worked studies are distributed throughout the book and placed immediately after the mathematics, model, or system decision they test. Every lesson contains at least one; the data, hallucination, authorization, and systems lessons contain additional studies where engineering judgment benefits from explicit numbers. The assumptions are stated, the calculation is shown, and the conclusion identifies what the result does *not* prove. These studies complement, but do not replace, the four module-level Colab laboratories.



Contents

Introduction	iii
Prerequisites and compact bridge	vi
Reader guide	ix
I Foundations of Physical Intelligence	1
1 The Physical AI Stack	2
2 The Physical Interface	13
3 Data and Learning from Demonstrations	24
II From Perception to Action	34
4 VLMs for Grounded Perception	35
5 VLAs and Robot Foundation Models	44
6 Training and Retraining with Hugging Face and LeRobot	54



III	Predictive Physical Intelligence	64
7	World Models: Learning Possible Futures	65
8	JEPA and Predictive Representations	75
9	World-Action Models and Planning	84
IV	Reliable Physical AI Systems	93
10	Simulation, Digital Twins, and Sim-to-Real	94
11	Physical AI Hallucinations	104
12	From Prediction to Authorization	117
13	Physical AI Systems Engineering	130
	Closing	141
A	Mathematical Foundations	142
B	Notation and Terminology	160
C	Engineering Reference	179
D	Literature Landscape	189
	About the author	204

MODULE I

Foundations of Physical
Intelligence



LESSON 1

The Physical AI Stack

LEARNING OBJECTIVES

After this lesson you can define Physical AI operationally; distinguish VLM, VLA, world model, world foundation model, WAM, JEPA, robot foundation model, simulator, and digital twin; map these model classes onto a real-time system; and choose evaluation levels that expose physical rather than merely linguistic competence.

1.1 From content generation to causal intervention

A language model changes a string. A Physical AI system can change the world. This distinction sounds obvious, but it alters the entire engineering objective. Prediction error is no longer only a statistical quantity. A prediction may become an action, an action changes the next observation, and the next observation changes every later decision. The model therefore operates inside a closed causal loop:

$$\mathbf{o}_t \xrightarrow{\text{encode}} \mathbf{z}_t \xrightarrow{\text{infer}} \mathbf{a}_{t:t+H-1} \xrightarrow{\text{control}} \mathbf{u}_t \xrightarrow{\text{world}} \mathbf{x}_{t+1} \xrightarrow{\text{sense}} \mathbf{o}_{t+1}. \quad (1.1)$$

Terms: t is the decision index; $\mathbf{o}_t$ an observation; $\mathbf{z}_t$ a learned representation; $\mathbf{a}_{t:t+H-1}$ an action chunk of horizon H ; $\mathbf{u}_t$ the executed control input; $\mathbf{x}_{t+1}$ the next physical state.

The environment is not a passive dataset. Its motion, contact, people, illumination, friction, network delay, and actuator response determine the distribution of future inputs. An error at time t can therefore move the system into a state absent from training. Physical AI engineering is the design of the complete loop, including the learned model, coordinate transforms, low-level controller, monitor, fallback, logging, and recovery behavior.

DEFINITION - Physical AI

Physical AI is artificial intelligence coupled to the physical world through sensors and actuators, such that learned perception, prediction, reasoning, or policy outputs influence physical evolution under embodiment, time, uncertainty, and constraints.

This definition includes manipulators, mobile robots, autonomous vehicles, drones, industrial cells, medical devices, and interactive agents. It excludes a VLM used only to caption an image, even if that image came from a robot. The same VLM becomes part of Physical AI when its output affects planning or action.

1.2 The terms that must not be collapsed

The frontier is untidy: names overlap, papers extend familiar labels, and product language often outruns mathematical precision. Throughout this book I use a deliberately strict test. Ask *what conditional distribution is learned?* and then ask *what role does the output play in the loop?* If those two answers differ, the models belong in different categories even when their backbones look similar.

1.2.1 Vision-language model: perception and semantics

DEFINITION - VLM

A **vision-language model** maps visual input and language context to a semantic representation or language output. A generic form is

$$p_{\theta}(\mathbf{y} \mid \mathbf{I}, \mathbf{q}), \quad (1.2)$$

where $\mathbf{I}$ is one or more images, $\mathbf{q}$ a text prompt, $\mathbf{y}$ a textual or structured answer, and θ the learned parameters. A VLM understands or describes; it does not inherently define an executable action interface.

A VLM can identify a cup, answer where it is, compare objects, or propose a high-level plan. It may still fail at metric depth, occlusion, state change, or affordance. It becomes an action component only when another module grounds its output in the robot's state and command space.

1.2.2 Vision-language-action model: a policy interface

DEFINITION - VLA

A **vision-language-action model** is a learned policy that conditions robot action on visual observation and language:

$$\pi_{\theta}(\mathbf{a}_{t:t+H-1} \mid \mathbf{I}_{\leq t}, \mathbf{q}, \mathbf{r}_{\leq t}). \quad (1.3)$$

Here $\mathbf{r}_{\leq t}$ denotes optional robot state or proprioceptive history. The output can be discrete action tokens, continuous controls, trajectories, action chunks, or parameters of an action distribution.

Adding action output changes the engineering contract. Every output dimension must be tied to a specified embodiment, update frequency, coordinate frame, unit, normalization rule, prediction horizon, and downstream controller. “Move left” is language. $[-0.012, 0, 0, 0, 0, 0, 0]$ becomes an action only after the seven dimensions, their frame, their timing, and their admissible bounds are defined.

1.2.3 World model: consequence prediction

DEFINITION - World model

A **world model** learns how a relevant environment state or latent representation evolves, optionally under actions:

$$p_{\theta}(\mathbf{z}_{t+1:t+H} \mid \mathbf{z}_{\leq t}, \mathbf{a}_{t:t+H-1}, \mathbf{c}). \quad (1.4)$$

$\mathbf{z}_t$ is a state or latent state, $\mathbf{a}_t$ an action, and $\mathbf{c}$ context. It forecasts consequences; by itself it need not select reward-maximizing action.

World models may predict pixels, tokens, 3-D occupancy, object states, proprioception, rewards, termination, or compressed latent features. A forward dynamics network for one robot is a world model at narrow scope. Generative video models can be world models when time and action conditioning support physical prediction rather than visual continuation alone.

DEFINITION - World foundation model

A **world foundation model** is a large, reusable predictive model trained across broad environments, tasks, or embodiments and adapted to multiple downstream physical prediction or control problems. Foundation-scale reuse, rather than a particular architecture, distinguishes it from a task-specific world model.

1.2.4 WAM: futures and actions in one model family

DEFINITION - World-action model

A **world-action model (WAM)** models future world evolution and action within a unified generative or predictive formulation. One useful factorization is

$$p_{\theta}(\mathbf{z}_{t+1:t+H}, \mathbf{a}_{t:t+H-1} \mid \mathbf{z}_{\leq t}, \mathbf{q}). \quad (1.5)$$

The joint distribution can be implemented jointly or as coupled forward and inverse/action factors. A WAM is distinguished from a conventional VLA by explicit future-world modeling and from a conventional world model by explicit action generation.

The name is new and not perfectly standardized. A VLA with an auxiliary video loss is not automatically a WAM; future prediction must be structurally available for learning, selection, diagnosis, or planning.

1.2.5 JEPA: predict representations, not necessarily pixels

DEFINITION - JEPA

A **joint-embedding predictive architecture** learns a predictor that maps the representation of observed context to the representation of a target. In simplified form,

$$\mathcal{L}_{\text{JEPA}} = \|g_{\theta}(f_{\theta}(\mathbf{x}_{\text{ctx}}), \mathbf{m}) - \text{sg}(f_{\xi}(\mathbf{x}_{\text{tgt}}))\|_2^2. \quad (1.6)$$

f_{θ} is a context encoder, f_{ξ} a target encoder, g_{θ} a predictor, $\mathbf{m}$ target-location or mask information, and sg stop-gradient. A JEPA predicts abstract target embeddings and can discard pixel detail irrelevant to structure or action.

JEPA is an architectural principle, not a synonym for world model. Image JEPA predicts missing-region representations in a static image. Video JEPA predicts spa-

tiotemporal representations. An action-conditioned video JEPA can serve as a latent world model.

1.2.6 Robot foundation model

DEFINITION - Robot foundation model

A **robot foundation model** is pretrained on sufficiently diverse robot, multi-modal, or web data to support reuse across multiple tasks, environments, or embodiments after prompting, fine-tuning, or lightweight adaptation. It may be a VLA, a perception model, a WAM, or a component of a hierarchical controller.

“Foundation” describes transfer scope, not safety, autonomy, or general intelligence. A large model that works only on one fixed cell after full retraining is not functionally foundational for the new domain.

1.2.7 Simulator and digital twin

DEFINITION - Simulator versus digital twin

A **simulator** executes an encoded approximation of physical rules and environment behavior. A **digital twin** is a maintained virtual representation tied to a specific real asset, process, or system through configuration and data. A learned world model infers dynamics from data; a simulator encodes them; a digital twin is defined by correspondence to a particular physical counterpart. Hybrid systems combine all three.

1.3 Comparison table

Class	Conditions on	Produces	Not guaranteed
VLM	vision, text	text or semantic features	executable or physically grounded action
VLA	vision, text, robot context	robot action distribution	explicit future-world prediction or safety
World model	state/history, often action	possible future state	action selection
World foundation model	broad world data	reusable future model	embodiment-specific accuracy
WAM	context, goal	future-state/action joint model	runtime admissibility
JEPA	context representation, mask/action	target representation	pixel reconstruction or policy
Robot foundation model	broad robot/multimodal data	reusable robot capability	universal embodiment transfer
Simulator	encoded model, action	computed trajectory/sensors	match to a specific real asset
Digital twin	live/specific asset data	synchronized virtual asset	learned generalization

Table 1.1: *Model class should be assigned by interface and objective.*

1.4 Physical AI hallucinations

Language hallucination is often defined as generated content unsupported by evidence. Physical action requires a broader concept.

DEFINITION - Physical AI hallucination

A **Physical AI hallucination** is a model-generated perception, prediction, plan, affordance, or action whose represented premise or implied physical consequence is unsupported by the available evidence, task specification, embodiment, or admissible dynamics. It can be semantically plausible while being spatially, temporally, kinematically, dynamically, or operationally false.

Examples include reporting a graspable handle that is occluded and absent; choosing a grasp pose beyond reach; predicting that an object will pass through another; issuing a joint increment in the wrong robot convention; continuing a stale action after the scene changes; or producing a smooth trajectory that violates acceleration or contact limits. Lesson 11 develops a full evidence-based taxonomy. The essential point is early: fluent explanation and visually plausible futures are weak evidence of physical validity.

1.5 The complete engineering stack

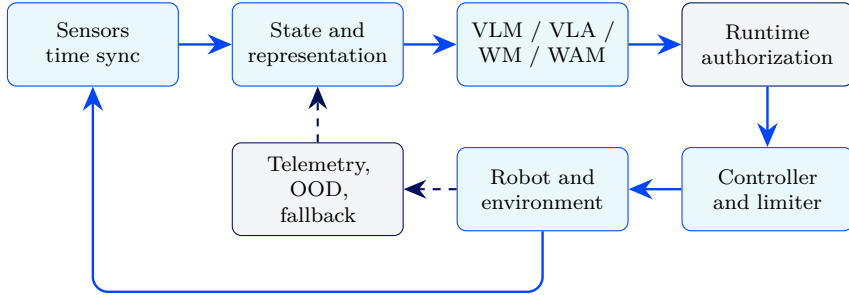


Figure 1.1: A model is one component inside a timed, monitored feedback loop.

The stack has three rates. The sensor rate f_s can be tens to thousands of hertz; the policy rate f_p is often slower; the actuator servo rate f_c can be much faster. If policy inference takes T_{inf} , communication takes T_{net} , monitoring takes T_{mon} , and control preparation takes T_{ctl} , a basic deadline is

$$T_{\text{inf}} + T_{\text{net}} + T_{\text{mon}} + T_{\text{ctl}} \leq \frac{1}{f_p}. \quad (1.7)$$

Terms: All T terms are worst-case or high-quantile latencies at deployment, not average notebook timings; f_p is the required policy update frequency.

When the inequality fails, engineers may reduce input resolution, quantize the model, predict an action chunk, decouple inference from control, or move compute. Every option changes risk. Longer action chunks tolerate inference delay but increase open-loop exposure; asynchronous inference improves throughput but can execute stale intentions unless observation-action freshness is checked.

PHYSICAL FAILURE MODE - Unverified physical handoff

A component-level metric does not validate an interface. Verify schema, frame, units, rate, latency, reset behavior, and action semantics before any model output can influence hardware.





EXERCISE 1.1 - A policy that meets average latency but misses its deadline

A mobile manipulator requires $f_p = 10$ Hz, so its deadline is 100 ms. Use high-quantile budget allowances $T_{\text{inf}} = 61$ ms, $T_{\text{net}} = 12$ ms, $T_{\text{mon}} = 18$ ms, and $T_{\text{ctl}} = 14$ ms. Their sequential sum is 105 ms, so the budget misses the deadline by 5 ms. Here p95 means the 95th percentile: 95 percent of measured cycles are no slower than that value. The p95 of the total path must be computed from synchronized end-to-end samples; a sum of separately estimated component p95 values is only a conservative timing budget, not the exact p95 of the sum. If monitoring and control preparation truly execute in parallel, the budgeted critical path becomes $61 + 12 + \max(18, 14) = 91$ ms, leaving 9 ms of margin. Schedulability is still not task safety. Suppose the platform moves at $v = 0.60$ m/s and verified emergency deceleration is $a = 1.50$ m/s². Using the 91 ms response time,

$$d_{\text{stop}} = vT + \frac{v^2}{2a} = 0.60(0.091) + \frac{0.60^2}{2(1.50)} = 0.1746 \text{ m.}$$

Adding 0.08 m for localization, actuator, and modeling allowances gives a minimum illustrated clearance of 0.255 m. This is one necessary physical check under stated assumptions, not a complete safety certification; contention, stale observations, slope, payload, and surface friction still require validation.

1.6 Levels of success

Physical AI evaluation should progress through four levels:

1. **Representation:** Does the model encode relevant objects, geometry, language, and temporal change?
2. **Offline prediction:** Are held-out actions, states, rewards, or futures predicted under the logged distribution?
3. **Closed-loop task:** Does the policy recover from its own errors and complete tasks across variations?
4. **Operational integrity:** Does the system meet latency, intervention, safety-envelope, reliability, and observability requirements?

An offline action mean-squared error can improve while closed-loop success falls because demonstrations are multimodal: two experts may go around an obstacle on opposite sides, and their mean goes through it. A task-success number can hide excessive collisions or human resets. A safe system can be useless if it refuses every proposal. Report progress and risk together.

A deployment scorecard therefore contains at least success rate, completion time, intervention rate, constraint violation rate, calibration, out-of-distribution coverage, worst-case latency, and behavior after sensor or model faults. Metrics must be stratified by environment, object, instruction, embodiment, and horizon.

PHYSICAL FAILURE MODE - Category error

Do not compare a VLM answer score, a VLA action loss, and a world-model video metric as though they measured the same capability. First specify the interface and the downstream decision that consumes it.



1.7 A 2026 frontier snapshot

The durable categories above contain rapidly changing systems. RT-1 and RT-2 established transformer-based robot policies and transfer from web-scale vision-language knowledge [7, 8]. Open X-Embodiment, Octo, and OpenVLA made cross-embodiment data and generalist policy research more reproducible [9-11]. Flow and diffusion policies such as π_0 and SmolVLA extended continuous action generation [12, 13]. V-JEPA 2 demonstrated action-conditioned predictive representations, while Cosmos and world-action research expanded the world-model interface [14, 15]. In 2026, WAM publications explicitly framed joint future and action modeling as a distinct frontier [16, 17].

1.8 Engineering checklist

Before selecting a model, write down:

- ▶ the observation modalities, timestamps, coordinate frames, and missing-data behavior;
- ▶ the action representation, units, bounds, policy rate, horizon, and controller contract;
- ▶ whether the problem requires semantic understanding, action generation, future prediction, or all three;
- ▶ which uncertainty is stochastic, which is missing state, and which is distribution shift;

- ▶ the offline, closed-loop, and operational metrics, plus the safe fallback;
- ▶ the boundary at which a learned proposal becomes an authorized physical command.

That specification often reveals that the right architecture is not a single enormous model. A compact VLM, policy, state estimator, and certified controller can outperform a monolith when data are limited and failure costs are high. Conversely, broad pretraining may be decisive when tasks and language vary. Physical AI is architecture selection under interface constraints.



Figure 1.2: *Semantic competence becomes physical competence only through explicit interfaces.*

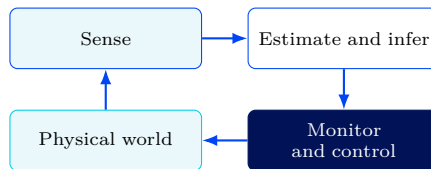


Figure 1.3: *The deployed unit is a causal feedback loop, not an isolated checkpoint.*

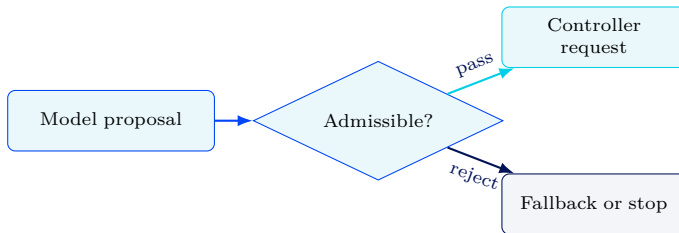


Figure 1.4: *A Physical AI stack separates generation from authorization.*



Principal references

- [1] Mechanics, planning, control, and the distinction between model action and robot control.
- [2] Belief-state and uncertainty foundations.
- [8] A landmark VLA case connecting web-scale knowledge to robot control.
- [9] Cross-embodiment data and RT-X evidence.
- [11] An open VLA implementation reference.
- [18] The modern neural world-model framing.
- [19] The JEPA-centered predictive-learning program.
- [16] The explicit 2026 WAM definition and taxonomy.



LESSON 2

The Physical Interface

LEARNING OBJECTIVES

After this lesson you can specify state, observation, and action interfaces; transform points and twists between coordinate frames; derive forward and differential kinematics; interpret rigid-body dynamics; formulate Bayesian state estimation; and account for time, latency, and partial observability in a learning system.

2.1 The interface is part of the model

Robot-learning failures are often blamed on architecture when the actual defect is an ambiguous interface. A dataset calls a tensor “state” although it contains only joint position. An action is documented as an incremental Cartesian end-effector command, meaning a pose update relative to a specified frame, but its rotation convention is absent. Camera frames and robot frames are offset by one sample. A policy trained at 20 Hz is executed at 10 Hz without rescaling its increments. None of these errors can be repaired by more parameters.

This emphasis comes directly from navigation practice. A small timing or frame error can overwhelm a far more interesting modeling improvement, yet it is easy to miss because the tensors still have the expected shape. The first serious review of a Physical AI system should therefore begin with interfaces, not with the network diagram.

Define a Physical AI interface with five attributes for every variable: meaning, shape, units, coordinate frame, and timestamp. For a trajectory, also define frequency and interpolation. A practical schema entry is not `action: float[7]`; it is:

`action.ee_delta: [ $\Delta x, \Delta y, \Delta z, \Delta r_x, \Delta r_y, \Delta r_z, g$ ], meters and axis-angle radi-`

ans in the base frame, gripper command $g \in [-1, 1]$, sampled at 20 Hz, applied relative to the measured pose at observation timestamp.

2.2 State, observation, information state, and belief

For a controlled Markov process,

$$p(\mathbf{x}_{t+1} \mid \mathbf{x}_{0:t}, \mathbf{u}_{0:t}) = p(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{u}_t). \quad (2.1)$$

Terms: $\mathbf{x}_t$ is a Markov state; $\mathbf{u}_t$ is control; $\mathbf{x}_{0:t}$ denotes the complete history. The equality says the current state and control contain all information required by the model of the next state.

The physical world can be Markov while the recorded vector is not. Joint positions omit velocities; an image hides contact force; a planar object pose omits friction and pressure distribution. Calling the recorded vector state does not make it sufficient.

The observation model is

$$\mathbf{o}_t = h(\mathbf{x}_t) + \mathbf{v}_t, \quad (2.2)$$

where h maps state to sensor space and $\mathbf{v}_t$ is measurement noise. A policy can condition on an **information state** $\mathbf{I}_t = \psi(\mathbf{o}_{0:t}, \mathbf{a}_{0:t-1})$, a compressed history produced by a filter, recurrent network, or transformer. A probabilistic **belief state** is

$$b_t(\mathbf{x}) = p(\mathbf{x}_t = \mathbf{x} \mid \mathbf{o}_{0:t}, \mathbf{u}_{0:t-1}). \quad (2.3)$$

Terms: b_t is a distribution over possible current states given all data; ψ is an information-state mapping.

An engineer chooses which object the policy consumes. Raw history permits learning but increases compute and data demand. A filtered state injects structure but can hide multimodality. A belief or ensemble exposes uncertainty but requires a downstream model able to use it.

2.3 Action is not control

A high-level learned action $\mathbf{a}_t$ may be a skill token, Cartesian waypoint, pose delta, joint target, velocity, torque, or chunk. The plant receives a control input $\mathbf{u}_t$. Their connection can be written

$$\mathbf{u}_{t,k} = \kappa(\mathbf{x}_{t,k}, \mathbf{a}_t), \quad k = 0, \dots, M - 1, \quad (2.4)$$



where κ is a lower-level controller and M fast control steps occur during one policy interval. A VLA can therefore be evaluated at the action interface without pretending it directly controls motor current.

The representation changes the learning problem:

- ▶ **joint position** is easy to supervise but embodiment-specific and can be discontinuous across inverse-kinematics branches;
- ▶ **Cartesian delta** transfers task geometry better but requires frame conventions and a controller to resolve redundancy;
- ▶ **velocity** is locally smooth but integrates bias and depends on policy rate;
- ▶ **torque** is expressive and contact-sensitive but demands high-rate, safety-critical learning;
- ▶ **action chunk** amortizes inference and models temporal coherence but increases commitment before feedback.

If $\Delta \mathbf{p}_t$ is a learned displacement per decision, executing it at a different interval changes velocity. If $\mathbf{v}_t$ is a velocity, the discrete displacement is $\Delta \mathbf{p}_t = \mathbf{v}_t \Delta t$. Store which one the dataset contains.

2.4 Coordinate frames and rigid transformations

A 3-D point is meaningful only relative to a coordinate frame. Let ${}^A \mathbf{p}$ denote point p expressed in frame A . A rigid transformation from frame B to frame A is

$${}^A T_B = \begin{bmatrix} {}^A R_B & {}^A \mathbf{t}_B \\ \mathbf{0}^\top & 1 \end{bmatrix} \in SE(3), \quad \begin{bmatrix} {}^A \mathbf{p} \\ 1 \end{bmatrix} = {}^A T_B \begin{bmatrix} {}^B \mathbf{p} \\ 1 \end{bmatrix}. \quad (2.5)$$

Terms: ${}^A R_B \in SO(3)$ rotates coordinates from B to A ; ${}^A \mathbf{t}_B \in \mathbb{R}^3$ is the origin of B expressed in A ; $SE(3)$ is the special Euclidean group.

Rotation matrices satisfy $R^\top R = I$ and $\det R = 1$. Composition is ordinary matrix multiplication:

$${}^A T_C = {}^A T_B {}^B T_C, \quad ({}^A T_B)^{-1} = {}^B T_A. \quad (2.6)$$

The order matters. A common data bug applies ${}^B T_A$ when the stored calibration is ${}^A T_B$. Unit tests should transform known basis points and round-trip random poses.

PHYSICAL FAILURE MODE - Frame mismatch

A numerically valid pose can be physically wrong when source frame, target frame, handedness, or rotation convention differs. Transform and test frames at the interface; tensor shape cannot reveal this error.



Numerical check. Let ${}^A R_B$ be a 90° yaw, ${}^A t_B = [1, 2, 0]^\top$ m, and ${}^B p = [1, 0, 0]^\top$ m. Rotation gives $[0, 1, 0]^\top$, then translation gives ${}^A p = [1, 3, 0]^\top$ m. Applying ${}^B T_A = ({}^A T_B)^{-1}$ returns the original point, providing a simple round-trip test of both order and frame labels.

An orientation may be stored as Euler angles, quaternion, rotation matrix, 6-D continuous representation, or axis-angle. Quaternions q and $-q$ represent the same rotation, so naive regression can punish equivalent labels. Euler angles have singularities and wraparound. For local Cartesian action, the rotation vector $\phi = \theta\omega$ is useful: ω is a unit axis and θ the rotation angle. For large rotations, use geodesic losses or group-aware integration.

Geodesic rotation error

$$d_R(R_1, R_2) = \cos^{-1} \left(\frac{\text{tr}(R_1^\top R_2) - 1}{2} \right). \quad (2.7)$$

Terms: $R_1, R_2 \in SO(3)$; tr is matrix trace; $d_R \in [0, \pi]$ is the shortest rotation angle between orientations.



EXERCISE 2.1 - A frame error that looks like a policy error

A camera estimates an object at ${}^C\mathbf{p} = [0.20, 0.05, 0.80]^\top$ m. The calibrated transform is ${}^B\mathbf{p} = {}^B R_C {}^C\mathbf{p} + {}^B\mathbf{t}_C$, with a 90° yaw,

$${}^B R_C = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad {}^B\mathbf{t}_C = [0.40, -0.10, 0.20]^\top \text{ m.}$$

The rotated point is ${}^B R_C {}^C\mathbf{p} = [-0.05, 0.20, 0.80]^\top$ m, so translation gives ${}^B\mathbf{p} = [0.35, 0.10, 1.00]^\top$ m. If the camera coordinates are incorrectly treated as base coordinates, the commanded point is $\mathbf{p}_{\text{wrong}} = [0.20, 0.05, 0.80]^\top$ m. Defining $\mathbf{e} = {}^B\mathbf{p} - \mathbf{p}_{\text{wrong}}$ gives $\mathbf{e} = [0.15, 0.05, 0.20]^\top$ m and $\|\mathbf{e}\|_2 = \sqrt{0.15^2 + 0.05^2 + 0.20^2} = 0.255$ m. The sign is now explicit, and the magnitude verifies the stated error. No policy retraining can repair an unmodeled frame contract. This calculation assumes synchronized rigid calibration; it does not include calibration covariance [1, 20].

2.5 Camera geometry

For a pinhole camera, a 3-D camera-frame point (X, Y, Z) projects to pixel (u, v) :

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad \lambda = Z. \quad (2.8)$$

Terms: f_x, f_y are focal lengths in pixels; (c_x, c_y) is the principal point; K is the intrinsic matrix; $Z > 0$ is depth. Lens distortion is omitted here but must be calibrated or modeled.

The equation is exact for an undistorted pinhole model in the stated convention. Its first two rows give

$$u = f_x \frac{X}{Z} + c_x, \quad v = f_y \frac{Y}{Z} + c_y.$$

For $(X, Y, Z) = (0.10, -0.05, 1.00)$ m, $f_x = f_y = 600$ pixels, and $(c_x, c_y) = (320, 240)$, the projection is $(u, v) = (380, 210)$ pixels. Changing the crop or re-size changes the effective values of f_x, f_y, c_x, c_y .

Pixels do not carry metric scale without depth, known geometry, multiview constraints, or learned prior. VLM language such as “slightly left” can be semantically correct while metrically unusable. A VLA trained end-to-end may learn calibration

implicitly, but camera replacement, crop, resize, or mount motion changes that hidden mapping. Store raw calibration and the exact image transform. If an image of original size (W, H) is resized and cropped, update the effective intrinsics; do not feed old K alongside new pixels.

2.6 Kinematics

Let $\mathbf{q} \in \mathbb{R}^n$ be joint coordinates. Forward kinematics maps configuration to end-effector pose:

$$T_{0e} = F(\mathbf{q}). \quad (2.9)$$

The differential relation is

$$\mathbf{V}_e = J(\mathbf{q})\dot{\mathbf{q}}, \quad (2.10)$$

where $\mathbf{V}_e \in \mathbb{R}^6$ is an end-effector twist, $J \in \mathbb{R}^{6 \times n}$ the Jacobian, and $\dot{\mathbf{q}}$ joint velocity. A twist stacks angular and linear velocity in a documented convention.

To track a desired twist $\mathbf{V}_d$, a damped least-squares inverse is

$$\dot{\mathbf{q}} = J^\top (JJ^\top + \lambda^2 I)^{-1} \mathbf{V}_d. \quad (2.11)$$

Terms: $\lambda > 0$ is damping; I is the 6×6 identity. Damping limits amplification near singular configurations.

EXERCISE 2.2 - Damping a near-singular Cartesian command

Consider a two-axis translational subproblem with

$$J = \begin{bmatrix} 1 & 0 \\ 0 & 0.05 \end{bmatrix} \text{ m/rad}, \quad \mathbf{V}_d = \begin{bmatrix} 0.10 \\ 0.10 \end{bmatrix} \text{ m/s}, \quad \lambda = 0.10 \text{ m/rad}.$$

The undamped inverse requests $\dot{\mathbf{q}} = [0.10, 2.00]^\top$ rad/s because the second singular direction has little Cartesian authority. Damped least squares instead gives

$$\dot{\mathbf{q}} = J^\top (JJ^\top + \lambda^2 I)^{-1} \mathbf{V}_d = \begin{bmatrix} 0.099 \\ 0.400 \end{bmatrix} \text{ rad/s}.$$

The realized Cartesian velocity is $J\dot{\mathbf{q}} = [0.099, 0.020]^\top$ m/s. Damping deliberately sacrifices motion in the weak direction to avoid the unsafe 2 rad/s joint request; the controller must still apply explicit joint-rate bounds.

This controller is a natural boundary below a Cartesian-action VLA. The policy proposes $\mathbf{V}_d$; the controller enforces rate limits and maps it to joints. Null-space motion



can avoid joint limits, but the policy must be evaluated with the actual controller because controller behavior changes the closed-loop distribution.

2.7 Dynamics and contact

Manipulator dynamics can be expressed as

$$M(\mathbf{q})\ddot{\mathbf{q}} + C(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) + \boldsymbol{\tau}_f = \boldsymbol{\tau} + J(\mathbf{q})^\top \mathbf{f}_{\text{ext}}. \quad (2.12)$$

Terms: M is the positive-definite inertia matrix; $C\dot{\mathbf{q}}$ Coriolis and centrifugal effects; $\mathbf{g}$ gravity torque; $\boldsymbol{\tau}_f$ friction; $\boldsymbol{\tau}$ commanded joint torque; $\mathbf{f}_{\text{ext}}$ external wrench.

A learned policy need not model each term explicitly, but the platform still obeys them. Acceleration, torque, power, and contact limits bound feasible action. Contact also creates hybrid dynamics: a small pose change can switch between free motion, impact, sticking, and sliding. A single smooth regressor tends to average these modes. History, tactile input, mixture distributions, and explicit mode variables can help.

For a one-dimensional contact normal with signed gap $\phi(\mathbf{q})$ and normal force λ_n , ideal nonpenetration is

$$\phi(\mathbf{q}) \geq 0, \quad \lambda_n \geq 0, \quad \phi(\mathbf{q})\lambda_n = 0. \quad (2.13)$$

Terms: Positive gap means separation; positive λ_n pushes bodies apart; complementarity means either separation with no contact force or contact with zero gap.

A concrete violation is easy to compute. If a prediction places one body 2 mm inside another, then $\phi = -0.002$ m and the nonpenetration condition already fails. If it predicts a 3 mm positive gap together with a 40 N contact force, then $\phi\lambda_n = 0.003 \times 40 = 0.12$ N m rather than zero: the model has created force without contact. A video can look smooth while either inconsistency is hidden between frames. Such an output is a dynamic Physical AI hallucination because the predicted state and force cannot satisfy the declared contact model.

2.8 State estimation

For nonlinear stochastic dynamics,

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t, \quad (2.14)$$

$$\mathbf{o}_t = h(\mathbf{x}_t) + \mathbf{v}_t, \quad (2.15)$$

the Bayes filter alternates prediction and correction.

► LANDMARK RESULT 2.1 - Belief-state sufficiency

Let the information history be

$$\mathcal{H}_t = (\mathbf{o}_{1:t}, \mathbf{u}_{1:t-1}), \quad b_t(\mathbf{x}) = p(\mathbf{x}_t = \mathbf{x} \mid \mathcal{H}_t).$$

Under Markov transition and reward models, together with a conditionally independent observation model, prediction and correction give

$$\bar{b}_t(\mathbf{x}_t) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}, \mathbf{u}_{t-1}) b_{t-1}(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}, \quad (2.16)$$

$$b_t(\mathbf{x}_t) = \eta_t p(\mathbf{o}_t \mid \mathbf{x}_t) \bar{b}_t(\mathbf{x}_t), \quad (2.17)$$

where η_t normalizes the posterior. The belief b_t is a sufficient information state: future prediction and expected-return control can use b_t instead of the entire history, so a POMDP can be written as a fully observed MDP on belief distributions [2].

Engineering meaning. A recurrent hidden vector is not automatically a belief state. The result requires the declared transition and observation assumptions, recorded actions, and a posterior that represents relevant uncertainty. Model mismatch, unmodeled contact, stale observations, and approximate inference weaken the conclusion.

Terms: $\mathcal{H}_t$ is observation-action history; $\bar{b}_t$ is predicted belief; b_t is corrected belief; η_t is its normalizer.

Kalman filters assume linear-Gaussian structure; extended and unscented filters approximate nonlinear systems; particle filters represent multimodal beliefs; learned filters combine neural features with recursive estimation. The correct choice depends on the dynamics, observability, nonlinearity, belief shape, and consequences of estimation error.

An estimator must expose age and covariance. A numerically precise pose that is 300 ms old can be more dangerous than a noisy current pose. Store the measurement timestamp τ_o , inference timestamp τ_i , and actuation timestamp τ_u . Observation age at actuation is $A = \tau_u - \tau_o$.

2.9 Time, synchronization, and latency

Suppose a sensor samples a moving coordinate $x(t)$ with timestamp error δt . A first-order approximation gives

$$\delta x \approx \dot{x}(t)\delta t. \quad (2.18)$$

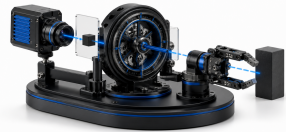
At 1 m/s, a 100 ms error is 10 cm. A dataset can therefore have perfect tensor alignment and poor physical alignment.

For multirate sensors, choose a reference clock, record hardware timestamps, estimate clock offsets, and specify resampling. Nearest-neighbor image matching is acceptable only when maximum time error is bounded relative to motion. Interpolate joint encoders on their manifold; never linearly average equivalent quaternions without sign handling and normalization.

Latency also changes feedback stability. For a scalar linear plant $x_{t+1} = ax_t + bu_t$ with delayed feedback $u_t = -kx_{t-d}$, stability depends on delay d , not only on gain k . Learned action chunks hide this issue if evaluated only offline. Measure end-to-end latency distribution under realistic concurrency, warmup, camera decoding, network transport, and thermal conditions.

PHYSICAL FAILURE MODE - Stale sensing

A precise estimate with an old timestamp is not current state. Carry capture and decision times, bound observation age, and reject or compensate measurements that exceed the control budget.



ENGINEERING NOTE - Interface acceptance test

For each trajectory, reconstruct a short segment in physical units, render frame axes, numerically differentiate positions, compare implied velocities with logged velocities, replay the low-level controller in simulation, and verify that every timestamp is monotonic. This catches more deployment defects than another architecture sweep.

2.10 Partial observability as a design decision

When a policy fails, distinguish three cases: the necessary variable is available but the model did not use it; it is present only in history; or it is never sensed. Larger networks help mainly in the first two. No architecture can infer an arbitrary hidden contact mode without informative evidence. Add sensing, active perception, conservative action, or a belief over modes.

A useful empirical test compares one-step predictors with increasing history length. If held-out likelihood improves strongly from $L = 1$ to $L > 1$, the recorded observation is not Markov for that task. If improvement saturates while residuals remain structured, a missing sensor or unmodeled regime is likely.

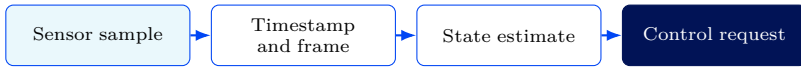


Figure 2.1: *A physical interface carries values and semantics together.*

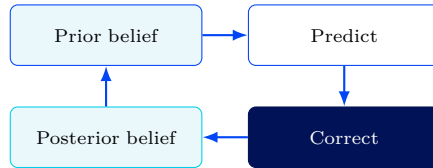


Figure 2.2: *State estimation closes a loop between prediction and measurement.*

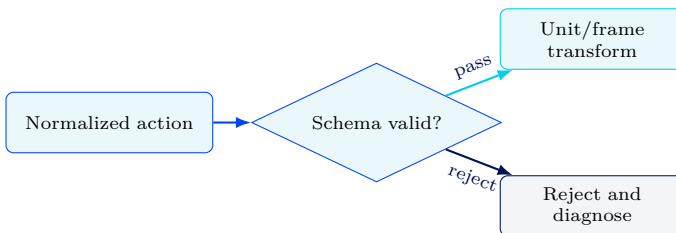


Figure 2.3: *Every action must be interpreted before it is executed.*



Principal references

- [1] Rigid transformations, kinematics, dynamics, and control.
- [21] Computational robotics, vision, and control workflows.
- [20] Modern state-estimation derivations and manifold treatment.
- [2] Probabilistic sensing and belief updates.
- [22] Mathematical manipulation foundations.
- [23] Classical robot mechanics and control interfaces.
- [24] Projective geometry and multi-view camera models.
- [25] State, motion, and estimation for mobile systems.

LESSON 3

Data and Learning from Demonstrations

LEARNING OBJECTIVES

After this lesson you can design an episode schema, normalize heterogeneous observations and actions, derive behavioral cloning and sequence objectives, explain covariate shift and DAgger, choose action chunks, construct splits that measure generalization, and audit a robot-learning dataset before training.

3.1 A dataset is a sampled interaction process

An image dataset contains exchangeable examples under a convenient approximation. A robot dataset contains trajectories produced by a policy interacting with dynamics. Ordering, delay, controller state, and episode boundaries are part of the signal.

Represent episode e as

$$\mathcal{D}^{(e)} = \left\{ \mathbf{o}_t^{(e)}, \mathbf{s}_t^{(e)}, \mathbf{a}_t^{(e)}, \mathbf{g}^{(e)}, r_t^{(e)}, d_t^{(e)}, \mathbf{m}_t^{(e)}, \tau_t^{(e)} \right\}_{t=0}^{T_e-1}. \quad (3.1)$$

Terms: $\mathbf{o}_t$ is observation; $\mathbf{s}_t$ recorded robot/environment state; $\mathbf{a}_t$ demonstrated action; $\mathbf{g}$ task or language goal; r_t optional reward; d_t termination flag; $\mathbf{m}_t$ validity mask; τ_t timestamp; T_e episode length.

A validity mask states which recorded values may participate in training or evaluation. For a scalar field, $m_{t,j} = 1$ means feature j at time t is observed, synchronized, and accepted; $m_{t,j} = 0$ means it is missing, padded, stale, or rejected by quality control. A dropped camera frame can therefore have $m_t^{\text{image}} = 0$ while joint encoders remain valid, and every padded step after episode termination must have $m_{t,j} = 0$. Masked losses divide by the number of valid entries, not by the padded tensor size.

The schema must also include calibration, robot type, control mode, software revision, operator, site, camera configuration, success label, intervention reason, and data license. Without provenance, adaptation becomes guesswork and leakage becomes invisible.

3.2 Collection policy shapes the data

Demonstrations can come from kinesthetic teaching, teleoperation, scripted controllers, autonomous rollouts, motion planners, video-to-action inference, or simulation. Each induces a distribution $d_{\pi_D}(s)$ over visited states, where π_D is the data-collection policy. The learned policy creates another distribution d_{π_θ} . Behavioral cloning is reliable when these distributions remain close and labels are appropriate.

High success alone is not sufficient. Perfect demonstrations cover a thin tube around expert behavior and may contain no recovery. Include corrections, near-boundary states, object and scene variation, negative outcomes, and explicit reset behavior. If failures are dangerous, collect recoveries in simulation or under a safety controller.

Quality has at least six dimensions:

- ▶ **semantic validity:** instruction and success label match what occurred;
- ▶ **temporal validity:** observations and actions refer to the intended physical instant;
- ▶ **geometric validity:** calibration and frames are correct;
- ▶ **control validity:** action represents what the robot actually attempted or executed;
- ▶ **coverage:** important tasks, objects, viewpoints, speeds, and recovery states appear;
- ▶ **independence of evaluation:** test conditions are not near-duplicates of training.

3.3 Behavioral cloning

Given demonstrations from expert π_E , maximum-likelihood behavioral cloning solves

$$\theta^* = \arg \min_{\theta} -\frac{1}{N} \sum_{i=1}^N \log \pi_{\theta}(\mathbf{a}_i \mid \mathbf{o}_{\leq i}, \mathbf{g}_i). \quad (3.2)$$

Terms: N is the number of supervised decision points; π_{θ} is the learned policy density; $\mathbf{a}_i$ the expert action; $\mathbf{o}_{\leq i}$ observation history; $\mathbf{g}_i$ goal.

For a Gaussian policy with fixed isotropic variance, negative log likelihood differs

from mean-squared error only by scale and a constant:

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N} \sum_i \| \mathbf{a}_i - \boldsymbol{\mu}_\theta(\mathbf{o}_{\leq i}, \mathbf{g}_i) \|_2^2. \quad (3.3)$$

MSE is inappropriate when actions are multimodal. At a symmetric obstacle, left and right paths are both valid; the conditional mean may collide. A mixture density policy uses

$$\pi_\theta(\mathbf{a} \mid \mathbf{o}) = \sum_{k=1}^K \alpha_k(\mathbf{o}) \mathcal{N}(\mathbf{a}; \boldsymbol{\mu}_k(\mathbf{o}), \Sigma_k(\mathbf{o})), \quad (3.4)$$

Terms: K is mixture count; $\alpha_k \geq 0$ and $\sum_k \alpha_k = 1$; $\boldsymbol{\mu}_k$ and Σ_k are component mean and covariance. Diffusion and flow policies represent richer multimodality without an explicit finite mixture.

Discrete action-token policies instead minimize cross entropy. If continuous component a_j is quantized into bin y_j , then

$$\mathcal{L}_{\text{token}} = - \sum_{j=1}^{d_a} \log p_\theta(y_j \mid \mathbf{o}, \mathbf{g}, y_{<j}). \quad (3.5)$$

Terms: d_a is action dimension; $y_{<j}$ are earlier generated action tokens. Quantization resolution trades precision for classification robustness.

EXERCISE 3.1 - Why an average action can collide

Suppose demonstrations pass an obstacle using two equally valid endpoint displacements,

$$\mathbf{a}_L = [-0.30, 0.20]^\top \text{ m}, \quad \mathbf{a}_R = [0.30, 0.20]^\top \text{ m},$$

with equal probability. An MSE regressor predicts the conditional mean $[0, 0.20]^\top \text{ m}$. If the obstacle occupies $|x| \leq 0.10 \text{ m}$ along that corridor, both demonstrated modes clear it while the mean collides. A two-component mixture can preserve the two modes instead of averaging them.

If the horizontal choices are tokenized as left and right with probability 0.5 each, the negative log likelihood of the observed token is $-\log 0.5 = 0.693$. After context resolves the right-hand route and the policy assigns probability 0.9, the loss falls to $-\log 0.9 = 0.105$. The number is meaningful only if the token decoder maps each class back to a physically documented action.

3.4 Why one-step imitation drifts

Behavioral cloning trains under expert states but deploys under its own. If a small error changes the next state, later observations depart from the demonstration distribution. In a simplified analysis, per-step error ϵ can yield horizon cost proportional to $O(T^2\epsilon)$, because errors compound and later states are unfamiliar.

Dataset Aggregation (DAgger) addresses this by iterating:

1. run a mixture of learner and expert policies;
2. query the expert action at states the learner visits;
3. aggregate those labeled states into the dataset;
4. retrain the policy.

At iteration k , a mixture policy can be $\pi_k = \beta_k \pi_E + (1 - \beta_k) \pi_{\theta_k}$, with β_k decreasing toward zero. DAgger shifts supervision toward $d_{\pi_{\theta}}$, but expert queries can be costly or unsafe. Alternatives include noise injection, perturb-and-recover demonstrations, autonomous relabeling, simulation, offline RL, and human intervention logs.

PHYSICAL FAILURE MODE - Executed versus commanded action

Teleoperation command, controller setpoint, clipped command, and measured motion are different labels. Train on the variable the policy will be asked to produce, but log all four when possible. Saturated commands teach a false mapping if only requested action is stored.



EXERCISE 3.2 - Why a small one-step error can dominate a rollout

Suppose an imitation policy has independent per-step probability $p = 0.98$ of remaining inside the demonstrated state tube. Under the deliberately simple independence approximation, the probability of staying inside for $H = 50$ steps is $p^H = 0.98^{50} \approx 0.364$. A 98% one-step figure therefore does not imply reliable long-horizon behavior.

If interactive collection raises the in-tube probability to 0.995, the same approximation gives $0.995^{50} \approx 0.778$. The point is sensitivity, not an exact field estimate: errors are usually correlated, and recovery can either improve or worsen the result. DAgger addresses the distribution mechanism by collecting labels on learner-induced states [26].

3.5 Action chunks and sequence supervision

An action-chunk policy predicts H future actions at once:

$$\hat{A}_t = [\hat{a}_t, \dots, \hat{a}_{t+H-1}] = f_\theta(o_{t-L+1:t}, g). \quad (3.6)$$

Terms: L is observation-history length; H action horizon; $\hat{A}_t \in \mathbb{R}^{H \times d_a}$.

Chunking offers three advantages: temporal coherence, fewer expensive model calls, and easier representation of coordinated contact. It also creates overlap. At time $t+1$, a new chunk may disagree with the tail of the previous chunk. Temporal ensembling combines predictions for the same absolute action time:

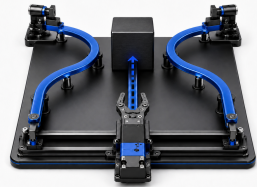
$$\bar{a}_\tau = \frac{\sum_{t: t \leq \tau < t+H} w_{\tau-t} \hat{a}_\tau^{(t)}}{\sum_{t: t \leq \tau < t+H} w_{\tau-t}}, \quad w_j = \exp(-\kappa j). \quad (3.7)$$

Terms: $\hat{a}_\tau^{(t)}$ is the prediction for absolute time τ made at t ; w_j discounts older chunk positions; $\kappa \geq 0$ is decay.

Choose H relative to task timescale and policy latency. Very short chunks recover feedback quickly but may be jittery. Very long chunks smooth behavior but can remain committed after a disturbance. Receding-horizon execution usually applies only the first $h_{\text{exec}} < H$ actions before replanning.

PHYSICAL FAILURE MODE - Action averaging

When demonstrations contain two safe modes, their conditional mean can cross an obstacle. Preserve multimodality and evaluate closed-loop roll-outs; low regression loss alone is not sufficient evidence.



Sequence padding must be masked. If $m_{t,j} = 0$ marks positions beyond an episode, the chunk loss is

$$\mathcal{L}_{\text{chunk}} = \frac{\sum_{t,j} m_{t,j} \ell(\hat{a}_{t+j}, a_{t+j})}{\sum_{t,j} m_{t,j}}. \quad (3.8)$$

Training on zero-padded labels without a mask teaches spurious episode-end motion.



3.6 Normalization across robots and datasets

For feature x_j , standardization uses

$$\tilde{x}_j = \frac{x_j - \mu_j}{\sigma_j + \epsilon}. \quad (3.9)$$

Terms: μ_j and σ_j are training-set mean and standard deviation; $\epsilon > 0$ avoids division by zero. At inference the same stored statistics must be used.

Robot actions often contain outliers and hard bounds, so quantile scaling can be more robust:

$$\tilde{x}_j = 2 \frac{\text{clip}(x_j, q_j^\ell, q_j^u) - q_j^\ell}{q_j^u - q_j^\ell} - 1. \quad (3.10)$$

Terms: q_j^ℓ and q_j^u are lower and upper training quantiles; clip truncates outside values.

Normalization is part of the checkpoint. Version it with the model and robot configuration. Recomputing statistics on test data leaks evaluation distribution. Combining robots also requires semantic normalization: joint 3 of one arm is not joint 3 of another. Common Cartesian action spaces, embodiment tokens, morphology encoders, and per-robot heads are different solutions to this alignment problem.

For image input, preserve whether pixels are RGB or BGR, range $[0, 1]$ or $[0, 255]$, gamma encoding, resolution, crop, augmentation, and camera order. A silent camera-key swap can yield plausible tensors and catastrophic control.

3.7 Dataset splitting without leakage

Randomly splitting frames leaks adjacent observations. Randomly splitting episodes can still leak the same scene, operator, object instance, or scripted trajectory. Choose the split unit according to the claim:

For numerical scale, independently assigning every frame of a 30 Hz trajectory to an 80/20 train/test split puts an adjacent pair on opposite sides with probability $2(0.8)(0.2) = 0.32$. A ten-second, 300-frame trajectory has 299 adjacent pairs, so the expected number of cross-split adjacent pairs is about $299 \times 0.32 \approx 96$. Those pairs are only 33 ms apart and can make test performance nearly meaningless.

- ▶ **new episode:** recovery from stochastic variation in known settings;
- ▶ **new object instance:** object-level generalization;
- ▶ **new scene or site:** background and geometry shift;

- ▶ **new task or instruction composition:** semantic generalization;
- ▶ **new embodiment:** morphology and controller transfer;
- ▶ **future time:** robustness to hardware and process drift.

Store split manifests, not only random seeds. Deduplicate images and trajectory hashes. If foundation-model pretraining data are unknown, acknowledge possible semantic leakage and prefer physical closed-loop tests using private configurations.

3.8 Evaluation: from loss to behavior

Offline metrics include negative log likelihood, MSE, angular error, endpoint error, action smoothness, and calibration. For action chunks, endpoint and temporal errors are

$$\text{ADE} = \frac{1}{H} \sum_{j=0}^{H-1} \|\hat{\mathbf{p}}_{t+j} - \mathbf{p}_{t+j}\|_2, \quad (3.11)$$

$$\text{FDE} = \|\hat{\mathbf{p}}_{t+H-1} - \mathbf{p}_{t+H-1}\|_2. \quad (3.12)$$

Terms: ADE is average displacement error; FDE is final displacement error; $\mathbf{p}$ is a position or state coordinate. Neither determines task success when multiple futures are valid.

Closed-loop metrics must include task success with confidence intervals, partial progress, time, collisions, constraint violations, human interventions, fallback rate, and reset count. Use identical initial-state distributions and enough trials. Report conditional results by difficulty rather than only a pooled mean.

The binomial standard error of observed success $\hat{p} = s/n$ is approximately $\sqrt{\hat{p}(1 - \hat{p})/n}$, where s successes occur in n trials. With only ten trials, a reported 80 percent success rate is highly uncertain. Prefer exact or Wilson intervals near 0 and 1.

3.9 Open robot-learning data

Open X-Embodiment assembled heterogeneous datasets across many embodiments and motivated standardized cross-robot training. DROID contributed large-scale in-the-wild manipulation data collected across sites. BridgeData V2, LIBERO, PushT, and simulation suites cover complementary regimes. LeRobot provides a common dataset representation, conversion tools, policy implementations, and Hub integration.



Scale is not a substitute for compatibility. Combining datasets can hurt when action semantics, language quality, timing, or observation keys differ. Build a dataset matrix with rows as sources and columns as action representation, frequency, embodiment, sensors, license, success labels, task taxonomy, and calibration. Exclude or adapt sources whose semantics cannot be reconciled.

EXERCISE 3.3 - A complete demonstration and storage budget

Assume 12 tasks, 50 demonstrations per task, 90 seconds per demonstration, and 30 Hz logging. The dataset contains

$$12 \times 50 \times 90 \times 30 = 1,620,000 \text{ time steps,}$$

or 15 hours of robot interaction. Two compressed RGB streams at 2.5 Mbit/s each require approximately

$$5 \text{ Mbit/s} \times 54,000 \text{ s} / 8 \approx 33.75 \text{ GB.}$$

If state plus action contain 64 float32 values, the uncompressed numerical stream adds about $1.62 \times 10^6 \times 64 \times 4 \approx 415 \text{ MB}$ before metadata and indices. A 20% recollection reserve raises the planned interaction time to 18 hours. These figures are an engineering budget under stated codec assumptions, not Mobile ALOHA's published storage footprint.

3.10 A practical audit before training

Run the following checks:

1. schema and dtype validation, including missing keys and NaNs;
2. timestamp monotonicity, sensor-action lag estimate, and policy-rate histogram;
3. per-dimension action distribution, clipping frequency, and unit sanity;
4. video-contact alignment and representative episode playback;
5. frame-transform round trips and calibration drift;
6. episode length, success, task, site, operator, and robot balance;
7. duplicate and leakage analysis across proposed splits;
8. baseline behavior cloning plus closed-loop or replay sanity check;
9. a dataset card that records collection policy, known failure modes, license, and intended use.

ENGINEERING NOTE - The smallest convincing baseline

Before fine-tuning a foundation VLA, train a compact MLP or ACT-style policy on state and action. If the baseline cannot overfit a tiny subset, the pipeline is defective. If it succeeds on state but not images, investigate visual preprocessing. If offline loss is low but rollouts fail, investigate shift, timing, multimodality, and controller mismatch.



Figure 3.1: A demonstration becomes supervision only after its contracts are audited.

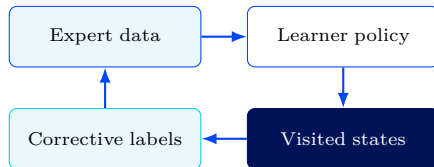


Figure 3.2: Interactive imitation changes the states represented in the data.

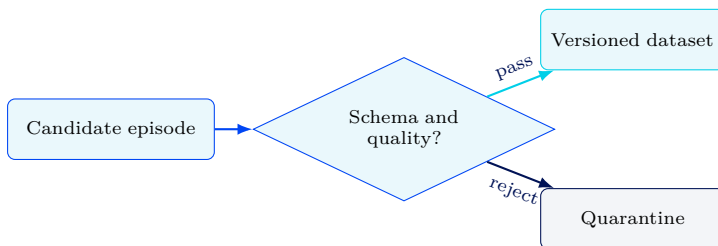


Figure 3.3: Dataset release requires semantic as well as numerical validity.



Principal references

- [26] Distribution shift and interactive imitation learning.
- [27] Action diffusion for visuomotor policies.
- [28] Action chunking and low-cost bimanual data collection.
- [9] Cross-embodiment robot datasets.
- [29] Large-scale in-the-wild manipulation data.
- [30] Broad robot-learning data and transfer.
- [31] Lifelong transfer benchmarking.
- [32] Open end-to-end robot-learning infrastructure.
- [33] Large-scale real-robot data collection.

MODULE II

From Perception to Action



LESSON 4

VLMs for Grounded Perception

LEARNING OBJECTIVES

After this lesson you can derive the main VLM representation and attention objectives; distinguish contrastive, generative, and fusion architectures; design spatially grounded outputs; adapt a VLM with full fine-tuning or LoRA; and evaluate semantic competence separately from geometry, state change, and affordance.

4.1 What a VLM contributes

Place a mug, a hex key, and a sealed box on a workbench. A person can ask for “the tool beside the mug” without supplying class IDs, coordinates, or a segmentation mask. The robot still receives pixels. Vision-language models are valuable because they connect that human description to visual evidence.

They bring object categories, attributes, relations, instructions, and broad web knowledge into Physical AI. That makes them strong proposal generators for perception and high-level planning. It does not make metric geometry, current object state, or control appear automatically; those properties need their own interfaces and tests.

A VLM module can serve at least five roles:

- ▶ encode images and language into aligned embeddings;
- ▶ answer questions about a scene;
- ▶ ground phrases to regions, points, masks, or 3-D entities;
- ▶ generate structured scene descriptions or task decompositions;
- ▶ supply a semantic backbone to a VLA or world model.

The required role determines the output contract. Free-form text is suitable for explanation; a controller needs coordinates, identities, uncertainties, and timestamps.

4.2 Vision tokens

For an image $I \in \mathbb{R}^{H \times W \times C}$, a vision transformer divides it into patches of size $P \times P$. The number of patches is $N = (H/P)(W/P)$. Flattened patch $\mathbf{x}_i \in \mathbb{R}^{P^2 C}$ becomes token

$$\mathbf{z}_i^{(0)} = E\mathbf{x}_i + \mathbf{p}_i, \quad (4.1)$$

Terms: $E \in \mathbb{R}^{d \times P^2 C}$ is a learned projection; d is embedding width; $\mathbf{p}_i \in \mathbb{R}^d$ is a positional embedding; $i = 1, \dots, N$.

Patch size controls a physical tradeoff. Large patches reduce compute but can erase small grasp features. High resolution improves localization but increases token count quadratically with side length. Cropping can remove context or change intrinsics. Multi-scale encoders, region features, and image pyramids are useful when small objects and global layout both matter.

Text is tokenized into subwords $w_1, \dots, w_M$ and embedded as

$$\mathbf{h}_j^{(0)} = W_e \text{onehot}(w_j) + \mathbf{r}_j, \quad (4.2)$$

where W_e is a token-embedding matrix and $\mathbf{r}_j$ positional information. Robot instructions contain uncommon object names, coordinates, and abbreviations; inspect their tokenization. A label split into many rare pieces can be poorly represented.

4.3 Attention

Given token matrix $X \in \mathbb{R}^{n \times d}$, scaled dot-product attention is

$$\text{Attn}(X) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad Q = XW_Q, K = XW_K, V = XW_V. \quad (4.3)$$

Terms: $W_Q, W_K \in \mathbb{R}^{d \times d_k}$ and $W_V \in \mathbb{R}^{d \times d_v}$ are learned projections; Q, K, V are queries, keys, values; softmax is row-wise.

Padding, causal structure, or visibility rules can optionally mask selected query-key pairs. In that case, add a mask $M \in \mathbb{R}^{n \times n}$ to the logits before softmax, with $M_{ij} = 0$ for a permitted pair and $M_{ij} = -\infty$ for a blocked pair. Implementations use the corresponding dtype-safe large negative value rather than literal infinity.

Multi-head attention computes this operation in several subspaces:

$$\text{MHA}(X) = \text{Concat}(H_1, \dots, H_h)W_O, \quad (4.4)$$



where H_j is attention head j , h is head count, and W_O projects concatenated values. Residual connections, layer normalization, and a feed-forward network complete a transformer block.

Cross-attention uses queries from one modality and keys/values from another. If language tokens query visual tokens,

$$\text{CrossAttn}(X_L, X_V) = \text{softmax}\left(\frac{X_L W_Q (X_V W_K)^T}{\sqrt{d_k}}\right) X_V W_V. \quad (4.5)$$

Terms: X_L is the language-token matrix and X_V the visual-token matrix. The output updates language representations using image evidence.

Attention weight is not a calibrated explanation. It is an internal routing coefficient and can remain high on a region while the final answer is wrong. Verify grounding with output-level perturbations and task-specific labels.

4.4 Three VLM training paradigms

4.4.1 Contrastive alignment

CLIP-style training uses paired image and text embeddings. Let normalized embeddings be $\mathbf{v}_i = f_V(I_i)/\|f_V(I_i)\|$ and $\mathbf{t}_i = f_T(q_i)/\|f_T(q_i)\|$ for batch size B . The symmetric contrastive loss is

$$\mathcal{L}_{I \rightarrow T} = -\frac{1}{B} \sum_{i=1}^B \log \frac{\exp(\mathbf{v}_i^T \mathbf{t}_i / \tau)}{\sum_{j=1}^B \exp(\mathbf{v}_i^T \mathbf{t}_j / \tau)}, \quad (4.6)$$

$$\mathcal{L}_{T \rightarrow I} = -\frac{1}{B} \sum_{i=1}^B \log \frac{\exp(\mathbf{t}_i^T \mathbf{v}_i / \tau)}{\sum_{j=1}^B \exp(\mathbf{t}_i^T \mathbf{v}_j / \tau)}, \quad (4.7)$$

$$\mathcal{L}_{\text{CLIP}} = \frac{1}{2} (\mathcal{L}_{I \rightarrow T} + \mathcal{L}_{T \rightarrow I}). \quad (4.8)$$

Terms: $\tau > 0$ is a learned or fixed temperature; matching index i is the positive pair; other batch members are negatives.

Contrastive encoders support retrieval, classification, reward features, and initialization. They do not generate detailed answers unless connected to a decoder. Batch composition matters because false negatives can describe the same object.

4.4.2 Autoregressive generation

A generative VLM models answer tokens $y_1, \dots, y_T$:

$$p_{\theta}(\mathbf{y} \mid I, \mathbf{q}) = \prod_{t=1}^T p_{\theta}(y_t \mid y_{<t}, I, \mathbf{q}), \quad \mathcal{L}_{\text{NLL}} = - \sum_{t=1}^T \log p_{\theta}(y_t \mid y_{<t}, I, \mathbf{q}). \quad (4.9)$$

Terms: $\mathbf{q}$ is prompt token sequence; $y_{<t}$ are preceding target tokens; teacher forcing supplies ground-truth prefixes during training.

The decoder can produce language, bounding-box tokens, keypoints, robot program tokens, or a structured JSON-like schema. Constrained decoding reduces syntax errors but cannot prove the values correspond to the scene.

4.4.3 Perceiver and adapter fusion

Large language models can receive compressed visual tokens through a learned connector. Flamingo-style gated cross-attention inserts image access into a language model; PaLM-E injects continuous sensor and visual embeddings; projection adapters map a frozen vision encoder into a frozen or lightly tuned language backbone. These designs exploit pretrained capability and lower adaptation cost, but the connector can become an information bottleneck.

4.5 Grounding for physical use

Grounding converts a phrase such as “the blue handle” into a referent. Useful output spaces include:

- ▶ bounding box $\mathbf{b} = (c_x, c_y, w, h)$ in normalized image coordinates;
- ▶ point (u, v) for a target pixel;
- ▶ segmentation mask $S \subset \{1, \dots, H\} \times \{1, \dots, W\}$;
- ▶ 3-D point ${}^C\mathbf{p}$ or pose BT_O ;
- ▶ persistent object identity in a scene graph.

For box prediction, generalized intersection-over-union can supplement coordinate loss. Standard IoU is

$$\text{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad (4.10)$$

where A and B are predicted and target pixel regions. IoU is zero for nonoverlapping boxes and gives no distance gradient there; L1 coordinates or generalized IoU address this.

Metric grounding requires camera geometry. With depth $D(u, v)$, pixel $\tilde{\mathbf{p}} = (u, v, 1)^\top$ back-projects to

$${}^C \mathbf{p} = D(u, v) K^{-1} \tilde{\mathbf{p}}, \quad {}^B \mathbf{p} = {}^B T_C \begin{bmatrix} {}^C \mathbf{p} \\ 1 \end{bmatrix}. \quad (4.11)$$

Terms: K is camera intrinsic matrix; ${}^B T_C$ camera-to-base transform; homogeneous division is implicit in the second expression.

Depth noise, calibration error, segmentation edges, and time skew should propagate into uncertainty. If pixel covariance is Σ_{uvd} , a first-order 3-D covariance is

$$\Sigma_p \approx J_g \Sigma_{uvd} J_g^\top, \quad (4.12)$$

where $g(u, v, d) = dK^{-1}[u, v, 1]^\top$ and $J_g = \partial g / \partial (u, v, d)$. A downstream grasp planner can then expand margins or request another view.

PHYSICAL FAILURE MODE - Grounding depth error

Correct object identity with wrong depth or occlusion order produces an empty-space grasp. Calibrate geometry and test the metric handoff independently from semantic accuracy.



4.6 Temporal and state grounding

A single image is insufficient for velocity, permanence behind occlusion, or whether a drawer is opening versus closing. Video VLMs add frame or tube tokens, temporal position, and causal or bidirectional attention. If frames are $I_{t-L+1:t}$, the answer distribution is $p_\theta(\mathbf{y} \mid I_{t-L+1:t}, \mathbf{q})$.

The input window must carry physical timing. Uniformly sampling eight frames from clips of different duration produces ambiguous velocity. Provide timestamps or normalize sampling consistently. If a scene can change during inference, tag the output with the final observation time and reject it when age exceeds the planning budget.

Object-state understanding should use closed vocabularies where operationally possible: open/closed/unknown, free/occluded/unknown, or probability distributions rather than confident prose. “Unknown” is a capability, not a defect.

4.7 Fine-tuning and LoRA

Full fine-tuning updates every parameter and offers maximum capacity at high memory and overfitting risk. Freezing the backbone and training a projection head is cheap but limited. Low-rank adaptation updates a weight $W_0 \in \mathbb{R}^{d_{out} \times d_{in}}$ through

$$W = W_0 + \Delta W, \quad \Delta W = \frac{\alpha}{r} BA, \quad (4.13)$$

Terms: $A \in \mathbb{R}^{r \times d_{in}}$, $B \in \mathbb{R}^{d_{out} \times r}$; rank $r \ll \min(d_{in}, d_{out})$; α scales the update; W_0 is frozen.

Trainable parameters are roughly $r(d_{in} + d_{out})$ instead of $d_{in}d_{out}$. LoRA can target attention query/value projections, MLP layers, connector, or language head. The choice should follow where the task differs: new visual domain may require vision adaptation; new vocabulary may require language/connector changes; new structured output may need a decoder head.

For a small Physical AI dataset, a staged approach is effective:

1. freeze encoders and train a grounded output head;
2. inspect failure by object, view, distance, and state;
3. add LoRA to the connector and upper encoder blocks;
4. unfreeze more only if validation remains bias-limited;
5. preserve a broad-data replay subset to reduce catastrophic forgetting.

Use augmentations consistent with action semantics. Horizontal flipping swaps left and right; random crop changes effective camera geometry; color jitter can hide safety markings. When applying an image transform, update labels and calibration.

4.8 Evaluating a VLM for Physical AI

General VQA accuracy is not enough. Build an evaluation matrix:

Dimension	Example measure
Existence	precision/recall for objects, including negative scenes
Identity	persistent association across view and occlusion
Geometry	point/box/mask error and 3-D pose error
Relations	left/right, support, containment, collision ordering
State	open/closed, occupied/free, connected/disconnected
Affordance	grasp/place/press feasibility conditioned on embodiment
Temporal change	direction, speed bin, event order, stale-scene detection
Uncertainty	calibration, selective risk, abstention quality
Robustness	lighting, blur, clutter, prompt variation, distractors



For a probabilistic binary output p_i and label $y_i \in \{0, 1\}$, the Brier score is

$$\text{BS} = \frac{1}{N} \sum_{i=1}^N (p_i - y_i)^2. \quad (4.14)$$

Lower is better, but calibration should also be plotted by confidence bin. Selective prediction evaluates error when the model is allowed to abstain. For retained set $S_c = \{i : c_i \geq c\}$, risk is $R(c) = |S_c|^{-1} \sum_{i \in S_c} \ell_i$. The risk-coverage curve reveals whether confidence identifies difficult cases.

Prompt robustness matters because downstream systems often template instructions. Evaluate paraphrases and adversarial presuppositions: “Which red tool should the robot grasp?” when no red tool exists. The correct response must preserve the negative premise rather than comply linguistically.

EXERCISE 4.1 - Calibration changes a grounding decision

In 100 predictions assigned confidence 0.9, only 63 are correct. This bin contributes $\frac{100}{N} |0.63 - 0.90|$ to expected calibration error (ECE), where N is the full evaluation size. If $N = 500$, the contribution is 0.054, already substantial before the other bins are counted.

Temperature scaling fits a positive scalar temperature on a held-out calibration set, usually by minimizing negative log likelihood while leaving the predicted class ranking unchanged [34]. On data drawn from the same distribution it often reduces ECE, but it is not guaranteed to improve calibration under distribution shift and can overfit a small calibration set. It also cannot repair missing geometry. For an action-facing VLM, evaluate object existence, mask/box overlap, depth ordering, temporal persistence, and abstention separately. A lower held-out ECE supports threshold selection; it does not prove physical grounding or safe action.

4.9 From VLM to action without category error

A safe modular path is:

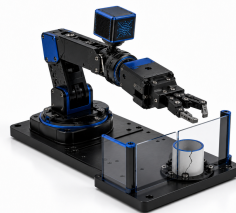
image $\rightarrow$ grounded entities $\rightarrow$ metric scene state
 $\rightarrow$ planner $\rightarrow$ controller $\rightarrow$ monitor.

The VLM proposes semantic entities; calibrated geometry produces metric state; a planner selects action; control and runtime checks enforce constraints. End-to-end

VLMs can learn several arrows jointly, but the interfaces remain useful for testing and fallback.

PHYSICAL FAILURE MODE - Semantics is not affordance

Recognizing “mug” does not establish that the mug is reachable, unbroken, empty, light enough, stable under the intended grasp, or permitted to move. Affordance is a relation among object, state, embodiment, task, and time.



ENGINEERING NOTE - A grounded output contract

Return `entity_id`, class distribution, image region, 3-D pose distribution, state distribution, visibility, observation timestamp, and an explicit unknown flag. Preserve raw evidence references so later failures can be traced to the source image and calibration.



Figure 4.1: *Grounded VLM use preserves the path from pixels to physical variables.*

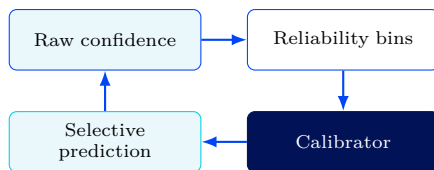


Figure 4.2: *Calibration requires a held-out feedback loop.*

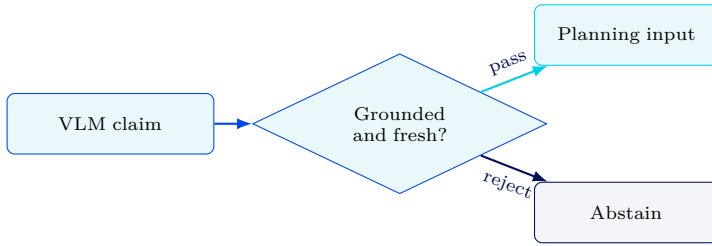


Figure 4.3: *Language output should not bypass grounding checks.*

Principal references

- [35] Contrastive vision-language pretraining.
- [36] Noisy web-scale vision-language alignment.
- [37] Few-shot multimodal generation.
- [38] Embodied multimodal language modeling.
- [39] Sigmoid vision-language alignment objective.
- [40] Promptable segmentation foundation model.
- [41] Low-rank parameter-efficient adaptation.
- [34] Neural confidence calibration.
- [42] Model interpretation foundations.

LESSON 5

VLAAs and Robot Foundation Models

LEARNING OBJECTIVES

After this lesson you can formalize a VLA policy; compare token, regression, autoregressive, diffusion, and flow action heads; explain action chunking and two-system architectures; assess foundation-model transfer; and choose an architecture from embodiment, data, latency, and multimodality requirements.

5.1 A VLA is a conditional action generator

A vision-language-action model receives visual evidence, a language-conditioned task, and usually robot state. The most general practical interface is

$$\mathbf{a}_{t:t+H-1} \sim \pi_{\theta}(\cdot \mid \mathbf{I}_{t-L+1:t}, \mathbf{r}_{t-L+1:t}, \mathbf{q}, \mathbf{e}, \mathbf{c}). \quad (5.1)$$

Terms: $\mathbf{I}$ is image history from one or more cameras; $\mathbf{r}$ proprioceptive history; $\mathbf{q}$ language instruction; $\mathbf{e}$ embodiment descriptor; $\mathbf{c}$ optional context such as calibration or previous action; L history length; H action horizon.

The model name says nothing about action semantics. Before training, specify $\mathbf{a}_t \in \mathcal{A}_e$, where the action space depends on embodiment e . An action decoder D_e and controller κ_e form the physical interface:

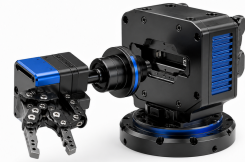
$$\mathbf{a}_t^{\text{phys}} = D_e(\mathbf{a}_t^{\text{model}}, \eta_e), \quad \mathbf{u}_{t,k} = \kappa_e(\mathbf{x}_{t,k}, \mathbf{a}_t^{\text{phys}}). \quad (5.2)$$

Terms: η_e contains robot-specific normalization, limits, frames, and control mode; $\mathbf{u}_{t,k}$ is the fast control command.

This decoder belongs in the model card. Two checkpoints with identical tensors are not equivalent if one expects meters in base frame and the other normalized deltas in tool frame.

PHYSICAL FAILURE MODE - Action-space mismatch

Normalized deltas, absolute poses, torques, and controller setpoints are not interchangeable. A VLA checkpoint is unsafe without its decoder, units, frame, rate, and embodiment contract.



5.2 The common VLA backbone

A typical VLA contains:

1. a vision encoder, often pretrained by contrastive or self-supervised learning;
2. a text tokenizer and language model or text encoder;
3. a projector that aligns vision/proprioception with the transformer width;
4. a multimodal transformer that fuses tokens and context;
5. an action expert or head that produces robot commands;
6. normalization, temporal aggregation, and embodiment-specific decoding.

Let visual tokens be $V \in \mathbb{R}^{N_v \times d}$, language tokens $L \in \mathbb{R}^{N_l \times d}$, robot tokens $R \in \mathbb{R}^{N_r \times d}$, and action-query tokens $A_0 \in \mathbb{R}^{H \times d}$. A unified transformer computes

$$Z = \text{Transformer}_\theta([V; L; R; A_0], M), \quad (5.3)$$

where $[\cdot; \cdot]$ concatenates along sequence and M determines which tokens can attend to which. The action output is decoded from the action-token positions. A causal mask enables autoregressive generation; bidirectional action queries enable parallel chunks.

The backbone can be monolithic or divided. A slow semantic system interprets scene and task; a fast action expert handles continuous control. This separation allows web-scale language knowledge and robot-specific dynamics to use different widths, rates, and training data.

5.3 Five action-generation families

5.3.1 Direct regression

A deterministic head predicts $\hat{A} = g_\theta(Z_A) \in \mathbb{R}^{H \times d_a}$ and minimizes

$$\mathcal{L}_{\text{reg}} = \frac{1}{\sum_{t,j} m_{t,j}} \sum_{t,j} m_{t,j} \|W(\hat{\mathbf{a}}_{t,j} - \mathbf{a}_{t,j})\|_1. \quad (5.4)$$

Terms: $m_{t,j}$ is a validity mask; W weights action dimensions or converts normalized error to comparable physical units. Smooth L1 or MSE are alternatives.

Regression is fast and stable when the conditional action is nearly unimodal. It averages distinct strategies and often produces cautious or physically invalid midpoints in multimodal tasks.

5.3.2 Action tokenization

Each continuous component can be discretized. Uniform binning over $[l_j, u_j]$ maps

$$y_j = \text{round}\left((B - 1) \frac{\text{clip}(a_j, l_j, u_j) - l_j}{u_j - l_j}\right), \quad (5.5)$$

where B is bin count. The model predicts tokens autoregressively or in parallel with cross entropy.

Tokenization reuses language-model machinery and represents multimodality. Its resolution is $(u_j - l_j)/(B - 1)$; precision degrades when ranges are wide. Per-dimension bins ignore correlation unless the transformer models it through token order. Vector quantization can learn joint action codes but adds a codebook and possible collapse.

5.3.3 ACT and latent-variable chunks

Action Chunking with Transformers uses a conditional variational autoencoder. An encoder approximates posterior $q_\phi(\mathbf{z} \mid A, \mathbf{o})$, while a decoder predicts the chunk from observation and latent style. The objective is

$$\mathcal{L}_{\text{ACT}} = \mathbb{E}_{q_\phi(\mathbf{z} \mid A, \mathbf{o})} [\ell(g_\theta(\mathbf{o}, \mathbf{z}), A)] + \beta D_{\text{KL}}(q_\phi(\mathbf{z} \mid A, \mathbf{o}) \parallel \mathcal{N}(0, I)). \quad (5.6)$$

Terms: A is demonstrated action chunk; $\mathbf{z}$ latent style; ℓ reconstruction loss; D_{KL} Kullback-Leibler divergence; β regularization weight. At inference $\mathbf{z}$ can be fixed or sampled from the prior.

ACT is attractive for low-cost imitation and bimanual coordination. A VLA can replace or condition its visual-language front end while retaining chunk decoding.

EXERCISE 5.1 - Choosing an executed prefix for an action chunk

A policy predicts $H = 16$ actions at 20 Hz, a horizon of 0.8 s. Executing all 16 before replanning exposes the robot open-loop for the full 0.8 s. Executing the first $K = 4$ actions reduces exposure to $K/20 = 0.20$ s and invokes inference 5 times per second.

If p95 inference plus transfer is 150 ms, $K = 2$ gives only 100 ms between replans and is not schedulable without asynchronous inference; $K = 4$ leaves 50 ms. The selected prefix therefore couples model latency, control rate, and disturbance tolerance. ACT, diffusion, and flow policies change how the chunk is generated, not this timing obligation [27, 28, 43].

PHYSICAL FAILURE MODE - Open-loop drift

Long action chunks reduce inference load but delay correction. Measure drift across the executed prefix and shorten, replan, or interrupt when state divergence exceeds the deployment envelope.



5.3.4 Diffusion policy

Diffusion treats an action chunk as a sample refined from noise. A forward process corrupts clean action A^0 :

$$A^k = \sqrt{\bar{\alpha}_k} A^0 + \sqrt{1 - \bar{\alpha}_k} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I), \quad (5.7)$$

where k is diffusion step and $\bar{\alpha}_k$ a cumulative noise schedule. A denoiser conditioned on observation learns

$$\mathcal{L}_{\text{diff}} = E_{A^0, k, \epsilon} \left[\left\| \epsilon - \epsilon_\theta(A^k, k, \mathbf{o}, \mathbf{q}) \right\|_2^2 \right]. \quad (5.8)$$

Terms: ϵ_θ predicts injected noise; iterative reverse updates turn random A^K into an action sample.

Diffusion represents multimodal, correlated chunks and often yields smooth behavior. Iterative sampling increases inference cost. Distillation, fewer solver steps, consistency models, and receding-horizon reuse reduce latency.

5.3.5 Flow matching

Flow matching learns a continuous velocity field transporting noise $A_0 \sim p_0$ to data $A_1 \sim p_{\text{data}}$. For simple linear interpolation

$$A_s = (1 - s)A_0 + sA_1, \quad u_s(A_s \mid A_0, A_1) = A_1 - A_0, \quad s \in [0, 1]. \quad (5.9)$$

A conditional vector field $v_\theta(A_s, s, \mathbf{o}, \mathbf{q})$ minimizes

$$\mathcal{L}_{\text{FM}} = \mathbb{E} [\|v_\theta(A_s, s, \mathbf{o}, \mathbf{q}) - (A_1 - A_0)\|_2^2]. \quad (5.10)$$

At inference solve the ordinary differential equation

$$\frac{dA_s}{ds} = v_\theta(A_s, s, \mathbf{o}, \mathbf{q}), \quad A_{s=0} \sim p_0. \quad (5.11)$$

Terms: s is generative time, not physical time; the solver integrates from noise distribution p_0 to an action sample. More sophisticated probability paths can improve training.

Flow action heads, used by π_0 -family systems and successors, can require fewer inference evaluations than classical diffusion while preserving continuous multimodality. Solver count still belongs in the latency budget.

5.4 What makes a robot model foundational

A robot foundation model is judged by reuse. Let domains be $d \in \mathcal{D}$, with tasks, scenes, and embodiments. Pretraining minimizes a weighted multi-domain objective

$$\mathcal{L}(\theta) = \sum_{d \in \mathcal{D}} \lambda_d \mathbb{E}_{(\mathbf{o}, \mathbf{q}, A) \sim \mathcal{D}_d} [\ell_d(\pi_\theta(\mathbf{o}, \mathbf{q}, \mathbf{e}_d), A)]. \quad (5.12)$$

Terms: λ_d balances domains; $\mathbf{e}_d$ is embodiment context; ℓ_d can differ by action representation.

Raw sampling proportional to dataset size lets the largest robot dominate. Temperature sampling uses $p_d \propto n_d^\gamma$, with dataset size n_d and $0 < \gamma < 1$, to flatten imbalance. Per-dataset normalization and heads prevent incompatible actions from being averaged.



Transfer should be measured by adaptation efficiency: performance versus target demonstrations, trainable parameters, compute, and hardware changes. A model is more foundational if it reaches useful behavior with fewer target examples and preserves prior capability.

Negative transfer remains possible. Web vision features may recognize objects but ignore contact. Cross-robot data may create conflicting action conventions. Language can dominate vision, causing the policy to follow a typical instruction despite contrary scene evidence. Foundation pretraining increases prior knowledge and can increase prior dominance.

5.5 Architectural case studies

5.5.1 RT-1 and RT-2

RT-1 demonstrated a transformer policy trained over large real-robot data with tokenized actions. RT-2 co-fine-tuned vision-language data and robot trajectories, representing actions as tokens so web-derived semantic knowledge could transfer to control. The engineering lesson is not that every action should be text. It is that shared token machinery can connect broad semantic pretraining to robot output when action encoding is precise.

5.5.2 Open X-Embodiment, Octo, and OpenVLA

Open X-Embodiment standardized a broad collection of robot datasets and RT-X experiments. Octo trained an open generalist policy across heterogeneous demonstrations with flexible observation and action heads. OpenVLA combined pretrained SigLIP and DINOv2 visual features with a large language backbone and robot action tokens, enabling open fine-tuning studies. These projects shifted robot foundation research from closed demonstrations toward inspectable data and checkpoints.

5.5.3 SmolVLA and accessible adaptation

SmolVLA targets affordable, efficient robot learning. Its importance is system-level: smaller models lower memory, reduce latency, and make asynchronous inference and community hardware practical. A compact VLA fine-tuned on well-aligned target data can be preferable to a much larger model whose rate or memory violates the platform budget.

5.5.4 Flow and two-system models

π_0 introduced a flow-matching action expert conditioned by a pretrained vision-language model. Later systems increasingly separate semantic reasoning from high-rate action. NVIDIA GR00T releases use a vision-language backbone with a diffusion-transformer-style action system; Gemini Robotics combines embodied reasoning and direct action capabilities; humanoid systems such as Helix emphasize whole-body coordination. Exact releases evolve, but the stable pattern is division of labor between broad multimodal understanding and embodiment-specific continuous generation.

5.6 Training objectives beyond imitation

A VLA can combine losses:

$$\mathcal{L} = \lambda_a \mathcal{L}_{\text{action}} + \lambda_l \mathcal{L}_{\text{language}} + \lambda_v \mathcal{L}_{\text{vision}} + \lambda_f \mathcal{L}_{\text{future}} + \lambda_s \mathcal{L}_{\text{safety}}. \quad (5.13)$$

Terms: The λ values balance action, language, visual representation, future-prediction, and safety-related auxiliary objectives. A term is useful only when its labels and gradients align with downstream behavior.

Language and vision replay reduce catastrophic forgetting during robot fine-tuning. Future prediction can improve temporal features but may consume capacity on visually unpredictable detail. Safety losses need careful design: binary safe/unsafe labels do not encode how to recover, and class imbalance can create a model that always predicts safe.

Preference optimization or reinforcement learning can refine behavior after imitation. For expected return

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \gamma^t r(s_t, \mathbf{a}_t) \right], \quad (5.14)$$

Terms: τ is a trajectory; r reward; $\gamma \in [0, 1]$ discount. Physical deployment also needs explicit constraints, because a scalar reward can trade safety against progress.

Constrained optimization writes

$$\max_{\theta} J(\theta) \quad \text{s.t.} \quad J_{c_k}(\theta) = \mathbb{E} \left[\sum_t \gamma^t c_k(s_t, \mathbf{a}_t) \right] \leq d_k, k = 1, \dots, K. \quad (5.15)$$

Terms: c_k is cost for constraint k ; d_k is allowed expected budget. Expected constraints alone do not prevent rare catastrophic events, so runtime controls remain necessary.

5.7 Choosing a VLA architecture

Use a requirements matrix:

Requirement	Favors	Watch
Low latency, unimodal action	regression or small ACT	average action under hidden multimodality
Multimodal contact strategy	diffusion or flow	inference steps and sampling variance
Reuse language backbone	action tokens or conditioned expert	quantization and prior dominance
Long coordinated motion	action chunks	stale commitment and overlap
Many embodiments	shared backbone, embodiment tokens, separate heads	semantic action alignment
Small target data	frozen backbone plus LoRA/head	representation mismatch
Torque/high-rate control	hierarchical model with fast controller	safety, rate, hardware identification

Model size should be chosen after the latency and memory envelope. Peak training memory includes parameters, optimizer states, gradients, activations, image tokens, and action horizon. Inference memory includes key-value caches and concurrency. Quantization can change action quality in subtle dimensions; validate per action component and closed loop.

5.8 Evaluation and the embodiment gap

Offline action loss, closed-loop success, robustness, and physical integrity measure different properties. Use cross-factor evaluation over task g , object o , scene s , embodiment e , and disturbance d :

$$\hat{P}_{\text{succ}}(g, o, s, e, d) = \frac{1}{n_{\text{gosed}}} \sum_{i=1}^{n_{\text{gosed}}} \mathbf{1}\{\text{trial } i \text{ succeeds}\}. \quad (5.16)$$

Terms: n_{gosed} is trial count in the stratum; $\mathbf{1}\{E\}$ is one when event E occurs and zero otherwise. Pool only after inspecting the strata.

Track action saturation, jerk, collision margin, intervention, replans, latency, and recovery. For normalized jerk

$$J_{\text{jerk}} = \frac{1}{T} \int_0^T \|\ddot{\mathbf{p}}(t)\|_2^2 dt, \quad (5.17)$$

where $\mathbf{p}(t)$ is an end-effector or base trajectory. Smoothness is not safety, but extreme jerk exposes unstable action generation or temporal aggregation.

An embodiment gap occurs when a semantically correct plan maps poorly to hardware: different reach, joint limits, gripper geometry, camera placement, latency, or low-level control. Evaluate zero-shot transfer and adapted transfer separately. Report how much target data and compute adaptation used.

ENGINEERING NOTE - Minimal VLA model card

Record visual resolution and camera keys; prompt template; proprioceptive fields; model action order; normalization statistics; physical units and frame; policy and servo rates; horizon and executed prefix; generative solver steps; supported embodiments; training data revisions; latency percentiles; known failure conditions; and required runtime guardrails.



Figure 5.1: A VLA converts multimodal context into an action distribution.

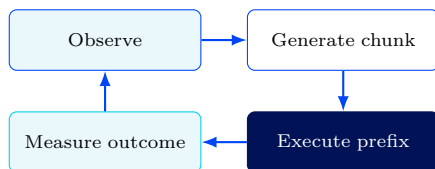


Figure 5.2: Closed-loop VLA execution replans after a verified prefix.

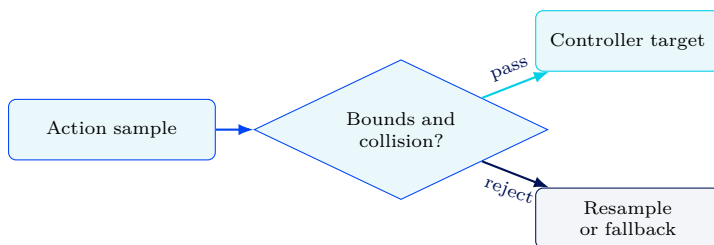


Figure 5.3: A sampled action is a proposal until constraints pass.



Principal references

- [7] Transformer robot control at scale.
- [8] Web-knowledge transfer into a VLA.
- [9] Cross-embodiment data and RT-X.
- [10] Open generalist robot policy.
- [11] Open-source VLA architecture and training.
- [12] Flow-based general robot control.
- [13] Efficient accessible VLA adaptation.
- [27] Diffusion action generation.
- [43] Flow-matching objective.
- [28] Latent action chunks.

LESSON 6

Training and Retraining with Hugging Face and LeRobot



Hugging Face

LEARNING OBJECTIVES

After this lesson you can construct a LeRobot-style dataset, select a policy and adaptation strategy, configure a reproducible training run, use parameter-efficient fine-tuning, diagnose loss and rollout failures, export an inference contract, and integrate TensorFlow components without corrupting action semantics.

6.1 The open training stack

Hugging Face provides model, dataset, and artifact distribution; LeRobot adds robot-learning datasets, policy implementations, hardware interfaces, and training/evaluation workflows. Their real value is not a shorter installation command. It is the possibility of handing another engineer a dataset, a configuration, and a revision that can actually be inspected and reproduced.

Commands age quickly. The workflow should not. At the book cutoff, LeRobot Dataset v3 is the current episode-oriented representation in the project documentation, with video, columnar metadata, statistics, and Hub integration. Pin a tested release or commit and record it in the run manifest. If I could keep only one artifact from a training run, it would be that manifest rather than a screenshot of the loss curve.



6.2 Project specification before installation

Write an interface document containing:

- ▶ robot type, firmware, low-level control mode, and servo rate;
- ▶ policy observation keys, image resolutions, camera order, and proprioception;
- ▶ model action vector, normalization, units, frame, bounds, and policy rate;
- ▶ episode success definition and reset procedure;
- ▶ target latency and hardware memory;
- ▶ dataset license, privacy, and sharing policy;
- ▶ training objective, validation split, closed-loop metrics, and fallback.

Then create a software bill of materials: Python, CUDA, PyTorch, Transformers, LeRobot, PEFT, Accelerate, video codec, robot drivers, and exact revisions. A checkpoint without this environment and interface is incomplete.

6.3 Dataset creation

A LeRobot dataset organizes episodes and features so sequence windows can be sampled without losing temporal metadata. A conceptual feature schema is:

```
features = {
    "observation.images.front": VideoFrame(height=480, width=640),
    "observation.state": FloatVector(names=state_names),
    "action": FloatVector(names=action_names),
    "timestamp": FloatScalar(unit="seconds"),
    "task": Utf8String(),
    "episode_index": IntScalar(),
}
```

The code is illustrative; use the exact feature constructors of the pinned version. The semantics are permanent. Each vector dimension needs a name. Each episode needs a task, success status, and provenance. Dataset statistics must be computed from training episodes only and stored with the revision.

For sampled frame t , a training item may include offsets $\Delta = \{-L+1, \dots, 0, \dots, H-1\}$:

$$\mathcal{B}_t = \{\mathbf{o}_{t+\delta_o} : \delta_o \in \Delta_o, \mathbf{a}_{t+\delta_a} : \delta_a \in \Delta_a\}. \quad (6.1)$$

Terms: Δ_o defines observation history and Δ_a action targets; boundary elements require masks or controlled padding. Offsets are in physical time or sample indices under a documented rate.

Before uploading, run a dataset contract test: decode several videos, verify tensor shapes and ranges, compare timestamps with frame count, check action names and

statistics, visualize actions over video, and ensure episode finalization completes metadata. Do not discover on the robot that the gripper dimension was inverted.

PHYSICAL FAILURE MODE - Preprocessing mismatch

A model trained with one crop, normalization, camera order, or tokenizer can fail under another while every tensor shape remains valid. Version preprocessing as part of the model.



6.4 Choosing the first policy

Begin with the smallest policy that can test the hypothesis.

Policy	Useful first case	Primary engineering cost
MLP/state BC	schema, timing, control sanity	no visual generalization
ACT	modest demonstrations, co-ordinated chunks	CVAE tuning and chunk execution
Diffusion policy	multimodal continuous action	iterative inference
SmolVLA	language-conditioned accessible VLA	image/token memory and pre-trained interface
OpenVLA / larger VLA	broad semantic transfer	GPU memory, latency, action adaptation
Flow VLA	rich continuous chunks	solver and implementation complexity

A baseline ladder isolates defects: state-only BC, image BC without language, language-conditioned compact policy, then foundation VLA. If every model is changed at once, an improvement cannot be attributed.

6.5 Training configuration

For batch $\mathcal{B}$ of size B , gradient accumulation over G microbatches approximates effective batch BG without storing all activations. With data-parallel world size W ,

effective batch is $B_{\text{eff}} = BGW$. Record it because learning-rate schedules often depend on it.

The AdamW update is

$$\mathbf{m}_k = \beta_1 \mathbf{m}_{k-1} + (1 - \beta_1) \mathbf{g}_k, \quad (6.2)$$

$$\mathbf{v}_k = \beta_2 \mathbf{v}_{k-1} + (1 - \beta_2) \mathbf{g}_k \odot \mathbf{g}_k, \quad (6.3)$$

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \alpha_k \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{v}}_k + \epsilon}} - \alpha_k \lambda \boldsymbol{\theta}_k. \quad (6.4)$$

Terms: $\mathbf{g}_k$ is minibatch gradient; $\mathbf{m}, \mathbf{v}$ moment estimates; β_1, β_2 decay factors; hats denote bias correction; $\odot$ elementwise product; λ decoupled weight decay.

Use separate learning rates for pretrained backbone and new action head. A representative ratio is action-head learning rate several times the backbone rate, but tune from validation. Warmup prevents early large updates to pretrained features. Gradient clipping $\mathbf{g} \leftarrow \mathbf{g} \min(1, c/\|\mathbf{g}\|_2)$ limits global norm to c .

Mixed precision reduces memory. BF16 has broader exponent range than FP16 and is generally easier for large transformers on supported hardware. Keep normalization statistics and numerically sensitive reductions in adequate precision. Gradient checkpointing recomputes activations to trade time for memory.

A run configuration should be declarative:

```
seed: 17
dataset_revision: "<immutable revision>"
policy: "smolvla_or_act"
observation_history: 2
action_horizon: 50
execute_horizon: 10
image_resolution: [256, 256]
optimizer: {name: adamw, lr_head: 1e-4, lr_backbone: 1e-5}
precision: bf16
max_steps: 30000
eval_every: 1000
checkpoint_every: 2000
```

Store the resolved configuration, not only command-line overrides.

6.6 Parameter-efficient adaptation

If a pretrained VLA is close to the target domain, LoRA or another PEFT method reduces trainable state. For transformer layer l , the update is $\Delta W_l = (\alpha_l/r_l)B_lA_l$. The number of trainable parameters is

$$N_{\text{LoRA}} = \sum_{l \in S} r_l (d_l^{\text{in}} + d_l^{\text{out}}), \quad (6.5)$$

where $\mathcal{S}$ is the set of adapted matrices. Report $N_{\text{LoRA}}/N_{\text{total}}$.

Adaptation choices:

- ▶ **action head only**: new embodiment with reusable perception and language;
- ▶ **connector plus action head**: different cameras or proprioception;
- ▶ **upper transformer LoRA**: new objects, task phrasing, and action dependencies;
- ▶ **vision LoRA**: strong domain shift in appearance;
- ▶ **full fine-tuning**: large target corpus or severe mismatch with enough compute.

Use target-data validation and a broad capability retention set. A target-only LoRA can overfit few backgrounds while appearing stable in training. Adapter composition is convenient but not algebraically guaranteed to preserve behavior; validate merged and unmerged inference separately.

PHYSICAL FAILURE MODE - Adapter overfit

A target-only adapter can improve one domain while erasing retained capabilities. Evaluate a capability-retention set and compare merged, unmerged, and base checkpoints.



Quantized LoRA stores the frozen backbone at low precision while training adapters. It reduces memory, not necessarily inference latency. Action output is sensitive to small numerical changes near clipping boundaries, so compare physical-unit deltas, saturation, and closed-loop behavior.

EXERCISE 6.1 - LoRA capacity and memory are design variables

For a frozen weight $W \in \mathbb{R}^{4096 \times 4096}$, LoRA with rank $r = 16$ trains $A \in \mathbb{R}^{16 \times 4096}$ and $B \in \mathbb{R}^{4096 \times 16}$. The trainable count is $16(4096 + 4096) = 131,072$, compared with 16,777,216 full parameters: 0.78125%.

Across 32 adapted matrices this is 4.19 million trainable parameters. Optimizer state and activations can still dominate memory, so the percentage is not a GPU-memory promise. Record actual peak allocation, tokens/images/actions per second, and frozen-module hashes [41, 44].



6.7 A reproducible training loop

The framework-independent loop is:

```
for batch in train_loader:
    batch = move_to_device(batch)
    batch = apply_train_only_augmentation(batch)
    prediction = policy(batch)
    loss, metrics = objective(prediction, batch)
    loss.backward()
    clip_grad_norm_(trainable_parameters, max_norm)
    optimizer.step(); scheduler.step(); optimizer.zero_grad()
    log(metrics, data_revision, global_step)
```

Every logged scalar should have an interpretation. Global loss can hide a dead gripper dimension. Track per-camera preprocessing, per-action-component loss in normalized and physical units, sequence-position loss, gradient norm by module, action saturation, throughput, memory, and data-loader stalls.

For action dimensions with different physical consequences, use weighted losses. If normalized labels have covariance Σ_a , a whitened error is

$$\ell_a = (\hat{a} - a)^\top (\Sigma_a + \epsilon I)^{-1} (\hat{a} - a). \quad (6.6)$$

Terms: This balances correlated variation but is not automatically a safety weighting. A small gripper error and small base-speed error can have different risk despite equal standardized scale.

6.8 Validation and checkpoint selection

Do not select by training loss. Use an immutable validation split with coverage of target variations and at least one closed-loop or simulation evaluation at milestones. Composite selection can be

$$J_{\text{select}} = \text{hat}p_{\text{success}} - \lambda_v \hat{p}_{\text{violation}} - \lambda_i \hat{p}_{\text{intervention}} - \lambda_t \max(0, T_{95} - T_{\text{budget}}). \quad (6.7)$$

Terms: $\hat{p}$ terms are empirical rates; T_{95} 95th-percentile inference latency; T_{budget} deadline; λ values express release priorities. Do not use a weighted score to conceal a hard safety requirement.

Maintain hard gates separately: no forbidden collision in a specified test, action schema hash match, latency below limit, model passes smoke test, and fallback verified. The best research checkpoint need not be releasable.

Checkpoint artifacts should include weights/adapters, configuration, dataset revision, normalization statistics, tokenizer/processor revision, action schema, code commit, environment lock, evaluation report, and model card. Compute a content hash for release bundles.

6.9 Inference and asynchronous control

Synchronous inference waits for each chunk. Asynchronous inference allows control to continue while a new chunk is generated. Let τ_o be the observation time used for a proposal, τ_r receive time, and τ_e execution time. Freshness is $F = \tau_e - \tau_o$. Reject or replan when $F > F_{\max}$.

The inference client should send observation ID, timestamps, robot/configuration ID, current mode, previous-action summary, and requested horizon. The response should contain the same IDs, model revision, action chunk, units/schema version, uncertainty if available, and generation latency. Never match responses by arrival order alone.

Real-time chunking commonly executes only a prefix, then blends or replaces it. If new action $\mathbf{a}^{new}$ arrives while old action $\mathbf{a}^{old}$ remains, a blend over K steps is

$$\mathbf{a}_k = (1 - \rho_k)\mathbf{a}_k^{old} + \rho_k\mathbf{a}_k^{new}, \quad \rho_k = \frac{k+1}{K}, \quad k = 0, \dots, K-1. \quad (6.8)$$

Blend only compatible physical spaces, then apply bounds. For contact, averaging can be invalid; a state-aware switch or replanning policy is safer.

6.10 TensorFlow interoperability

Robot-learning policy implementations are often PyTorch-first, while production vision or signal pipelines may use TensorFlow. Interoperate at stable boundaries:

- ▶ exchange NumPy/DLPack tensors with explicit dtype, device, and layout;
- ▶ export isolated components through ONNX or a deployment runtime only after numerical comparison;
- ▶ keep normalization and action decoding in a framework-independent specification;
- ▶ use identical seeded synthetic inputs for golden tests.

For model outputs $Y^{(P)}$ and $Y^{(T)}$ from PyTorch and TensorFlow, report

$$\delta_\infty = \max_i |Y_i^{(P)} - Y_i^{(T)}|, \quad \delta_{rel} = \frac{\|Y^{(P)} - Y^{(T)}\|_2}{\|Y^{(P)}\|_2 + \epsilon}. \quad (6.9)$$

Acceptable tolerance depends on precision and downstream sensitivity. An action decoder test should compare denormalized units and the final clipped command, not only latent logits.

TensorFlow can also train a compact policy using the same episode windows. Matching two frameworks in Module I teaches that data semantics and objectives matter more than API. For a foundation VLA, avoid converting an entire training stack unless there is a deployment requirement; conversion increases the surface for silent mismatch.

6.11 Diagnosis by symptom

Symptom	First checks
Loss does not fall	tiny-subset overfit, masks, label offsets, learning rate, frozen parameters
Good loss, static robot	denormalization, action order, clipping, controller mode, gripper convention
Jitter	action timestamp, chunk overlap, rate mismatch, noisy state, regression multimodality
Language ignored	prompt template, tokenizer, task balance, frozen connector, instruction leakage
Vision ignored	image key order, blank/normalized pixels, task solvable from state, augmentation
Works only in training room	scene leakage, crop/calibration, insufficient diversity, prior dominance
GPU out of memory	image tokens, horizon, batch, precision, gradient checkpointing, LoRA
Slow closed loop	codec, transfer, solver steps, synchronization, warmup, server queue

Use ablations as causal tests: blank language, shuffle images, freeze proprioception, replace goal, delay observations, or run the expert action decoder. A policy that retains performance with shuffled images is not visually grounded, regardless of its architecture name.

6.12 Release workflow

Move through four environments:

1. **offline**: schema tests, unit tests, held-out metrics, perturbations;

2. **simulation/replay**: closed-loop or counterfactual evaluation and monitor tuning;
3. **guarded hardware**: reduced speed/force, restricted workspace, operator stop, detailed logging;
4. **operational trial**: representative workload, shift monitoring, defined rollback.

Promotion requires a signed evaluation artifact and explicit rollback. Hub publication is distribution, not authorization. Private repositories, access controls, and artifact signing may be required for sensitive robots.

ENGINEERING NOTE - The retraining loop

Collect failure and intervention episodes; label their physical cause; add a balanced recovery subset; train an adapter or new checkpoint; replay the previous release test set; evaluate the failure slice and broad retention; validate the run-time monitor; then promote by immutable revision. Do not train directly on every production anomaly without curation.

Recommended practice: complete Exercise 2 in the companion course bundle.



Figure 6.1: A reproducible robot-learning run begins with immutable inputs.

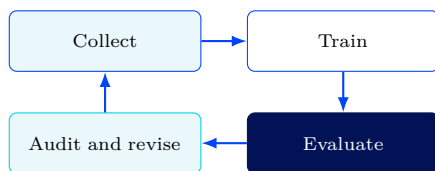


Figure 6.2: Training evidence links data, model, metrics, and release decision.

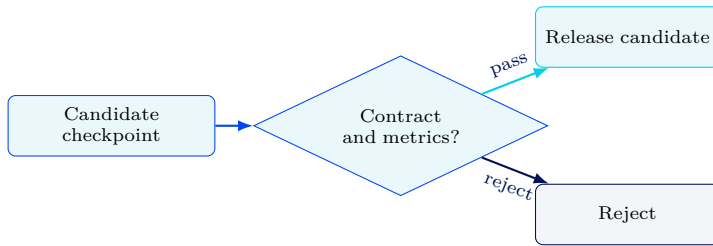


Figure 6.3: *Checkpoint selection uses physical criteria beyond loss.*

Principal references

- [32] LeRobot’s end-to-end open stack.
- [45] Pinned public documentation for the 0.6 release family.
- [13] Compact VLA adaptation.
- [28] ACT baseline and action chunks.
- [41] Low-rank adaptation.
- [44] Quantized parameter-efficient fine-tuning.
- [46] Decoupled weight decay.
- [47] PyTorch system design.
- [48] TensorFlow system design.

MODULE III

Predictive Physical Intelligence



LESSON 7

World Models: Learning Possible Futures

LEARNING OBJECTIVES

After this lesson you can distinguish observation prediction from state prediction; derive deterministic, probabilistic, latent, and recurrent state-space world models; train multi-step models; use learned dynamics for planning; evaluate uncertainty and long-horizon consistency; and identify when visually plausible futures are physically wrong.

7.1 The object being learned

Push the same box twice and the outcomes need not match: the contact point shifts, friction changes, and part of the scene may be hidden. A world model is our attempt to answer a conditional question despite that uncertainty: given what is known now and a candidate action, what may happen next?

The answer can live in physical state, pixels, tokens, occupancy, or a learned representation. Let $\mathbf{x}_t$ be physical state, $\mathbf{o}_t$ observation, $\mathbf{a}_t$ action, and $\mathbf{z}_t$ latent state. Three common targets are

$$p_{\theta}(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t), \quad \text{state dynamics,} \quad (7.1)$$

$$p_{\theta}(\mathbf{o}_{t+1} \mid \mathbf{o}_{\leq t}, \mathbf{a}_t), \quad \text{observation prediction,} \quad (7.2)$$

$$p_{\theta}(\mathbf{z}_{t+1} \mid \mathbf{z}_{\leq t}, \mathbf{a}_t), \quad \text{latent dynamics.} \quad (7.3)$$

These are not equivalent. Pixels include shadows and texture that may be irrelevant to action. A low-dimensional state omits unmeasured contact or intent. A latent can preserve useful predictive structure but has no automatic physical meaning. Choose the target from the decision the model must support.

7.2 Deterministic one-step dynamics

The simplest model predicts

$$\hat{\mathbf{x}}_{t+1} = f_{\theta}(\mathbf{x}_t, \mathbf{a}_t), \quad \mathcal{L}_1 = \frac{1}{N} \sum_t \|W(\hat{\mathbf{x}}_{t+1} - \mathbf{x}_{t+1})\|_2^2. \quad (7.4)$$

Terms: W rescales state dimensions or applies a task metric. A residual form $\hat{\mathbf{x}}_{t+1} = \mathbf{x}_t + \Delta t g_{\theta}(\mathbf{x}_t, \mathbf{a}_t)$ makes time step explicit.

One-step accuracy does not ensure rollout accuracy. Recursive prediction uses

$$\hat{\mathbf{x}}_{t+k+1} = f_{\theta}(\hat{\mathbf{x}}_{t+k}, \mathbf{a}_{t+k}), \quad \hat{\mathbf{x}}_t = \mathbf{x}_t. \quad (7.5)$$

A local bias changes the input distribution at every step. Train with multi-step loss

$$\mathcal{L}_H = \sum_{k=1}^H \lambda_k \|W(\hat{\mathbf{x}}_{t+k} - \mathbf{x}_{t+k})\|_2^2, \quad (7.6)$$

where λ_k weights horizons. Backpropagation through the rollout improves self-consistency but increases memory and can favor over-smooth predictions.

Teacher forcing feeds true $\mathbf{x}_{t+k}$ during training; free rollout feeds predicted state. Scheduled sampling mixes them but changes the statistical objective. A robust workflow reports both one-step and free-rollout errors and tests structured perturbations.

► LANDMARK RESULT 7.1 - Finite-horizon simulation lemma

Consider a true finite-horizon MDP $\mathcal{M}$ and learned model $\widehat{\mathcal{M}}$ with the same state and action spaces, horizon H , and rewards in $[0, R_{\max}]$. Use

$$\text{TV}(P, Q) = \frac{1}{2} \int |dP - dQ|.$$

If, uniformly over stage, state, and action,

$$|r_t - \widehat{r}_t| \leq \epsilon_r, \quad \text{TV}(P_t, \widehat{P}_t) \leq \epsilon_p,$$

then one common finite-horizon form of the simulation lemma gives, for every fixed Markov policy π ,

$$\left| V_{\mathcal{M}}^{\pi}(s) - V_{\widehat{\mathcal{M}}}^{\pi}(s) \right| \leq H\epsilon_r + \frac{H(H-1)}{2} R_{\max} \epsilon_p. \quad (7.7)$$

The bound follows by applying the one-step total-variation error to the remaining value range and recursing backward; it is a transparent finite-horizon specialization of the classical simulation-lemma idea [49].

Engineering meaning. Uniform one-step transition error can create quadratic horizon dependence in the worst case. Replanning from new measurements reduces open-loop exposure, but does not remove bias inside each candidate rollout. The bound is conservative and assumes a shared Markov state, reward scale, action semantics, and uniform error control; it is not a certificate for an ungrounded continuous latent.

Terms: $P_t, \widehat{P}_t$ are true and learned transition kernels; $r_t, \widehat{r}_t$ are rewards; ϵ_p, ϵ_r are uniform errors; V^{π} is finite-horizon value.

7.3 Probabilistic dynamics and aleatoric uncertainty

If the next state is stochastic or unobserved variables matter, predict a distribution. A diagonal Gaussian model gives

$$p_{\theta}(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t) = \mathcal{N}(\boldsymbol{\mu}_{\theta}, \text{diag}(\boldsymbol{\sigma}_{\theta}^2)). \quad (7.8)$$

Its negative log likelihood, omitting constants, is

$$\mathcal{L}_{\text{NLL}} = \frac{1}{2} \sum_j \left[\frac{(x_{t+1,j} - \mu_{\theta,j})^2}{\sigma_{\theta,j}^2} + \log \sigma_{\theta,j}^2 \right]. \quad (7.9)$$

Terms: $\mu_{\theta,j}$ and $\sigma_{\theta,j} > 0$ are predicted mean and standard deviation of state component j . The log term prevents arbitrary variance inflation.

This models conditional noise, not ignorance outside training. An ensemble f_{θ_m} , $m = 1, \dots, M$, gives epistemic dispersion. With member means μ_m and covariances Σ_m , total predictive covariance can be decomposed:

$$\Sigma_{\text{pred}} = \frac{1}{M} \sum_m \Sigma_m + \frac{1}{M} \sum_m (\mu_m - \bar{\mu})(\mu_m - \bar{\mu})^\top. \quad (7.10)$$

Terms: The first term is mean aleatoric covariance; the second is covariance among model means; $\bar{\mu} = M^{-1} \sum_m \mu_m$.

Ensembles are useful for model-predictive control and out-of-distribution caution, but shared data and architecture can make them confidently wrong together.

7.4 Latent variable world models

High-dimensional observations motivate an encoder $q_\phi(\mathbf{z}_t \mid \mathbf{o}_{\leq t}, \mathbf{a}_{< t})$, latent transition $p_\theta(\mathbf{z}_{t+1} \mid \mathbf{z}_t, \mathbf{a}_t)$, and decoder $p_\theta(\mathbf{o}_t \mid \mathbf{z}_t)$. A variational sequence objective is the evidence lower bound:

$$\log p(\mathbf{o}_{1:T} \mid \mathbf{a}_{1:T-1}) \geq \sum_{t=1}^T \mathbb{E}_{q_\phi} [\log p_\theta(\mathbf{o}_t \mid \mathbf{z}_t)] \quad (7.11)$$

$$- \sum_{t=1}^T D_{\text{KL}}(q_\phi(\mathbf{z}_t \mid \cdot) \parallel p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}, \mathbf{a}_{t-1})). \quad (7.12)$$

Terms: The reconstruction term makes latent state explain observations; the KL term aligns posterior state with action-conditioned prior dynamics. The dot denotes available history.

Recurrent State-Space Models separate deterministic memory $\mathbf{h}_t$ and stochastic state $\mathbf{z}_t$:

$$\mathbf{h}_t = f_\theta(\mathbf{h}_{t-1}, \mathbf{z}_{t-1}, \mathbf{a}_{t-1}), \quad (7.13)$$

$$\mathbf{z}_t \sim p_\theta(\mathbf{z}_t \mid \mathbf{h}_t), \quad (7.14)$$

$$\mathbf{o}_t \sim p_\theta(\mathbf{o}_t \mid \mathbf{h}_t, \mathbf{z}_t). \quad (7.15)$$

Terms: $\mathbf{h}_t$ carries deterministic temporal information; $\mathbf{z}_t$ captures stochastic uncertainty. During observation, an encoder posterior conditions on $\mathbf{o}_t$; during imagination, the prior is used.



PlaNet used latent dynamics for planning from pixels. Dreamer learned behavior from imagined latent trajectories, adding reward and continuation predictors. DreamerV3 showed a robust general recipe across domains. The central engineering idea is to plan in a predictive representation cheaper than rendering pixels.

7.5 Learning behavior in imagination

Let the world model generate latent trajectory under policy $\pi_\psi(\mathbf{a}_t \mid \mathbf{s}_t)$, with $\mathbf{s}_t = (\mathbf{h}_t, \mathbf{z}_t)$. An imagined return is

$$G_t = \sum_{k=0}^{H-1} \gamma^k \left(\prod_{j=0}^{k-1} c_{t+j} \right) \hat{r}_{t+k} + \gamma^H \left(\prod_{j=0}^{H-1} c_{t+j} \right) V_\omega(\mathbf{s}_{t+H}). \quad (7.16)$$

Terms: $\hat{r}_t$ is predicted reward; $c_t \in [0, 1]$ predicted continuation; γ discount; V_ω value model; empty product equals one.

The actor maximizes expected imagined return. This is efficient only when the model is accurate where the improving policy goes. Model exploitation occurs when the policy finds actions that produce unrealistically high predicted reward. Ensembles, uncertainty penalties, conservative objectives, short horizons, and real-data refresh reduce but do not eliminate it.

An uncertainty-penalized planning objective is

$$J(A) = \mathbb{E}[\hat{R}(A)] - \beta \sum_{k=1}^H \text{tr} \Sigma_{t+k}(A), \quad (7.17)$$

where $A = \mathbf{a}_{t:t+H-1}$, $\hat{R}$ predicted return, and Σ model uncertainty. β trades progress for uncertainty avoidance. Excessive β makes the planner inert; insufficient β invites exploitation.

7.6 Model-predictive control

MPC repeatedly solves

$$A_t^* = \arg \min_{\mathbf{a}_{t:t+H-1}} \sum_{k=0}^{H-1} \ell(\hat{\mathbf{x}}_{t+k}, \mathbf{a}_{t+k}) + \ell_f(\hat{\mathbf{x}}_{t+H}) \quad (7.18)$$

s.t. learned or hybrid dynamics and constraints.

Only the first action or short prefix is executed, then the problem is re-solved from a new observation.

The Cross-Entropy Method handles non-differentiable cost and complex models:

1. initialize action-sequence distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$;
2. sample N sequences and roll them through the model;
3. select the best K elite sequences;
4. update $\boldsymbol{\mu}$ and Σ from elites;
5. repeat and execute the first action from the best or mean sequence.

For bounded actions, sample in an unconstrained space and transform with tanh, or truncate carefully. Warm-start from the previous solution shifted by one step. Parallel world-model rollouts are ideal for GPUs, but planning latency must remain below the control deadline.

Differentiable shooting instead computes $\nabla_A J$ through the model. It can be fast but follows model gradients into unrealistic regions. Multiple starts and constraints are necessary.

EXERCISE 7.1 - One-step accuracy versus rollout utility

Consider scalar dynamics $x_{t+1} = 1.02x_t + u_t$, while the learned model uses $1.00x_t + u_t$. With $u_t = 0$ and $x_0 = 1$, the true state after 20 steps is $1.02^{20} = 1.486$, whereas the model predicts 1.000. The one-step error at $x = 1$ is only 0.02; the 20-step error is 0.486.

An MPC planner can limit this exposure by replanning from measured state, but only if sensing is timely and the model ranks candidate actions usefully. Ensembles and uncertainty-aware trajectory sampling expose regions without support [50]; they do not guarantee calibrated tail risk.

7.7 Pixels, video, and world foundation models

Video world models model

$$p_{\theta}(I_{t+1:t+H} \mid I_{\leq t}, A, \mathbf{q}, \mathbf{c}). \quad (7.19)$$

Generative transformers or diffusion models can create high-fidelity futures conditioned on actions, maps, or language. GAIA-1 illustrated driving world generation; NVIDIA Cosmos introduced a platform of world foundation models for Physical AI; Genie-class systems emphasized interactive environment generation. These systems are valuable for synthetic data, scenario expansion, representation pretraining, and planning research.

High perceptual quality is not physical accuracy. Standard video metrics such as FVD compare feature distributions, not conservation laws, contact, exact ego motion, or action causality. A model can generate a beautiful future that contradicts the control input. Evaluate action sensitivity:

$$S_a = \mathbb{E}_{A \neq A'} [d(\hat{Y}(I, A), \hat{Y}(I, A'))], \quad (7.20)$$

where d measures future difference. Too little sensitivity means action is ignored; uncontrolled sensitivity can indicate instability. Compare with real counterfactual data or simulation where available.

A world foundation model becomes operational only after domain adaptation and calibration. It may encode broad priors about object permanence and motion yet fail on a specific gripper, tire friction, lens, or process. Freeze broad representation layers and adapt action-conditioned dynamics, or combine learned perception with a calibrated simulator.

7.8 Hybrid and object-centric models

Known physics can be preserved while learning residuals:

$$\mathbf{x}_{t+1} = f_{\text{phys}}(\mathbf{x}_t, \mathbf{a}_t; \eta) + r_\theta(\mathbf{x}_t, \mathbf{a}_t). \quad (7.21)$$

Terms: f_{phys} is an analytical or simulator step with parameters η ; r_θ learns unmodeled effects. Regularize the residual and validate extrapolation.

Object-centric models represent a set $\{\mathbf{z}_t^{(i)}\}_{i=1}^{N_t}$ and interactions. A graph network update is

$$\mathbf{m}_{ij} = \phi_e(\mathbf{z}_t^{(i)}, \mathbf{z}_t^{(j)}, \mathbf{a}), \quad (7.22)$$

$$\mathbf{z}_{t+1}^{(i)} = \phi_v\left(\mathbf{z}_t^{(i)}, \sum_{j \neq i} \mathbf{m}_{ij}\right). \quad (7.23)$$

Terms: ϕ_e computes pairwise messages; ϕ_v updates object i ; aggregation gives permutation invariance. Object discovery and identity tracking are the hard parts in raw video.

Hybrid models are attractive when physical constraints are strong, labeled data limited, and interpretability important. Pure learned models are attractive when perception and contact are too complex to encode. A layered system can use both.

7.9 Consistency and physical evaluation

Evaluate at four levels:

1. **one-step fit**: NLL or state error under observed transitions;
2. **free rollout**: error and calibration across horizon;
3. **counterfactual control**: response to changed actions;
4. **decision quality**: task return, constraint violation, and model exploitation under planning.

For probabilistic state predictions $p(\mathbf{x}_{t+k})$, negative log likelihood and calibration should be horizon-stratified. Root mean squared error is

$$\text{RMSE}(k) = \sqrt{\frac{1}{Nd_x} \sum_{i=1}^N \left\| \hat{\mathbf{x}}_{t+k}^{(i)} - \mathbf{x}_{t+k}^{(i)} \right\|_2^2}. \quad (7.24)$$

Terms: d_x is state dimension; compute in normalized and physical units. A single average hides which coordinates drift.

Temporal compositionality compares direct $h+k$ prediction with composing h and k predictions:

$$\Delta_{h,k}^F = \left\| \hat{\phi}_{h+k}(\mathbf{x}, A_{0:h+k-1}) - \hat{\phi}_k(\hat{\phi}_h(\mathbf{x}, A_{0:h-1}), A_{h:h+k-1}) \right\|. \quad (7.25)$$

An exact flow is compositional; learned direct heads may not be. Large residual reveals an interface inconsistency even when each visual future looks plausible.

PHYSICAL FAILURE MODE - Rollout drift

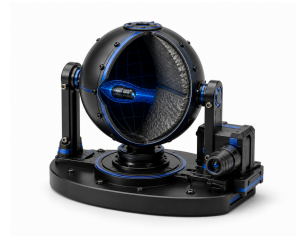
One-step accuracy does not control long-horizon error. Evaluate horizon-specific drift, uncertainty growth, and physical residuals under the action distributions a planner will query.



Physical checks include bounds on speed/acceleration, collision and penetration, reachable-set membership, action-state compatibility, energy/work, and contact mode. They should be evaluated on decoded monitored variables. Passing them is necessary under the selected envelope, not proof of truth.

PHYSICAL FAILURE MODE - Model exploitation

A planner optimizes what the model predicts, not what the world will do. Test candidate actions farther from demonstrations, measure ensemble disagreement, constrain the search distribution, and retain an independent runtime monitor.



ENGINEERING NOTE - World-model acceptance

Require horizon-specific accuracy, calibrated uncertainty, action-counterfactual sensitivity, compositional consistency, physical-residual tests, and improved closed-loop decisions over a non-model baseline. A lower video metric alone is insufficient.

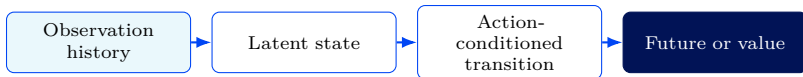


Figure 7.1: A world model separates representation, transition, and observation prediction.

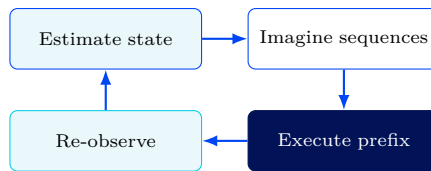


Figure 7.2: MPC repeatedly replaces imagined state with measured state.

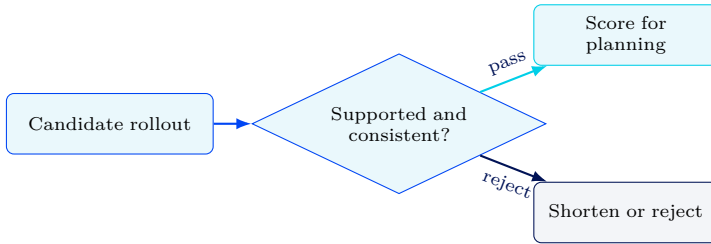


Figure 7.3: *Rollout use should shrink when model support is weak.*

Principal references

- [18] Neural world-model decomposition.
- [51] Latent dynamics for pixel planning.
- [52] Behavior learning in imagination.
- [53] General world-model learning across domains.
- [50] Probabilistic ensembles and trajectory sampling.
- [54] Model-bias-aware rollout design.
- [49] The classical simulation lemma connecting model error to return error.
- [55] Generative driving world model.
- [15] World foundation model platform for Physical AI.
- [56] Data-efficient probabilistic model-based control.



LESSON 8

JEPA and Predictive Representations

LEARNING OBJECTIVES

After this lesson you can derive a joint-embedding predictive objective; explain context, target, and predictor roles; identify representation-collapse controls; distinguish JEPA from reconstruction and generative world modeling; adapt video JEPA features to action-conditioned prediction; and evaluate whether latent prediction supports planning.

8.1 Predict what matters, not every pixel

Future pixels contain vast uncertainty. A hand may occlude an object; specular high-lights change; leaves move; the exact texture of a background does not determine the next safe action. Pixel-generative models are asked to predict all of this. A joint-embedding predictive architecture instead predicts the representation of a target from the representation of context.

Let $\mathbf{x}$ be an image or video. A context operator $C_m(\mathbf{x})$ and target operator $T_m(\mathbf{x})$ select visible and withheld regions or spacetime tubes under mask m . A JEPA uses

$$\mathbf{z}_c = f_\theta(C_m(\mathbf{x})), \quad (8.1)$$

$$\mathbf{z}_t = f_\xi(T_m(\mathbf{x})), \quad (8.2)$$

$$\hat{\mathbf{z}}_t = g_\theta(\mathbf{z}_c, m), \quad (8.3)$$

$$\mathcal{L}_{\text{pred}} = d(\hat{\mathbf{z}}_t, \text{sg}(\mathbf{z}_t)). \quad (8.4)$$

Terms: f_θ is context encoder; f_ξ target encoder; g_θ predictor; m identifies target position; d is embedding distance; sg stops target gradients.

The target is an embedding learned by the system, not a manually designed label. If the embedding discards unpredictable surface detail while retaining objects, motion, and interactions, prediction can concentrate on structure relevant to action.

8.2 Image JEPA

I-JEPA receives a large visible image context and predicts representations of multiple target blocks. Spatially distributed targets prevent the model from solving only local texture completion. A simplified batch objective with M_i targets for image i is

$$\mathcal{L}_{I\text{-JEPA}} = \frac{1}{B} \sum_{i=1}^B \frac{1}{M_i} \sum_{j=1}^{M_i} \|g_{\theta}(z_{c,i}, m_{ij}) - \text{sg}(z_{t,ij})\|_1. \quad (8.5)$$

Terms: B is batch size; m_{ij} describes target block j ; L1 embedding loss is representative.

The target encoder is updated as an exponential moving average of the context encoder:

$$\xi \leftarrow \mu\xi + (1 - \mu)\theta, \quad (8.6)$$

where momentum μ is close to one. This creates a slowly changing target. The predictor must infer target semantics from context rather than match raw pixels.

For Physical AI, image JEPA pretraining can improve object and spatial features without requiring language. It does not learn action or temporal causality. Use it as a visual backbone, not a policy claim.

8.3 Video JEPA

Video extends masking through spacetime. Let video $X \in \mathbb{R}^{T \times H \times W \times C}$ be tokenized into tubes. The context excludes target tubes that may span future frames. Prediction therefore requires temporal representation.

A future-target objective can be written

$$\mathcal{L}_{V\text{-JEPA}} = \sum_{k \in \mathcal{K}} w_k d\left(g_{\theta,k}(f_{\theta}(X_{t-L+1:t})), \text{sg}(f_{\xi}(X_{t+k}^{\text{target}}))\right). \quad (8.7)$$

Terms: $\mathcal{K}$ is a set of future offsets; w_k horizon weights; $g_{\theta,k}$ predicts the target representation at offset k . Target can be a full frame or selected tubes.

V-JEPA learns from video without text or action labels. It can encode motion and object persistence, but observational video confounds cause and correlation. If an agent always turns left before a door appears, the representation can predict the door without understanding the action effect.

8.4 Why representations do not collapse

A trivial solution maps every input to a constant, making prediction loss zero. JEPA systems rely on asymmetry and representation regularity to avoid it: stop-gradient, a momentum target, predictor architecture, mask design, normalization, and diverse data. Some predictive-learning methods add explicit variance and covariance terms.

For batch embeddings $Z \in \mathbb{R}^{B \times d}$, a variance regularizer can require each dimension to retain spread:

$$\mathcal{L}_{\text{var}} = \frac{1}{d} \sum_{j=1}^d \max \left(0, \gamma - \sqrt{\text{Var}(Z_{:,j}) + \epsilon} \right). \quad (8.8)$$

Terms: $\gamma > 0$ is target standard deviation; $Z_{:,j}$ is component j across batch.

A covariance regularizer discourages redundant components:

$$\mathcal{L}_{\text{cov}} = \frac{1}{d} \sum_{i \neq j} \text{Cov}(Z)_{ij}^2. \quad (8.9)$$

These terms are illustrative; I-JEPA's actual recipe relies primarily on architectural asymmetry and target dynamics. Monitor feature variance, effective rank, nearest neighbors, and downstream linear probes during training. Low loss with low-rank constant features is failure.

PHYSICAL FAILURE MODE - Representation collapse

Low predictive loss can coexist with constant or low-rank features. Monitor variance, covariance, effective rank, nearest neighbors, and downstream probes throughout training.



The effective rank of covariance eigenvalues λ_j can be summarized as

$$r_{\text{eff}} = \exp\left(-\sum_j p_j \log p_j\right), \quad p_j = \frac{\lambda_j}{\sum_k \lambda_k}. \quad (8.10)$$

Terms: High r_{eff} indicates information spread across dimensions; it is not proof that information is useful.

EXERCISE 8.1 - Detecting representational collapse

For a batch of N embeddings $z_i \in \mathbb{R}^d$, suppose the mean per-dimension standard deviation falls to 0.03, below a design floor $\gamma = 1$. A VICReg-style variance penalty contributes

$$\frac{1}{d} \sum_{j=1}^d \max(0, \gamma - \text{Std}(z_{\cdot j})) \approx 0.97.$$

The large penalty signals collapse even if predictor loss is small. If standard deviation is healthy but off-diagonal covariance is large, the representation can still be redundant. JEPA target encoders, masking, and regularization must be evaluated together [57, 58].

8.5 JEPA versus autoencoding and generation

An autoencoder minimizes reconstruction $d(D(E(x)), x)$. A masked autoencoder reconstructs missing pixels. A generative world model estimates a distribution over future observations or states. A JEPA predicts target embeddings and need not decode.

Method	Target	Implication
Autoencoder	same input pixels	preserves reconstructable detail
Masked autoencoder	missing pixels	learns context useful for reconstruction
Generative model	future sample/distribution	can visualize alternatives, costly detail
JEPA	learned target embedding	can ignore unpredictable pixel detail
Contrastive model	paired identity relative to negatives	aligns instances/modalities

JEPA does not automatically represent uncertainty. A point embedding predictor can average latent futures. Multiple predictor heads, latent variables, energy functions, or conditional generative heads can represent alternatives. A decoder can be added for visualization, but then decoded fidelity and predictive utility should be assessed separately.

8.6 Action-conditioned JEPA

To use a predictive representation as a world model, condition on action:

$$\hat{\mathbf{z}}_{t+k} = g_{\theta}(\mathbf{z}_t, \mathbf{a}_{t:t+k-1}, k), \quad \mathcal{L}_{A\text{-JEPA}} = \sum_{k=1}^H w_k \|\hat{\mathbf{z}}_{t+k} - \text{sg}(\mathbf{z}_{t+k})\|_2^2. \quad (8.11)$$

Terms: $\mathbf{z}_t = f_{\theta}(X_{\leq t})$; action sequence precedes target; k can be embedded explicitly.

Actions must be synchronized and semantically consistent. If videos from different robots use incompatible actions, condition on embodiment $\mathbf{e}$ or transform to a common physical action. Action-free pretraining can learn broad visual dynamics, followed by action-conditioned post-training on smaller robot data. V-JEPA 2 exemplifies this strategy: large-scale video pretraining, then action-conditioned adaptation for prediction and planning.

An action-effect diagnostic compares predictions under two actions:

$$\Delta_z(A, A') = \|g_{\theta}(\mathbf{z}_t, A) - g_{\theta}(\mathbf{z}_t, A')\|_2. \quad (8.12)$$

If $\Delta_z \approx 0$ for actions known to cause different outcomes, the predictor ignores action. If it is large for physically equivalent actions, the representation is unstable or improperly normalized.

8.7 Planning in embedding space

Suppose a goal observation has embedding $\mathbf{z}_g$. Plan actions by minimizing terminal latent distance plus control cost:

$$A^* = \arg \min_A d_g(\hat{\mathbf{z}}_{t+H}(A), \mathbf{z}_g) + \lambda_a \sum_{k=0}^{H-1} \|\mathbf{a}_{t+k}\|_2^2 + \lambda_u U(A). \quad (8.13)$$

Terms: d_g is a goal metric in representation space; $U(A)$ uncertainty or model-disagreement penalty.

The critical assumption is that latent distance correlates with task progress. A representation trained to identify activity may consider two object poses equivalent even though one is graspable and the other is not. Learn a task-specific metric, add goal/reward supervision, or plan over decoded monitored state.

For goal-conditioned value learning, train $V_\omega(z, z_g)$ from trajectory outcomes. A contrastive temporal loss can make nearby reachable states similar:

$$\mathcal{L}_{\text{temp}} = -\log \frac{\exp(s(z_t, z_{t+k})/\tau)}{\sum_j \exp(s(z_t, z_j)/\tau)}, \quad (8.14)$$

but similarity still conflates reachability direction and physical cost unless actions and time are included.

Gradient-based latent planning differentiates through g_θ ; CEM or sampling avoids local gradients. Test for model exploitation exactly as with generative world models.

8.8 Language and action: VLA-JEPA

VLA-JEPA-style systems combine vision, language, and action with predictive representation learning. A generic multi-task objective is

$$\mathcal{L} = \lambda_z \mathcal{L}_{\text{JEPA}} + \lambda_a \mathcal{L}_{\text{action}} + \lambda_l \mathcal{L}_{\text{language}} + \lambda_c \mathcal{L}_{\text{consistency}}. \quad (8.15)$$

The representation target encourages future structure; the action head learns control; language aligns task semantics; consistency can align direct and rolled predictions.

There are three possible architectures:

- ▶ **shared backbone**: all objectives update common features;
- ▶ **teacher-student**: a predictive encoder teaches the VLA backbone;
- ▶ **two-system**: JEPA world model evaluates or guides a separate action expert.

Shared training is efficient but gradients can conflict. For objective gradients $g_i = \nabla_{\theta} \mathcal{L}_i$, cosine similarity

$$c_{ij} = \frac{g_i^\top g_j}{\|g_i\|_2 \|g_j\|_2} \quad (8.16)$$

reveals conflict when negative. Dynamic loss weighting, gradient surgery, separate adapters, or staged training can help. Do not assume an auxiliary future loss improves policy; measure action and closed-loop outcomes.



8.9 Data and masking design

Masking defines what must be predicted. Tiny local masks permit texture shortcuts. Very large masks can make targets ambiguous and encourage broad semantics. For robot video, useful schemes include:

- ▶ spatial blocks around objects and contact regions;
- ▶ future tubes at multiple horizons;
- ▶ entire camera views to encourage multiview inference;
- ▶ proprioceptive channels or object slots;
- ▶ action-conditioned future windows with preserved timestamps.

Sampling should prevent static frames from dominating. Balance motion magnitude, task, contact, viewpoint, and episode phase. If random web video dominates action-conditioned robot data, the encoder may learn semantic motion yet ignore precise control. Stage pretraining and post-training with explicit evaluation.

Data augmentation must preserve prediction. Independent color jitter on context and target can make color intentionally unpredictable; this may be desirable or destructive depending on task. Independent spatial crop can invalidate correspondence. Record the transformation and target mask in a common coordinate system.

8.10 Evaluation of predictive representations

Use a layered suite:

1. **collapse**: variance, effective rank, nearest-neighbor diversity;
2. **linear probes**: object, state, motion, depth, contact, and task labels;
3. **prediction**: latent error across horizon and action counterfactuals;
4. **physical decoding**: kinematic and dynamic variables predicted from latent state;
5. **planning**: goal success, sample efficiency, constraint violations, and robustness;
6. **transfer**: target performance versus adaptation data and compute.

Latent mean-squared error is representation-dependent: multiplying every feature by a small constant reduces raw loss. Normalize target features or use cosine distance

$$d_{\cos}(\hat{\mathbf{z}}, \mathbf{z}) = 1 - \frac{\hat{\mathbf{z}}^T \mathbf{z}}{\|\hat{\mathbf{z}}\|_2 \|\mathbf{z}\|_2}. \quad (8.17)$$

Even normalized embedding accuracy can be physically irrelevant. A linear probe for end-effector pose, object motion, contact, and collision margin tests whether monitored variables remain recoverable.

To evaluate action conditioning, group initial contexts and apply controlled action alternatives in simulation or repeated trials. Estimate effect-prediction error

$$E_{\text{effect}} = \frac{1}{N} \sum_i \left\| (\hat{\mathbf{y}}_i^A - \hat{\mathbf{y}}_i^{A'}) - (\mathbf{y}_i^A - \mathbf{y}_i^{A'}) \right\|_2, \quad (8.18)$$

where $\mathbf{y}$ is decoded physical outcome. This subtracts common forecast bias and focuses on causal action effect.

8.11 When to use JEPA

JEPA is a strong choice when unlabeled image/video is abundant, pixel detail is partly unpredictable, transfer matters, and planning can operate in or decode from representation space. Pixel generation is preferable when synthetic imagery, likelihood, or interpretable futures are required. Explicit state models are preferable when state is available and safety checks demand direct variables. Hybrid models can use JEPA encoders with state decoders and action-conditioned latent transitions.

PHYSICAL FAILURE MODE - Latent plausibility

A small embedding error does not imply physical admissibility. If the target encoder is invariant to small penetration, joint-limit approach, or gripper force, the predictor will be invariant too. Preserve or decode the variables that the physical monitor needs.



ENGINEERING NOTE - Action-conditioned JEPA acceptance

Require noncollapsed representations, useful physical probes, correct counterfactual action effects, horizon calibration, planning improvement, and independent constraint monitoring. Report all six; a linear-probe score alone is not predictive control.



Figure 8.1: *JEPA predicts target representations rather than every target pixel.*

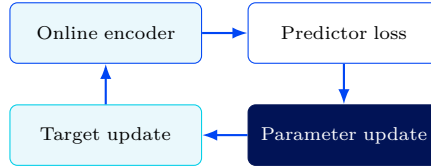


Figure 8.2: *A target network stabilizes self-supervised prediction.*

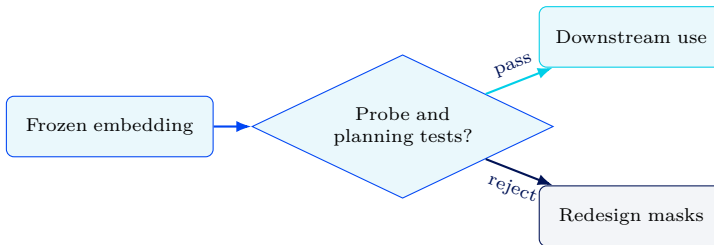


Figure 8.3: *A representation is useful only if it preserves task variables.*

Principal references

- [19] The JEPA research program.
- [57] Image joint-embedding predictive learning.
- [59] Latent video prediction.
- [14] Understanding, prediction, and planning with video JEPA.
- [60] Dense video self-supervised features.
- [61] VLA enhancement with latent world modeling.
- [58] Variance, invariance, and covariance regularization.
- [62] Masked visual representation learning.
- [63] Target-network self-supervision.

LESSON 9

World-Action Models and Planning

LEARNING OBJECTIVES

After this lesson you can formalize a world-action model; distinguish cascaded and joint WAMs from VLAs and world models; derive forward, inverse, and joint objectives; plan by sampling, optimization, or generative action inference; evaluate coupled futures; and construct a practical WAM around a pretrained perception and action stack.

9.1 Why join world and action?

A conventional world model predicts a future given an action. A conventional VLA predicts an action given observation and language. Planning uses both directions: which futures follow an action, and which actions can produce a desired future? A world-action model makes this coupling first-class.

Let $\mathbf{z}_{\leq t}$ be world history, $\mathbf{q}$ task, $A = \mathbf{a}_{t:t+H-1}$ action sequence, and $Z^+ = \mathbf{z}_{t+1:t+H}$ future sequence. A WAM represents

$$p_{\theta}(Z^+, A \mid \mathbf{z}_{\leq t}, \mathbf{q}). \quad (9.1)$$

This joint can be factorized in two useful orders:

$$p(Z^+, A \mid C) = p_{\phi}(Z^+ \mid C, A) \pi_{\psi}(A \mid C), \quad (9.2)$$

$$p(Z^+, A \mid C) = \rho_{\psi}(A \mid C, Z^+) p_{\phi}(Z^+ \mid C), \quad (9.3)$$

Terms: $C = (\mathbf{z}_{\leq t}, \mathbf{q})$; $p_{\phi}(Z^+ \mid C, A)$ is forward dynamics; $\pi_{\psi}(A \mid C)$ action prior/policy; $\rho_{\psi}(A \mid C, Z^+)$ inverse action model; $p_{\phi}(Z^+ \mid C)$ future prior.



The first factorization proposes actions and predicts consequences. The second proposes futures and infers actions to realize them. A fully joint transformer or diffusion/flow model can interleave state and action variables without explicitly choosing one order, but its conditional queries still correspond to these operations.

9.2 A precise boundary with neighboring terms

I use WAM in a deliberately narrow sense: future-world variables and actions must be explicit training or inference targets. This avoids turning the term into a fashionable synonym for any VLA that contains temporal context. The distinctions are:

- ▶ a **VLA** maps current multimodal context to action; it can have no queryable future;
- ▶ a **world model** maps context and action to future; it can have no action generator;
- ▶ a **WAM** models both and supports at least one coupled operation such as action-conditioned future, future-conditioned action, joint sampling, or future-based action selection;
- ▶ a **JEPA WAM** may use latent future representations rather than pixels;
- ▶ a **robot foundation WAM** is pretrained broadly enough for reuse across tasks or embodiments.

An implementation can be modular. A pretrained VLA plus an independently trained action-conditioned video model and planner is a cascaded WAM system when trained/evaluated as a coupled future-action interface. It need not be one neural network.

9.3 Forward and inverse dynamics

A forward model predicts outcome:

$$\hat{\mathbf{z}}_{t+1} = f_{\phi}(\mathbf{z}_t, \mathbf{a}_t), \quad (9.4)$$

while an inverse model predicts action:

$$\hat{\mathbf{a}}_t = g_{\psi}(\mathbf{z}_t, \mathbf{z}_{t+1}). \quad (9.5)$$

Their supervised losses are

$$\mathcal{L}_F = d_z(f_{\phi}(\mathbf{z}_t, \mathbf{a}_t), \mathbf{z}_{t+1}), \quad \mathcal{L}_I = d_a(g_{\psi}(\mathbf{z}_t, \mathbf{z}_{t+1}), \mathbf{a}_t). \quad (9.6)$$

Inverse dynamics is often multimodal: many joint trajectories reach the same object displacement. A deterministic inverse model averages them. Use a probabilistic, diffusion, or flow action model conditioned on current and desired future.

Forward-inverse consistency adds

$$\mathcal{L}_{\text{cycle}} = d_z(f_\phi(z_t, g_\psi(z_t, z_{t+1})), z_{t+1}). \quad (9.7)$$

This tests whether the inferred action reproduces the target under the learned forward model. It can still be jointly wrong: the inverse and forward models can collude outside data. Anchor each to real transitions and independent physical metrics.

For horizon H , a model can predict all states directly or roll recursively. Direct prediction captures global coherence; recursive prediction is naturally compositional. Combine them with consistency:

$$\mathcal{L}_{\text{dir/roll}} = \sum_{k=1}^H \|\hat{z}_{t+k}^{\text{direct}} - \hat{z}_{t+k}^{\text{roll}}\|_2^2. \quad (9.8)$$

9.4 Cascaded WAM architecture

A practical cascaded WAM has four modules:

1. **context encoder** $E(I_{\leq t}, r_{\leq t}, q) = z_t$;
2. **action proposer** $\pi_\psi(A \mid z_t, q)$;
3. **future predictor** $p_\phi(Z^+ \mid z_t, A)$;
4. **selector** scoring task progress, uncertainty, and constraints.

Sample N candidate chunks $A^{(i)}$, predict futures, and choose

$$i^* = \arg \min_i \left[J_{\text{task}}(Z^{+(i)}, A^{(i)}) + \beta U(Z^{+(i)}, A^{(i)}) + \sum_k \lambda_k C_k(Z^{+(i)}, A^{(i)}) \right]. \quad (9.9)$$

Terms: J_{task} is predicted cost; U uncertainty; $C_k \geq 0$ soft constraint penalty. Hard constraints should reject candidates rather than be traded by finite λ_k .

This architecture can wrap an existing VLA without retraining it. It provides inspectable futures and reranking. Its limitation is proposal coverage: if the VLA never proposes a needed recovery, the world model cannot select it. Add exploratory proposals, planner optimization, or an inverse model.



EXERCISE 9.1 - A WAM latency budget

A cascaded WAM generates $M = 8$ candidate futures in 70 ms, scores them in 2 ms each, and decodes the selected action in 18 ms. Sequential scoring yields $70 + 16 + 18 = 104$ ms. For a 10 Hz policy, this exceeds the 100 ms deadline before communication or monitoring.

Batching all scores into 5 ms lowers the model path to 93 ms, but leaves little systems margin. A Fast-WAM-style design can omit explicit test-time imagination when the training-time representation carries most of the benefit [64]; the decision must be made by closed-loop performance and latency, not visual appeal.

9.5 Joint sequence models

A joint transformer can interleave state and action tokens:

$$S = [z_t, \mathbf{a}_t, z_{t+1}, \mathbf{a}_{t+1}, \dots, z_{t+H}]. \quad (9.10)$$

With causal likelihood,

$$\mathcal{L}_{\text{joint}} = - \sum_j \log p_{\theta}(S_j \mid S_{<j}, C). \quad (9.11)$$

Masking allows different queries: mask future states to forecast under supplied actions; mask actions to infer control toward supplied future; mask both to sample a trajectory-action pair. Continuous variables may be quantized or predicted by distribution heads.

A joint diffusion model treats $Y = [Z^+, A]$ as continuous trajectory and denoises it:

$$\mathcal{L}_{\text{joint diff}} = E_{Y^0, k, \epsilon} \left[\left\| \epsilon - \epsilon_{\theta}(Y^k, k, C, M) \right\|_2^2 \right], \quad (9.12)$$

where mask M marks known conditions. Classifier-free or energy guidance can bias samples toward task goals. Inpainting known states or actions makes the same model serve planning and prediction.

Joint modeling captures correlation but increases scale and conditioning difficulty. Pixel-video tokens can dwarf action dimensions; losses need balancing. A model can improve video likelihood while ignoring action. Measure conditional mutual influence, not only pooled loss.

PHYSICAL FAILURE MODE - Joint hallucination

A WAM can generate an action and future that agree with each other yet disagree with the plant. Require independent state, dynamics, and runtime evidence before execution.



9.6 World-language-action models

Language supplies goals and abstractions. A world-language-action model may condition future and action on instruction, produce language rationales, or treat language as an intermediate plan. Hierarchically:

$$\mathbf{h}_{1:K} \sim p_{\theta}(\mathbf{h}_{1:K} \mid I_{\leq t}, \mathbf{q}), \quad (9.13)$$

$$A_k \sim \pi_{\psi}(A_k \mid \mathbf{z}_{t_k}, \mathbf{h}_k), \quad (9.14)$$

$$Z_k^+ \sim p_{\phi}(Z_k^+ \mid \mathbf{z}_{t_k}, A_k). \quad (9.15)$$

Terms: $\mathbf{h}_k$ is a high-level subgoal or skill; each has a lower-level action chunk and predicted result.

Language decompositions improve long horizons and observability but can hallucinate prerequisites. Ground every subgoal to a measurable condition and replan when it fails. The system should not execute step 3 because the language plan said step 2 was completed; it should check the world.

9.7 Generative action planning

Instead of sampling only from a policy, optimize action in the generative space. For a flow action model, initial noise ξ maps to action $A = G_{\theta}(\xi, C)$. Plan over ξ :

$$\xi^* = \arg \min_{\xi} J(p_{\phi}(Z^+ \mid C, G_{\theta}(\xi, C)), G_{\theta}(\xi, C)). \quad (9.16)$$

This keeps actions near the learned manifold while allowing task optimization. Gradient backpropagation through the generator and world model is possible. Sampling/CEM in ξ is robust to nondifferentiable constraints.



Goal guidance for diffusion can modify score or noise prediction using cost gradient:

$$\tilde{\epsilon}_\theta = \epsilon_\theta + s \sigma_k \nabla_{A^k} J(\hat{Z}^+(A^k)), \quad (9.17)$$

Terms: s is guidance strength; σ_k noise scale; sign depends on parameterization. Excessive guidance moves samples off the learned action manifold.

Maintain action bounds during every denoising or flow step. Post-hoc clipping can transform a coherent sequence into a different, dynamically inconsistent one.

9.8 Planning with learned terminal and constraint models

The task score can combine goal embedding, value, reward, and geometry:

$$J_{\text{task}} = \lambda_g d(\mathbf{z}_{t+H}, \mathbf{z}_g) - \lambda_V V_\omega(\mathbf{z}_{t+H}, \mathbf{q}) - \sum_{k=0}^{H-1} \gamma^k \hat{r}_{t+k}. \quad (9.18)$$

Terms: $\lambda_g, \lambda_V \geq 0$ weight terminal goal distance and value; d is the declared distance; $\mathbf{z}_g$ is the goal embedding; V_ω is a learned terminal-value model conditioned on task context $\mathbf{q}$; $\gamma \in [0, 1]$ is the discount factor; and $\hat{r}_{t+k}$ is predicted reward.

Learned rewards are themselves models and can be exploited. Use held-out human preference tests, task-state measures, and hard physical constraints.

For probabilistic collision event C , a chance constraint is

$$\mathbb{P}(C_k \mid C, A) \leq \delta_k, \quad k = 1, \dots, H. \quad (9.19)$$

Terms: δ_k is allowed probability under a calibrated predictive distribution. If probabilities are uncalibrated or omit model error, the inequality is not a safety guarantee.

Risk-sensitive planning uses Conditional Value at Risk. For random cost L ,

$$\text{CVaR}_\alpha(L) = \min_{\eta} \left[\eta + \frac{1}{1-\alpha} \mathbb{E}[(L - \eta)_+] \right], \quad (9.20)$$

where $(x)_+ = \max(x, 0)$ and α is a high quantile. CVaR emphasizes the worst $(1-\alpha)$ tail, but remains only as accurate as modeled uncertainty.

9.9 Frontier WAM systems

The WAM label consolidated in 2026 around models that connect generative futures with robot action. Several prior systems already embodied pieces of the idea. IRASim learned action-conditioned robot video generation. Dynamic World Action Models (DyWA) coupled action and future modeling. Fast-WAM explored efficient robot policies aided by video co-training while reducing future generation cost at inference. DAWN framed world-action pretraining for embodied agents. Contemporary world-language-action and embodied-brain surveys formalized the broader roadmap.

These systems vary: some generate videos for training but output action at inference; some use world prediction as auxiliary representation; some jointly generate action and future; some deploy future imagination for selection. The taxonomy should report which operation is available:

- ▶ *F*: action-conditioned future prediction;
- ▶ *I*: future-conditioned action inference;
- ▶ *J*: joint future-action generation;
- ▶ *P*: online planning or reranking;
- ▶ *D*: future prediction used only as a training objective.

Calling all of them WAMs is useful only if the operation flags remain visible.

9.10 Evaluation for coupled models

Evaluate each conditional direction and the decision loop:

1. forward future accuracy and calibration under logged actions;
2. inverse action likelihood or success for desired reachable futures;
3. joint sample coherence between action and future;
4. counterfactual sensitivity to changed action and goal;
5. planning success, compute, latency, risk, and fallback;
6. physical residuals for predicted states and selected actions.

Cycle error is useful but insufficient. Compare predicted and real action effect:

$$E_{AF} = \frac{1}{N} \sum_i d\left(\Delta \hat{\mathbf{y}}^{(i)}(A^{(i)}), \Delta \mathbf{y}_{\text{real}}^{(i)}(A^{(i)})\right). \quad (9.21)$$

Use physical coordinates such as object displacement, base pose, contact state, or minimum distance. Video metrics are secondary.

For joint samples $(Z^{+(i)}, A^{(i)})$, an independently trained or analytical one-step checker $r(Z^+, A)$ measures action-state compatibility. Avoid evaluating only with the same forward model used to generate them.

Measure planning regret relative to oracle or simulator where possible:

$$\text{Regret} = J_{\text{real}}(A_{\text{WAM}}) - J_{\text{real}}(A_{\text{oracle}}). \quad (9.22)$$

Separate model error, proposal failure, and selector failure by checking whether any sampled candidate was good and whether it was selected.

9.11 Build versus adopt

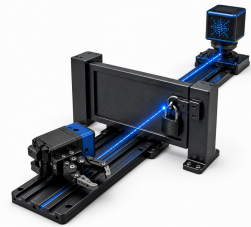
Build a compact WAM when target state/action data are available, domain physics are specific, and latency matters. Adapt a world foundation model when scene diversity and synthetic future generation dominate. Wrap an existing VLA with a future predictor when a deployed policy needs diagnosis and reranking. Train jointly only when data and compute support balanced objectives and the use case benefits from shared representation.

A staged build is:

1. freeze a strong visual/robot encoder;
2. train a state or latent action-conditioned predictor;
3. verify action counterfactuals and free rollouts;
4. connect an existing policy and rerank candidate chunks;
5. add inverse or joint generation only when proposal coverage limits performance;
6. place an independent physical monitor after selection.

PHYSICAL FAILURE MODE - Prediction is not authorization

The WAM can compare proposals, but it is trained from finite data and may jointly hallucinate a coherent action and future. Execute only after the selected action passes hardware bounds, timing, collision, and runtime physical checks.



Recommended practice: complete Exercise 3 in the companion course bundle.

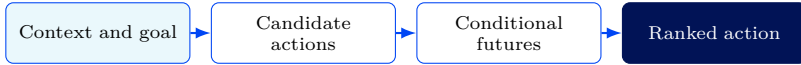


Figure 9.1: A cascaded WAM turns candidate interventions into predicted consequences.

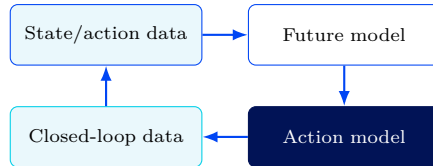


Figure 9.2: World-action learning couples forward prediction and inverse control.

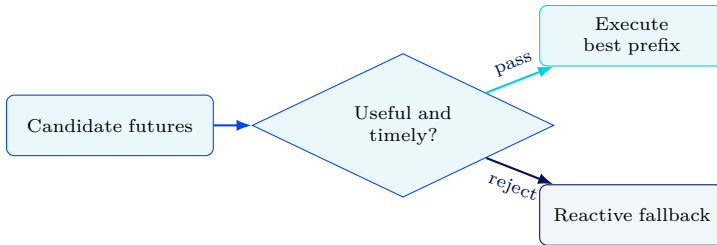


Figure 9.3: Predicted futures must support a decision under a deadline.

Principal references

- [16] Systematic WAM definition and taxonomy.
- [17] WAM-to-embodied-brain systems roadmap.
- [65] Action-conditioned world-model design space.
- [66] Interactive real-robot action simulation.
- [67] Dynamic WAM for manipulation.
- [64] Evidence separating training-time prediction from test-time imagination.
- [68] World-action pretraining.
- [69] World-language-action modeling.
- [32] Open robot-learning stack for implementation.

MODULE IV

Reliable Physical AI Systems

LESSON 10

Simulation, Digital Twins, and Sim-to-Real

LEARNING OBJECTIVES

After this lesson you can distinguish simulator, learned world model, and digital twin; specify a simulation validity envelope; derive domain randomization and system-identification objectives; combine synthetic and real data; design sim-to-real evaluation; and choose among physics, rendering, replay, and generative simulation.

10.1 Three virtual worlds

A simulator, world model, and digital twin can all produce futures, but their authority differs.

- ▶ A **simulator** executes encoded geometry, dynamics, sensors, agents, and rules under parameters η .
- ▶ A **learned world model** estimates future distributions from data under learned parameters θ .
- ▶ A **digital twin** maintains correspondence to a specific physical asset or process through identity, configuration, and data synchronization.

Let simulator transition be $\mathbf{x}_{t+1} = f_{\text{sim}}(\mathbf{x}_t, \mathbf{a}_t; \eta)$. A twin estimates time-varying parameters and state for asset j :

$$(\hat{\mathbf{x}}_t^{(j)}, \hat{\eta}_t^{(j)}) = \mathcal{U}(\hat{\mathbf{x}}_{t-1}^{(j)}, \hat{\eta}_{t-1}^{(j)}, \mathbf{o}_t^{(j)}, \mathbf{u}_{t-1}^{(j)}). \quad (10.1)$$

Terms: $\mathcal{U}$ is a synchronization/update procedure; superscript j ties the representation to a particular asset. Without this maintained identity and update, a configurable simulator is not a twin.



Hybrid simulation uses analytical dynamics for known structure, learned residuals for friction/contact, neural rendering for appearance, and real logs for agents. The right blend follows the validation question.

10.2 What simulation is for

Simulation serves different purposes:

1. **training**: generate interactions, labels, failures, and curriculum;
2. **evaluation**: repeat controlled scenarios and counterfactual actions;
3. **systems test**: verify timing, message contracts, fallback, and orchestration;
4. **design**: choose sensors, workspace, controller, and safety margins;
5. **digital twin operation**: monitor and forecast a particular system.

A simulator valid for camera software integration may be invalid for contact-policy training. Document the **validity envelope**: variables, ranges, temporal resolution, calibrated metrics, known mismatches, and intended decisions. “Photorealistic” is not a validity claim.

10.3 The reality gap

Let $P_s(\tau)$ and $P_r(\tau)$ be simulated and real trajectory distributions. The reality gap is not one number; it appears across observation, transition, action, initial-state, and task distributions:

$$p_s(o | x) \neq p_r(o | x), \quad (10.2)$$

$$p_s(x' | x, a) \neq p_r(x' | x, a), \quad (10.3)$$

$$p_s(x_0) \neq p_r(x_0), \quad p_s(g) \neq p_r(g). \quad (10.4)$$

Rendering mismatch includes lighting, material, blur, lens distortion, exposure, and clutter. Dynamics mismatch includes mass, friction, compliance, backlash, latency, actuator limits, and contacts. Procedural scenarios can also lack human behavior and rare but important failures.

The transfer gap for policy π can be reported as

$$\Delta_J(\pi) = J_r(\pi) - J_s(\pi), \quad (10.5)$$

where J is a defined return or success measure. A single gap hides causes; measure observation and transition residuals directly.

10.4 System identification

System identification estimates simulator parameters from real data. Given transitions $\mathcal{D}_r = \{\mathbf{x}_t, \mathbf{a}_t, \mathbf{x}_{t+1}\}$, estimate

$$\hat{\eta} = \arg \min_{\eta} \sum_t \|W(f_{\text{sim}}(\mathbf{x}_t, \mathbf{a}_t; \eta) - \mathbf{x}_{t+1})\|_2^2 + \lambda R(\eta). \quad (10.6)$$

Terms: W weights measured variables; R encodes plausible parameter prior; η can include mass, friction, damping, delays, or controller gains.

Parameters can be unidentifiable: multiple combinations produce the same observed trajectory. Excitation matters. Design safe actions that illuminate relevant dynamics, then report uncertainty rather than a single best value. Bayesian identification estimates $p(\eta \mid \mathcal{D}_r)$, which can become a domain-randomization distribution.

If the simulator is nondifferentiable, use derivative-free optimization, approximate Bayesian computation, or train a surrogate. Do not fit every parameter simultaneously from narrow trajectories; use hierarchical calibration: sensor intrinsics/extrinsics, timing, free-space actuation, then contact.

10.5 Domain randomization

Domain randomization trains over simulator parameters $\eta \sim p(\eta)$:

$$\theta^* = \arg \min_{\theta} \mathbb{E}_{\eta \sim p(\eta), \tau \sim P_s(\cdot \mid \eta, \pi_{\theta})} [\mathcal{L}(\pi_{\theta}; \tau)]. \quad (10.7)$$

Terms: $p(\eta)$ covers visual, dynamical, timing, and scenario variation; training can be imitation or reinforcement learning.

The goal is not maximum randomness. It is a distribution wide enough that real behavior is not a special fragile point, while preserving learnable structure. Excessive independent randomization creates impossible worlds and a conservative policy. Model correlations: mass and inertia, light and exposure, tire friction and braking response.

Uniform randomization $\eta_j \sim \mathcal{U}(l_j, u_j)$ is easy but arbitrary. Fit distributions from measurement, include a central nominal mass, and stress tails separately. Adaptive domain randomization changes $p(\eta)$ based on learner performance or real discrepancy. A minimax robust objective is

$$\min_{\theta} \max_{\eta \in \mathcal{E}} \mathcal{L}(\pi_{\theta}; \eta), \quad (10.8)$$



where $\mathcal{E}$ is a plausible uncertainty set. Worst-case training can sacrifice typical performance; distributional and risk-sensitive objectives offer a continuum.

EXERCISE 10.1 - A domain-randomization interval from measured uncertainty

Suppose joint friction estimates from hardware have mean 0.18 and standard deviation 0.03. A first design randomizes over $[0.09, 0.27]$, the mean plus or minus three standard deviations. If 7 of 200 validation measurements lie outside, empirical coverage is 96.5%, not 99.7%; the Gaussian interpretation is not supported.

Expanding the interval can improve coverage but also make training unnecessarily difficult. Stratify transfer by parameter, identify which mismatch changes the optimal action, and update the posterior from hardware-in-the-loop evidence. Domain randomization is an experimental distribution, not proof that the real world is contained [70, 71].

10.6 Representation and feature alignment

Sim-to-real can reduce observation gap by randomizing pixels, translating images, or learning domain-invariant features. Let feature encoder f_θ and domain discriminator D_ϕ operate on simulated and real observations. Adversarial alignment solves

$$\min_{\theta} \max_{\phi} \mathbb{E}_{o_r} \log D_\phi(f_\theta(o_r)) + \mathbb{E}_{o_s} \log(1 - D_\phi(f_\theta(o_s))). \quad (10.9)$$

The encoder tries to hide domain identity. Domain invariance can also remove task-relevant distinctions if real and simulated labels differ. Validate physical probes and task performance.

Image translation can make synthetic data look real or real data look synthetic. Preserve geometry, labels, safety colors, and small objects. Cycle consistency does not guarantee semantic preservation. Foundation visual encoders often reduce texture sensitivity, but camera and spatial grounding still need calibration.

An alternative is feature-level randomization and self-supervised pretraining on unlabeled real video. Fine-tune policy on simulation while exposing the encoder to both domains.

10.7 Synthetic demonstrations and labels

Simulation provides exact segmentation, depth, flow, pose, contact, force, collision, and failure labels. Use them to pretrain perception and world models. A synthetic

episode should store the same schema as real data plus simulator parameters and random seed. This allows stratified analysis and replay.

For mixed data, minimize

$$\mathcal{L} = \lambda_r \mathbb{E}_{\mathcal{D}_r} \ell_r + \lambda_s \mathbb{E}_{\mathcal{D}_s} w(\eta, \tau) \ell_s. \quad (10.10)$$

Terms: λ_r, λ_s balance real and synthetic samples; w downweights low-fidelity or out-of-envelope synthetic examples.

Sampling ratio matters more than total counts. If synthetic samples dominate, the policy may learn simulator artifacts. Stage training: broad synthetic pretraining, mixed adaptation, and real-only calibration. Maintain a real validation set never used for parameter tuning.

Synthetic language also needs curation. Templated instructions can leak task identity through wording; large language models can add diversity but may describe states absent from the scene. Bind generated text to simulator state and validate referred entities.

10.8 Generative world models as simulators

World foundation models can generate video futures, expand scenarios, or provide learned sensor simulation. They cover appearance and semantics that rigid engines struggle with. Their limitations are control fidelity, identity persistence, exact geometry, and rare-event correctness.

A hybrid rollout can use physical state from simulator and generated images from neural renderer:

$$\mathbf{x}_{t+1} = f_{\text{sim}}(\mathbf{x}_t, \mathbf{a}_t; \eta), \quad (10.11)$$

$$I_{t+1} \sim p_\theta(I_{t+1} \mid \mathbf{x}_{t+1}, I_{\leq t}, \mathbf{c}). \quad (10.12)$$

This preserves controllable state while enriching perception. The renderer must not contradict state. Validate object count, pose projection, traffic signals, contact, and temporal identity.

Generative simulation is especially useful for scenario ideation. Treat generated episodes as hypotheses that must be reconstructed in a physical or validated simulator before safety conclusions.

10.9 Digital twins

A useful twin includes:

- ▶ asset identity and configuration management;
- ▶ synchronized state estimate and uncertainty;
- ▶ calibrated dynamics and sensor models;
- ▶ event and maintenance history;
- ▶ bidirectional but authorized data interfaces;
- ▶ validation metrics and staleness indicators.

Twin state is never perfectly current. Let last synchronized time be τ_s ; staleness at query τ_q is $S = \tau_q - \tau_s$. A twin prediction should include S and uncertainty. If staleness exceeds a mode-specific limit, downgrade or reject decisions.

Parameter drift can be monitored with residual

$$\mathbf{r}_t = \mathbf{o}_t - h_{\text{twin}}(\hat{\mathbf{x}}_t, \hat{\boldsymbol{\eta}}_t), \quad (10.13)$$

and change statistics over $\mathbf{r}_t$. Residual growth can indicate wear, sensor shift, load change, or model failure. Avoid immediate autonomous parameter updates from anomalous data; use bounded estimators and review.

For policy deployment, a twin can run shadow rollouts of candidate actions. This is useful only if the twin's applicable envelope covers the current mode. Attach a confidence level and independent runtime checks.

PHYSICAL FAILURE MODE - Stale digital twin

A digital twin that is not synchronized to identity, state, configuration, and time is a stale simulator. Record update age and invalidate decisions outside the maintained envelope.



10.10 Simulation platforms and selection

Manipulation and embodied research commonly use MuJoCo, Isaac Lab/Sim, Man-iSkill, Habitat, and task-specific environments; CARLA is prominent in autonomous

driving. Choose by physics and sensor needs, determinism, asset pipeline, GPU parallelism, controller integration, licensing, reproducibility, and hardware-in-the-loop support.

Do not choose a simulator because it appears most realistic in a screenshot. A simpler engine with validated contact and exact robot controller can be superior for action learning. Use multiple simulators for critical claims when feasible; shared success reduces dependence on one engine's artifacts.

10.11 A sim-to-real experimental ladder

1. validate simulator state/action units against analytical cases;
2. replay real open-loop actions and measure state residuals;
3. calibrate parameters on a training set and verify on distinct trajectories;
4. train baseline in nominal simulation;
5. add measured domain randomization and compare;
6. evaluate on held-out simulation corners and real guarded trials;
7. fine-tune with limited real data and report the adaptation curve;
8. test faults, latency, and recovery before normal-speed operation.

Transfer efficiency can be plotted as real performance $J_r(n)$ versus number n of real episodes. Report zero-shot $J_r(0)$, area under the adaptation curve, and final performance. This distinguishes a model that truly transfers from one that succeeds only after substantial real retraining.

To attribute improvements, keep simulator, data budget, architecture, and evaluation seeds fixed. Randomization plus adaptation can interact; use factorial ablations.

10.12 Scenario coverage and rare events

Random seeds are not a coverage argument. Define a scenario vector ξ containing task, geometry, material, initial state, agent behavior, sensor condition, latency, and fault variables. Partition the intended operational envelope into meaningful cells $\{C_j\}_{j=1}^J$. Empirical coverage is

$$C_{\text{scenario}} = \frac{\sum_{j=1}^J w_j \mathbf{1}\{n_j \geq n_j^{\min}\}}{\sum_{j=1}^J w_j}, \quad (10.14)$$

Terms: n_j is evaluated trials in cell j ; $n_j^{\min}$ the required count; w_j criticality or operational weight. The cell definition and uncovered mass are more informative than the scalar alone.

Use combinatorial coverage for categorical factors and boundary-value analysis for continuous ones. Cover nominal centers, individual extremes, and selected interactions such as low friction plus high latency. Full Cartesian enumeration is usually impossible; pairwise or risk-guided designs give tractable structure. Maintain a scenario manifest with parameters and seeds so every failure can be replayed.

Rare-event evaluation needs biased sampling. If target scenario density is $p(\xi)$ but trials are sampled from proposal $q(\xi)$, an importance-sampling estimator of failure probability is

$$\hat{p}_F = \frac{1}{N} \sum_{i=1}^N \frac{p(\xi_i)}{q(\xi_i)} \mathbf{1}_{\{F(\xi_i)\}}, \quad \xi_i \sim q. \quad (10.15)$$

Terms: The estimator requires support: $q(\xi) > 0$ wherever $p(\xi)\mathbf{1}_{\{F\}} > 0$. Large weights create high variance; report weight diagnostics and effective sample size.

Adaptive stress testing can search for likely failure trajectories while preserving a probability model. Adversarial optimization can expose defects, but an adversarially found trajectory is evidence of possibility, not an estimate of real frequency. Report discovery and probability estimation separately.

Coverage should include software and timing. Run identical scenarios under cold start, concurrent load, dropped frames, delayed state, and fallback activation. Simulation that models mechanics but executes the policy in an unrealistically synchronous loop misses an important part of Physical AI.

10.13 Co-simulation and hardware in the loop

Complex systems often split across simulators: rigid-body dynamics, traffic or human agents, network timing, sensor rendering, and production control software. Co-simulation needs a master clock and an explicit step policy. If subsystem i advances with step Δt_i , choose synchronization points and define interpolation, event handling, and rollback. Otherwise causality can be reversed by message order.

Hardware-in-the-loop replaces one or more virtual components with real compute, controllers, sensors, or actuators. It reveals codec latency, driver behavior, quantization, queueing, clock drift, and saturation absent from an in-process notebook. Progress from software-in-the-loop to controller-in-the-loop, sensor-in-the-loop, and constrained plant trials. At each stage compare the same transaction logs and residuals.

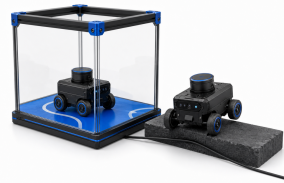
A digital twin can support shadow evaluation: the live system supplies current state, while a candidate policy proposes actions that are not executed. The twin estimates

their consequences and the active policy controls the plant. Shadow results are counterfactual model estimates, not real outcomes, but they are valuable for identifying proposal distributions and monitor interventions before canary release.

For every co-simulation or HIL setup, validate that the production action decoder, controller, monitor, and timing path are unchanged. A special “simulation mode” that bypasses bounds or uses a different coordinate convention invalidates transfer evidence.

PHYSICAL FAILURE MODE - Simulation success

Passing one million simulated episodes does not bound real risk if the simulator omits the failure mechanism. Every safety claim must name the validated simulator envelope and the real evidence supporting transfer.



ENGINEERING NOTE - Simulator card

Record engine and version, integration step, solver, contact model, robot/controller, sensor pipeline, calibrated parameters and confidence, randomization distribution, asset sources, known invalid regimes, determinism, real validation metrics, and intended decisions.



Figure 10.1: *Sim-to-real is a sequence of measured gaps.*

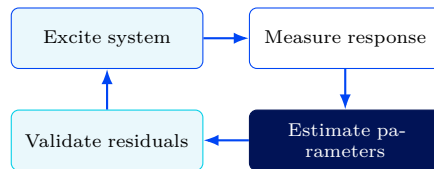


Figure 10.2: *System identification uses physical measurements to update the virtual model.*

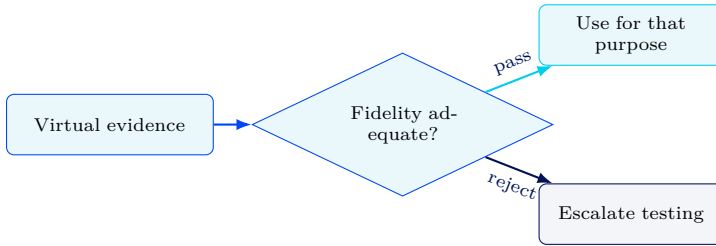


Figure 10.3: *A simulator is accepted only for a declared decision.*

Principal references

- [70] Foundational domain randomization.
- [71] End-to-end sim-to-real transfer.
- [72] Physics-simulator comparison.
- [73] GPU-parallel robot simulation.
- [74] Generalizable manipulation benchmark.
- [75] Human-avatar-robot co-habitat simulation.
- [76] Open urban-driving simulation.
- [15] Generative world foundation models.
- [77] Model selection and domain-shift foundations.

LESSON 11

Physical AI Hallucinations

LEARNING OBJECTIVES

After this lesson you can define Physical AI hallucination precisely; classify perceptual, grounding, affordance, world, temporal, action, and safety failures; separate hallucination from uncertainty and ordinary control error; build an evaluation dataset; and combine evidence, calibration, physical residuals, and runtime behavior to detect failures.

Provenance and scope

This lesson separates established literature from an original engineering synthesis. The exact Physical AI hallucination definition, integrated taxonomy, cross-model matrix, scenario schema and three-way evidence labels, severity-weighted miss measure, six-layer detector stack, mitigation sequence, operational scorecard, and incident record are proposed by Dr. Barak Or as part of the STATE16 Physical AI reliability framework. They are engineering proposals, not standards or consensus taxonomies. Citations identify the prior work supporting individual components.

11.1 Why the language definition is too narrow

Hallucination is a useful word, but an overloaded one. I do not use it as a synonym for every robot failure. In language models it often means an unsupported statement; in Physical AI the unsupported output may instead be a pose, affordance, future, action, or hidden premise. It need not contain words. A robot can silently “believe” an object is reachable by generating an out-of-reach action.

DEFINITION - Physical AI hallucination

A Physical AI hallucination is a model-generated perception, prediction, plan, affordance, or action whose represented premise or implied physical consequence lacks adequate support from the available evidence, task specification, embodiment, or admissible dynamics at the relevant time.

STATE16 provenance. The definition above is proposed by Dr. Barak Or for the STATE16 Physical AI reliability framework; it is not a standardized definition.

The definition rests on four anchors:

1. **evidence:** sensors, state estimate, memory, and validated knowledge;
2. **task:** what was requested, including constraints and permissions;
3. **embodiment:** geometry, control interface, capabilities, and limits;
4. **dynamics:** what can evolve under physical time and interaction.

An output can be plausible under a broad prior yet unsupported by the current scene. It can be correct semantically but false metrically. It can be physically possible but impossible for this robot. It can be safe in principle but stale by execution time.

11.2 Hallucination, uncertainty, and error

Not every error is hallucination. Sensor noise, controller tracking error, actuator fault, and unmodeled disturbance can create physical failure after a well-supported proposal. Conversely, a hallucinated action can accidentally succeed.

Use the following distinction:

- ▶ **estimation error:** inferred variable differs from reality;
- ▶ **prediction error:** future differs from outcome;
- ▶ **control error:** executed state differs from target;
- ▶ **hallucination:** the model asserts or operationalizes content without adequate support under its interface;
- ▶ **uncertainty failure:** confidence or abstention does not reflect what is unknown;
- ▶ **authorization failure:** an invalid proposal is allowed to execute, or a valid proposal is needlessly blocked.

These can co-occur. A VLA hallucinates a grasp pose because it assumes a hidden handle; confidence is overcalibrated; the runtime system fails to reject; the controller then tracks the pose accurately. The final controller has low tracking error while the system fails.

11.3 An evidence-based taxonomy

STATE16 provenance. The organization of failure modes by violated evidence, task, embodiment, and dynamics relations is the author’s synthesis of prior research and engineering practice, not a published consensus taxonomy.

The taxonomy below organizes failure by the violated support relation. It synthesizes VLM hallucination analysis, VLA action-hallucination work, robotics safety literature, and runtime authorization.

11.3.1 Perceptual existence and identity

Existence hallucination: the system represents an absent object, person, region, or event. **Omission:** a relevant present entity is missing. **Identity hallucination:** the entity exists but is assigned the wrong persistent identity or class.

Examples: reporting a tool behind an opaque box because similar tasks include it; swapping two identical bins after occlusion; missing transparent material. Test positive and negative scenes, counterfactual object removal, occlusion, and open-set categories.

For class c , existence score p_c should be calibrated. At threshold τ , report precision, recall, false-positive rate, and false-negative rate. Physical risk may weight false negatives and positives differently.

PHYSICAL FAILURE MODE - Phantom object

A detector or VLM can produce a precise region for an object that is absent. Require multi-view, temporal, geometric, or contact evidence before turning a semantic claim into action.



11.3.2 Attribute and state

The object is real but its property is unsupported: color, mass, temperature, damage, fullness, open/closed state, connection, or occupancy. Some attributes are visible; others require sensors or interaction. A VLM that infers “empty” from appearance can create unsafe lifting.

Represent latent attributes as unknown unless evidence exists. A state estimate should carry a distribution $p(s \mid o)$, not force a label. For decision a , expected loss is

$$\mathcal{R}(a \mid o) = \sum_s L(a, s) p(s \mid o). \quad (11.1)$$

Terms: s is possible object state; $L(a, s)$ consequence of action a . High-cost ambiguity should trigger sensing or abstention.

11.3.3 Spatial and geometric grounding

An entity is correct but its pixel, depth, pose, scale, orientation, topology, or relation is wrong. Examples include confusing image left with robot left, using a stale camera transform, placing a grasp behind the visible surface, or predicting a collision-free path through an unmodeled obstacle.

Measure 2-D region, 3-D position, rotation geodesic error, and collision margin. Decompose error into perception, calibration, and transform contributions. A relation such as “inside” or “supported by” should be evaluated from geometry and contact, not only text labels.

11.3.4 Semantic grounding and task-reference

The model resolves language to the wrong entity, goal, quantity, ordering, or negation. It may follow typical task priors instead of the current instruction. Examples: grasping the red cup when asked for the cup left of red; ignoring “do not open”; treating “place near” as “place inside.”

Use compositional and contrastive instructions with the same scene. Swap adjectives, relations, negation, and reference order. Hold visuals constant so linguistic grounding is isolated.

11.3.5 Affordance and capability

An affordance is not an object property alone. Define feasibility

$$F(a \mid o, e, g, t) \in \{0, 1\}, \quad (11.2)$$

where o is object/scene state, e embodiment, g task, and t time. An affordance hallucination asserts $F = 1$ without support.

Examples: a handle is visually graspable but too wide for the gripper; a step is traversable by a humanoid but not a wheeled base; a surface supports an empty gripper but not its carried load; a tool is reachable only after collision. Evaluate the same scene across embodiments and loads. Include “cannot determine” labels.

11.3.6 Kinematic hallucination

A proposed state sequence violates geometry, joint limits, reachability, velocity, acceleration, or collision-free motion. Smooth video is not enough. For predicted joint trajectory $\hat{q}_t$, simple residuals include

$$r_q = \max_{t,j} \left[\max(q_j^{min} - \hat{q}_{t,j}, 0, \hat{q}_{t,j} - q_j^{max}) \right], \quad (11.3)$$

$$r_v = \max_{t,j} \left[\frac{|\hat{q}_{t+1,j} - \hat{q}_{t,j}|}{\Delta t} - v_j^{max} \right]_+. \quad (11.4)$$

Terms: $[x]_+ = \max(x, 0)$; joint and rate bounds are platform-specific. Collision and cross-state reachability add stronger tests.

11.3.7 Dynamic and contact hallucination

A trajectory is kinematically smooth yet incompatible with force, torque, inertia, friction, contact, energy, or paired action. Examples: an object accelerates without force; a gripper passes through a rigid surface; a mobile robot turns faster than tire friction allows; a predicted action points right while the predicted object moves left.

With one-step dynamics mean $\mu_\theta(\mathbf{x}_t, \mathbf{a}_t)$ and scale σ_θ , standardized residual is

$$r_t^D = \left\| \frac{\mathbf{x}_{t+1} - \mu_\theta(\mathbf{x}_t, \mathbf{a}_t)}{\sigma_\theta(\mathbf{x}_t, \mathbf{a}_t) + \epsilon} \right\|_2. \quad (11.5)$$

This is model-relative. An analytical dynamics or reachability envelope provides stronger physical meaning when available.

11.3.8 Temporal hallucination

The content may be correct for the wrong time: stale object pose, ignored latency, impossible event order, incorrect duration, or continuing a plan after interruption. Define observation age at action $A_t = \tau_t^{exc} - \tau_t^{obs}$. A freshness violation is $[A_t - A_{max}]_+$.

Temporal grounding tests should vary playback speed, frame spacing, action delay, and scene change during inference. Store absolute timestamps; a sequence index alone is insufficient.

11.3.9 Causal and counterfactual hallucination

The model predicts a typical continuation while ignoring the supplied action or attributes an outcome to the wrong cause. A WAM may generate the object moving regardless of candidate action because similar videos contain motion.

Given paired actions A, A' from the same initial state, compare predicted effect difference with real effect difference:

$$r^{CF} = d\left((\hat{Y}^A - \hat{Y}^{A'}), (Y^A - Y^{A'})\right). \quad (11.6)$$

Randomized simulation or repeated controlled trials supply counterfactual evidence. Observational correlation alone cannot establish action causality.

11.3.10 Goal, plan, and completion hallucination

The system assumes a subgoal is achieved, invents a prerequisite, or declares completion from an ambiguous view. Language plans are especially vulnerable because each generated step can condition later steps. Bind every subgoal g_k to an observable predicate $c_k(\mathbf{x})$. Advance only when the predicate is verified with uncertainty below a threshold.

Task success should be measured from external state or independent evaluator. Model self-report is diagnostic, not ground truth.

11.3.11 Action-interface hallucination

The action is numerically valid but semantically incompatible with the configured robot: wrong dimension, joint order, sign, frame, scale, gripper convention, control mode, or rate. This category is often silent because tensors remain finite.

Use an action schema hash and dimensional names. Decode with a versioned transform D_e . Validate

$$\text{hash}(\text{model action schema}) = \text{hash}(\text{robot action schema}) \quad (11.7)$$

before activation. Then check physical bounds after denormalization.

11.3.12 Uncertainty and confidence hallucination

The system communicates unwarranted certainty, or its uncertainty omits model shift. Selective prediction should reduce error as coverage falls. Expected calibration error is

$$\text{ECE} = \sum_{b=1}^B \frac{|S_b|}{N} |\text{acc}(S_b) - \text{conf}(S_b)|. \quad (11.8)$$

Terms: S_b contains predictions in confidence bin b ; accuracy and mean confidence are computed within the bin. ECE depends on bins and does not capture rare catastrophic error.

Use class-conditional calibration, risk-coverage curves, ensemble disagreement, and out-of-distribution tests. A generative model’s token probability is not directly the probability an action is physically safe.

11.3.13 Safety and norm hallucination

The model proposes an action violating a safety rule, human preference, permission, right-of-way, or operational policy, perhaps because the physical action is otherwise feasible. “Can” and “may” differ. Encode operational constraints separately and enforce them with authorization, not prompt text alone.

Security failures such as prompt injection, sensor spoofing, and training-data poisoning can induce any hallucination category. A physically grounded system requires trust boundaries around observations, instructions, updates, and model artifacts.

11.4 A matrix across model families

Failure	VLM symptom	VLA symptom	World/WAM symptom
Existence	names absent object	reaches toward empty space	renders object into future
Spatial Affordance	wrong relation/point says “graspable”	wrong grasp/path generates infeasible grasp	penetrates or shifts geometry predicts successful effect
Temporal Dynamic	stale description unsupported force claim	stale action chunk acceleration/action mismatch	wrong duration/event order impossible motion/contact
Causal Completion Interface	answers from prior declares done malformed structure	ignores visual change stops early wrong action convention	future insensitive to action future depicts completed task paired action/state mismatch

Table 11.1: *The same unsupported premise appears differently at each model interface. This cross-model matrix is a STATE16 synthesis.*



11.5 Building an evaluation set

STATE16 provenance. The benchmark schema, paired-case design, three-way evidence labels, admissibility labels, and severity weighting in this section are proposed STATE16 evaluation methods.

A Physical AI hallucination benchmark should be scenario-based, not only question-based. Each item includes initial state, observations with timestamps, instruction, embodiment/configuration, candidate output, evidence labels, physical envelope, and consequences.

Construct paired cases:

- ▶ entity present versus removed;
- ▶ reachable versus just beyond reach;
- ▶ same image with different robot embodiment;
- ▶ same geometry with different mass/friction/load;
- ▶ synchronized versus delayed observation;
- ▶ consistent versus mismatched action-state future;
- ▶ benign instruction versus injected or contradictory instruction;
- ▶ successful subgoal versus visually similar incomplete state.

Label which evidence is available. A model should not be penalized for abstaining when force cannot be inferred visually. Use three labels: supported, contradicted, insufficient evidence. For action proposals, add admissible under envelope, inadmissible, and unknown.

Stratify by severity. Let $\mathcal{H}$ be the set of true hallucination cases, let $c_i > 0$ be the consequence cost of case i , and let $\hat{h}_i = 1$ when the detector flags that case. The severity-weighted miss rate is

$$\text{weighted miss rate} = \frac{\sum_{i \in \mathcal{H}} c_i (1 - \hat{h}_i)}{\sum_{i \in \mathcal{H}} c_i}. \quad (11.9)$$

Terms: The numerator is the total consequence cost of missed hallucinations; the denominator is the total consequence cost of all true hallucinations. A value of zero means none of the known hallucination cost was missed.

Report ordinary rates too; severity estimates can be subjective.

EXERCISE 11.1 - Turning a headline success rate into an engineering test plan

For planning only, scale the two published rates to 200 trials per condition. A 58.5% rate corresponds to 117 successes and 83 failures; an 82% rate corresponds to 164 successes and 36 failures. The difference is 47 outcomes per 200 trials.

That arithmetic does not identify a causal treatment because the conditions differ. It tells the test engineer how much failure mass needs explanation. If 25 of the 83 natural-home failures are semantic retrieval, 18 navigation, 22 grasping, and 18 placement or recovery, improving only the VLM can address at most the first group. Those category counts are a proposed logging design, not values reported by OK-Robot.

11.6 Detection is multi-layered

STATE16 provenance. The six-layer detector architecture and the max-normalized residual monitor below are a STATE16 engineering synthesis. Their thresholds require independent calibration for the target system.

No single detector covers all categories. Combine:

1. **evidence grounding:** region/pose/state linked to sensor evidence;
2. **cross-modal consistency:** language, vision, state, and action agree;
3. **uncertainty/OOD:** calibration, ensembles, density or feature-distance signals;
4. **temporal consistency:** timestamps, tracking, direct versus rolled future;
5. **physical residuals:** bounds, reachability, dynamics, collision, energy, contact;
6. **runtime outcome:** controller tracking, progress, fallback, and intervention.

If normalized residuals are r_k/b_k , a conjunctive monitor can use

$$S = \max_k \frac{r_k}{b_k}, \quad \text{reject if } S > \eta. \quad (11.10)$$

Terms: $r_k \geq 0$ is residual; $b_k > 0$ certified or calibrated bound; η operating threshold. The maximum preserves the active failure component.

Use certified bounds for formal rejection claims. Data-derived bounds define empirical guards only. Missing variables should disable or mark a check unavailable, not silently impute a pass.

Cross-checking two models helps only if their errors are sufficiently independent. A VLM critic sharing the same backbone and data can repeat the policy's prior. Prefer

heterogeneous evidence: geometry engine, state estimator, controller limits, independent dynamics, and sensors.

EXERCISE 11.2 - Why detector accuracy hides unsafe exposure

In 10,000 action proposals, suppose 200 are unsafe. A detector catches 180 and falsely rejects 490 of the 9,800 safe proposals. Recall is $180/200 = 90\%$, false-intervention rate is $490/9,800 = 5\%$, and precision is $180/(180+490) = 26.9\%$. Overall accuracy is $9,490/10,000 = 94.9\%$, yet 20 unsafe proposals remain.

Report unsafe survivors, severity, and exposure time, not accuracy alone. Process metrics such as cumulative safety cost and risk exposure time make nominally successful but unsafe trajectories visible [78]. The threshold must be tied to a fallback whose cost is also measured.

11.7 Action hallucination and horizon

Recent work on action hallucination in generative VLAs emphasizes that action errors have structure across topology, precision, and horizon. A sample can select the wrong motion mode; it can select the correct mode with poor precision; or a locally valid chunk can drift over a longer horizon.

Decompose action error relative to a set of valid actions $\mathcal{A}^*(o, g)$:

$$d(\hat{A}, \mathcal{A}^*) = \inf_{A \in \mathcal{A}^*} d(\hat{A}, A). \quad (11.11)$$

One expert demonstration is not the full valid set. Simulation, multiple experts, task constraints, and outcome-based evaluation approximate it.

Horizon risk accumulates. Let F_t denote failure at step t . Define the survival event, conditional hazard, and event of at least one failure by horizon H as

$$\begin{aligned} S_{t-1} &= \bigcap_{j=1}^{t-1} F_j^c, & S_0 &= \Omega, \\ h_t &= \mathbb{P}(F_t \mid S_{t-1}), & F_{\leq H} &= \bigcup_{t=1}^H F_t, \\ \mathbb{P}(F_{\leq H}) &= 1 - \prod_{t=1}^H (1 - h_t). \end{aligned} \quad (11.12)$$

Even small per-step hazards matter over long execution. Receding-horizon observation, frequent authorization, and recovery reduce exposure.

EXERCISE 11.3 - Base rates, residual exposure, and production cost

Suppose a shadow deployment logs 100,000 proposals and offline adjudication identifies 0.4%, or 400, as invalid. A learned detector with 99% recall leaves four invalid proposals. Deterministic joint, workspace, and rate bounds catch three, leaving one proposal that would reach the controller without another layer.

If the detector falsely blocks 2% of the 99,600 valid proposals, it creates 1,992 interventions. The system therefore needs both a severity-weighted residual-risk argument and a productive fallback. “99% recall” alone describes neither. These values are a design calculation; acceptance thresholds must come from the actual cell risk assessment.

11.8 Mitigation by design

STATE16 provenance. The before, during, and after mitigation sequence in this section is a STATE16 systems framework assembled from the cited component methods.

Mitigation should occur before, during, and after inference.

Before: curated negative and recovery data, counterfactuals, calibrated sensors, explicit unknown labels, action schemas, embodiment conditioning, safety constraints, and red-team cases.

During: constrained decoding, retrieval from verified task/state data, action bounds, multiple candidates, WAM reranking, uncertainty-aware planning, independent monitors, and freshness checks.

After: low-level control, collision checking, barrier filters, runtime assurance, fallback, human intervention, telemetry, incident review, and controlled retraining.

Retrieval can ground facts but not live geometry unless the retrieved data are current and registered. Chain-of-thought or rationale can improve task decomposition but can also produce a persuasive fiction. Treat rationale as trace, not proof.

For active perception, choose sensing action a_s to reduce expected uncertainty:

$$a_s^* = \arg \max_{a_s} [H(b_t) - \mathbb{E}_{o_{t+1}|a_s} H(b_{t+1})] - \lambda C(a_s), \quad (11.13)$$

Terms: $H(b) = -\int b(x) \log b(x) dx$ is belief entropy; $C(a_s)$ time or risk of sensing action. Looking again can be better than acting under unsupported certainty.

PHYSICAL FAILURE MODE - Persuasive output

Natural-language explanations, photorealistic videos, and smooth actions all increase perceived credibility. None is independent evidence of grounding. Design the interface so confidence comes from measurable support, not presentation quality.



11.9 Operational scorecard

STATE16 provenance. The combined operational scorecard and incident-record fields in this section are proposed STATE16 reliability practices.

For each category, report prevalence, detection AUC/AP, operating-point false reject and false accept, cost-weighted failures, time to detection, fallback success, retained task progress, and residual shift over deployment. Keep a confusion matrix by scenario and embodiment.

A monitor that blocks every action has perfect invalid-prevention but zero utility. Pair prevention with false intervention and progress:

$$P_{\text{prevent}} = \frac{\text{\#invalid proposals blocked}}{\text{\#invalid proposals}}, \quad (11.14)$$

$$P_{\text{false}} = \frac{\text{\#valid proposals blocked}}{\text{\#valid proposals}}, \quad (11.15)$$

$$R_{\text{progress}} = \frac{\text{progress with monitor}}{\text{progress of accepted nominal reference}}. \quad (11.16)$$

The goal is not to eliminate uncertainty. It is to prevent unsupported confidence from silently becoming physical action.

ENGINEERING NOTE - Incident record

Store raw observations, calibration/configuration hashes, instruction provenance, model and dataset revisions, normalized and physical actions, timestamps, uncertainty, monitor residuals, active rejection component, controller

output, state outcome, fallback, and human intervention. Without this chain, hallucinations are anecdotes rather than engineering data.

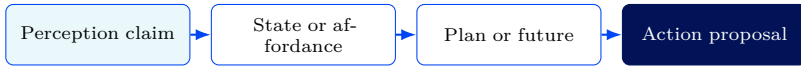


Figure 11.1: A physical hallucination can enter through any model boundary.

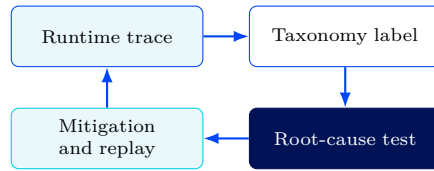


Figure 11.2: Hallucination analysis links incident traces back to data and interfaces.

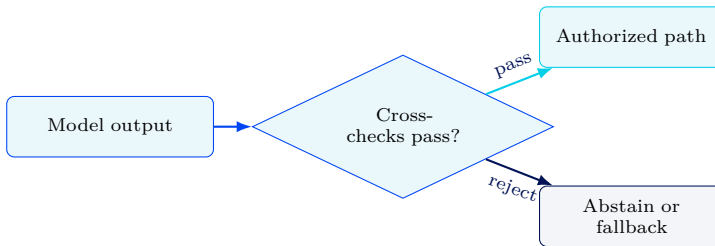


Figure 11.3: Multi-layer detection combines semantics, geometry, dynamics, and safety.

Principal references

- [79] Runtime action authorization and silent failure review.
- [80] Mechanistic analysis of action hallucination.
- [81] VLA threat, defense, evaluation, and deployment survey.
- [78] Diagnostic VLA safety taxonomy and process metrics.
- [82] Aleatoric and epistemic uncertainty.
- [83] Deep-ensemble uncertainty estimation.
- [84] OOD detection baseline.
- [85] Concrete AI safety problem framing.
- [86] Safe reinforcement-learning survey.



LESSON 12

From Prediction to Authorization

LEARNING OBJECTIVES

After this lesson you can separate prediction, planning, control, and authorization; derive uncertainty and out-of-distribution gates; formulate reachability, barrier, and runtime-assurance filters; implement the physical-admissibility monitor introduced in “Can Predicted Dynamics Exist in the Physical World?”; and evaluate prevention, false intervention, fallback, and retained progress.

12.1 The missing decision

A policy proposes. A controller tracks. Between them lies a decision often left implicit: should this proposal be allowed to influence the plant now? Runtime authorization makes that decision explicit.

Let model output be y_t , decoder D , monitored proposal $\hat{\xi}_{t:t+H} = D(y_t)$, envelope Θ_t , and monitor M . The interface is

$$M(\hat{\xi}_{t:t+H}, \Theta_t) \rightarrow (d_t, c_t, I_t, S_t), \quad (12.1)$$

Terms: $d_t \in \{\text{PASS}, \text{REJECT}\}$; c_t active condition; I_t offending time/dimension indices; S_t scalar severity score.

Authorization is narrower than global safety certification. It decides whether a particular decoded proposal is compatible with a specified envelope at a particular time. Rejection does not mean no safe action exists; it means this proposal should be blocked, replanned, modified, or routed to fallback [87].

12.2 A layered authorization architecture

Use layers with different evidence:

1. **interface**: schema, checksum, units, finite values, timestamps, mode;
2. **actuator**: position, velocity, acceleration, torque, power, rate, and saturation;
3. **geometry**: workspace, collision, reachability, geofence;
4. **dynamics**: action-state consistency, contact, energy, model residual;
5. **uncertainty**: calibration, ensemble disagreement, OOD, stale state;
6. **task and policy**: permissions, human zones, sequencing, operational rules;
7. **runtime assurance**: fallback controller and mode-switch logic.

Early layers should be deterministic and cheap. Later layers may use probabilistic models. A slow sophisticated critic cannot replace hard action bounds.

If checks run in parallel, decision latency is approximately their maximum plus aggregation; if sequential, it is their sum. Order sequential checks by low cost and high rejection yield, but preserve all required checks within the deadline.

12.3 Uncertainty and selective authorization

A regression ensemble yields action mean $\bar{\mathbf{a}}$ and covariance

$$\bar{\mathbf{a}} = \frac{1}{M} \sum_{m=1}^M \mathbf{a}^{(m)}, \quad \Sigma_a = \frac{1}{M-1} \sum_{m=1}^M (\mathbf{a}^{(m)} - \bar{\mathbf{a}})(\mathbf{a}^{(m)} - \bar{\mathbf{a}})^\top. \quad (12.2)$$

Gate on task-sensitive uncertainty, for example $S_U = \lambda_{\max}(W\Sigma_a W^\top)$, where W is a declared linear weighting or sensitivity map. A scalar trace can hide one dangerous direction.

► LANDMARK RESULT 12.1 - Split-conformal finite-sample coverage

For a probabilistic classifier or success model, split conformal prediction can provide finite-sample marginal coverage under exchangeability. Given n calibration nonconformity scores with order statistics $s_{(1)} \leq \dots \leq s_{(n)}$, define

$$q_\alpha = s_{(k)}, \quad k = \lceil (n+1)(1-\alpha) \rceil, \quad s_{(n+1)} := +\infty, \quad (12.3)$$

and $\Gamma_\alpha(x) = \{y : s(x, y) \leq q_\alpha\}$. Then $\mathbb{P}(Y_{\text{new}} \in \Gamma_\alpha(X_{\text{new}})) \geq 1 - \alpha$ under the stated assumptions [88]. Conformal coverage is not an authorization rule: a system must still specify, for example, that every outcome retained in $\Gamma_\alpha(x)$ is acceptable. Distribution shift, temporal dependence, and adaptive policies can invalidate simple exchangeability, so monitor empirical coverage and recalibrate by operating regime.

Out-of-distribution signals include feature distance, ensemble disagreement, density, reconstruction, and classifier energy. For embeddings $\mathbf{z}$, Mahalanobis distance is

$$D_M(\mathbf{z}) = \sqrt{(\mathbf{z} - \boldsymbol{\mu})^\top (\boldsymbol{\Sigma} + \lambda I)^{-1} (\mathbf{z} - \boldsymbol{\mu})}. \quad (12.4)$$

Terms: $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ are training/reference feature statistics; λ stabilizes covariance. High distance suggests shift, not necessarily failure; low distance does not prove validity.

Combine uncertainty with physical constraints. Uncertainty-only gates often miss confidently structured failures and block unfamiliar but valid behavior.

PHYSICAL FAILURE MODE - False-negative gate

A gate can pass an unsafe proposal when its detector, threshold, or monitored variables omit the active hazard. Validate miss rate under targeted faults and retain independent fallback logic.



12.4 Reachability

For continuous dynamics $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, pointwise admissible inputs $\mathcal{U}$, and admissible input signals $\mathcal{U}_T$, the forward reachable set over duration T is

$$\mathcal{R}_T(\mathbf{x}_0) = \{\mathbf{x}(T) : \dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}), \mathbf{x}(0) = \mathbf{x}_0, \mathbf{u}(\cdot) \in \mathcal{U}_T\}. \quad (12.5)$$

If a predicted state lies outside a conservative outer approximation $\overline{\mathcal{R}}_T \supseteq \mathcal{R}_T$, it is unreachable under assumptions. Inside the outer approximation is not a guarantee because the set may be loose.

For linear discrete dynamics $\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k$, horizon H gives

$$\mathbf{x}_H = A^H \mathbf{x}_0 + \sum_{i=0}^{H-1} A^{H-1-i} B \mathbf{u}_i. \quad (12.6)$$

With convex input sets, reachable sets can be propagated by Minkowski sums. Non-linear, contact, and high-dimensional robots require approximations, local linearization, zonotopes, sampling, or learned envelopes [89, 90].

Reachability can check collision avoidance by ensuring reachable occupancy avoids unsafe set $\mathcal{X}_{bad}$. It can also check a predicted rollout itself, asking whether every later state is reachable from every earlier state over elapsed time.

EXERCISE 12.1 - Stopping distance versus predicted free space

At $v = 15$ m/s, end-to-end response latency $t_\ell = 0.12$ s, and achievable deceleration $a = 6$ m/s², the idealized stopping distance is

$$d_{\text{stop}} = vt_\ell + \frac{v^2}{2a} = 15(0.12) + \frac{225}{12} = 20.55 \text{ m.}$$

If a world model predicts 25 m of free space while calibrated sensing supports only 18 m, the visually coherent prediction would authorize a maneuver that fails this simple envelope. Even the 20.55 m calculation omits grade, tire friction, actuator uncertainty, and comfort constraints; a production margin must cover them.



12.5 Control barrier functions

► LANDMARK RESULT 12.2 - Control-barrier forward invariance

Let safe set be $C = \{\mathbf{x} : h(\mathbf{x}) \geq 0\}$. For control-affine dynamics

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \quad (12.7)$$

a zeroing control barrier condition is

$$L_f h(\mathbf{x}) + L_g h(\mathbf{x})\mathbf{u} + \alpha(h(\mathbf{x})) \geq 0. \quad (12.8)$$

Terms: $L_f h = \nabla h^\top f$ and $L_g h = \nabla h^\top g$ are Lie derivatives; α is an extended class- $\mathcal{K}$ function, often $\alpha(s) = \kappa s$.

If h is continuously differentiable, the dynamics and selected feedback are sufficiently regular, closed-loop solutions exist, and a feasible control satisfying the inequality is selected for every state in C , then the resulting closed loop renders C forward invariant: $\mathbf{x}(0) \in C$ implies $\mathbf{x}(t) \in C$ for all later times in the solution interval [91]. This is the bridge from a pointwise inequality to a closed-loop safety statement.

Engineering meaning. The theorem does not absorb model error, sampled implementation, state-estimation error, delay, saturation, infeasible constraints, or an incorrectly specified safe set. Those effects require robust margins, feasibility handling, and independent runtime evidence.

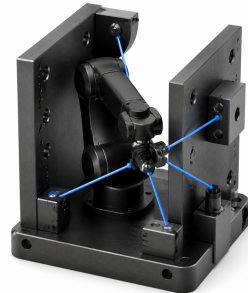
Filter learned proposal $\mathbf{u}_{nom}$ with a quadratic program:

$$\mathbf{u}^\star = \arg \min_{\mathbf{u} \in \mathcal{U}} \frac{1}{2} \|\mathbf{u} - \mathbf{u}_{nom}\|_2^2 \quad \text{s.t. the barrier condition.} \quad (12.9)$$

The filter minimally modifies action. Feasibility can fail when the state is already too close to danger, the model is wrong, or constraints conflict. Add slack only with explicit consequence; large slack means safety is no longer guaranteed. Log both nominal and filtered action.

PHYSICAL FAILURE MODE - Barrier infeasibility

A safety filter cannot manufacture feasible control when the state is already trapped, actuators saturate, or constraints conflict. Detect infeasibility early and route to a defined safe fallback.



For a discrete-time system, a common sufficient form is

$$h(\mathbf{x}_{t+1}) \geq (1 - \gamma)h(\mathbf{x}_t), \quad 0 < \gamma \leq 1,$$

evaluated with a trustworthy transition model and robust margins. More general discrete-time barrier definitions are possible, but the chosen inequality and control-feasibility assumptions must be stated explicitly.

12.6 Simplex and runtime assurance

The Simplex pattern includes an advanced controller π_A , a verified or otherwise assured baseline controller π_B , a decision module, and a recoverable region [92, 93]. The advanced learned controller operates while the switching condition holds; otherwise control authority moves to π_B .

Let $\mathcal{R}_B$ contain states from which the baseline can keep all safety constraints satisfied while remaining in, or driving the system to, a controlled-invariant safe set. The decision module must switch early enough that actuation and computation delay do not carry the state outside $\mathcal{R}_B$. Hysteresis prevents chattering:

$$\begin{aligned} &\text{switch to baseline if } S > \eta_{high}, \\ &\text{return to advanced if } S < \eta_{low}, \quad \eta_{low} < \eta_{high}. \end{aligned}$$

Fallback must be tested. A stop command can be unsafe for a drone, balancing robot, surgical device, or vehicle in traffic. Baseline behavior may be hover, controlled braking, hold, retreat, yield, or request human control. Its assumptions and authority must be independent enough to remain useful when the learned model fails.

12.7 Can Predicted Dynamics Exist in the Physical World?

The paper “Can Predicted Dynamics Exist in the Physical World?” formulates an interface-level question that recurs throughout this book: before a predicted trajectory or action chunk is passed downstream, is it compatible with a stated physical envelope? Its answer is not another predictor, but a set of observable tests placed between prediction and authorization [94].

Let a predictor produce sampled rollout

$$\hat{\mathbf{x}}_{t:t+K} = (\hat{\mathbf{x}}_t, \hat{\mathbf{x}}_{t+1}, \dots, \hat{\mathbf{x}}_{t+K}), \quad \tau_i = \tau_t + i\Delta t. \quad (12.10)$$

A state-only rollout is physically admissible if some admissible control signal, respecting platform magnitude and rate limits, could have generated all sampled states on this grid. If the proposal also declares an action sequence, the stronger action-conditioned requirement asks whether those particular actions are compatible with the proposed transitions. If contact, actuator state, or hidden variables matter, the monitored state must be augmented for a formal Markov interpretation; otherwise the checks are empirical monitored-envelope tests.

The paper derives four condition families from three observable interfaces: the multi-horizon prediction map, sampled state curve, and optional action-conditioned transitions.

Condition I: flow consistency

For action-conditioned prediction map $\hat{\phi}$, direct and composed forecasts should agree:

$$\Delta_{h,k}^{F,u} = \|\hat{\phi}_{h+k}(\mathbf{x}_t, \mathbf{u}_{0:h+k-1}) - \hat{\phi}_k(\hat{\phi}_h(\mathbf{x}_t, \mathbf{u}_{0:h-1}), \mathbf{u}_{h:h+k-1})\|. \quad (12.11)$$

Terms: h is prefix horizon; k suffix horizon; $\mathbf{u}_{a:b}$ action subsequence. Violation occurs when $\Delta_{h,k}^{F,u} > \epsilon_{h,k}^F$. For non-Markov predictors, advance the information state before the suffix.

This residual tests the forecasting interface, not state-trajectory feasibility by itself. A nonzero value proves that the learned predictor is not an exact compositional flow at that split; it does not, without additional assumptions, prove that either decoded trajectory is physically impossible.

Condition II: recursive reachability

Define reachable set $\mathcal{R}_\tau(\mathbf{x})$. Every later predicted state should be reachable from every earlier predicted state over the intermediate duration:

$$\hat{\mathbf{x}}_{t+h+k} \in \mathcal{R}_{k\Delta t}(\hat{\mathbf{x}}_{t+h}), \quad \Delta_{h,k}^R = d(\hat{\mathbf{x}}_{t+h+k}, \mathcal{R}_{k\Delta t}(\hat{\mathbf{x}}_{t+h})). \quad (12.12)$$

Terms: $d(\mathbf{x}, S) = \inf_{\mathbf{y} \in S} \|\mathbf{x} - \mathbf{y}\|$. An outer reachable approximation supports rejection when the point lies outside it.

Condition III: bounded differential growth

Finite differences are

$$\Delta^0 \hat{\mathbf{x}}_{t+i} = \hat{\mathbf{x}}_{t+i}, \quad \Delta^p \hat{\mathbf{x}}_{t+i} = \Delta^{p-1} \hat{\mathbf{x}}_{t+i+1} - \Delta^{p-1} \hat{\mathbf{x}}_{t+i}. \quad (12.13)$$

Define discrete derivative $D_{\Delta t}^p = \Delta^p / \Delta t^p$ and energy

$$E_p^\Delta(H) = \sum_{i=0}^{H-p} \|D_{\Delta t}^p \hat{\mathbf{x}}_{t+i}\|_2^2 \Delta t. \quad (12.14)$$

Violation occurs when $E_p^\Delta(H) > B_p(H)$, with bound induced by dynamics and actuator-rate constraints.

Condition IV: learned dynamics consistency

With dynamics model mean $\boldsymbol{\mu}_\theta(\mathbf{x}, \mathbf{u})$ and componentwise scale $\boldsymbol{\sigma}_\theta$, transition residual is

$$\Delta_i^D = \left\| \frac{\hat{\mathbf{x}}_{t+i+1} - \boldsymbol{\mu}_\theta(\hat{\mathbf{x}}_{t+i}, \mathbf{u}_{t+i})}{\boldsymbol{\sigma}_\theta(\hat{\mathbf{x}}_{t+i}, \mathbf{u}_{t+i}) + \epsilon \mathbf{1}} \right\|_2. \quad (12.15)$$

The division by $\boldsymbol{\sigma}_\theta + \epsilon \mathbf{1}$ is componentwise, and $\epsilon > 0$ prevents division by zero. This residual detects action-state incompatibility that a smooth kinematic curve can hide. It is model-relative unless the dynamics model exactly describes the admissible one-step successor relation.

One interpretable score

Normalize every residual by its bound and take the worst component:

$$S(\hat{\mathbf{x}}_{t:t+K}) = \max \left\{ \max_{h,k} \frac{\Delta_{h,k}^R}{\epsilon_{h,k}^R}, \max_{p,H} \frac{E_p^\Delta(H)}{B_p(H)}, \max_i \frac{\Delta_i^D}{\epsilon^D}, \max_{h,k} \frac{\Delta_{h,k}^F}{\epsilon_{h,k}^F} \right\}. \quad (12.16)$$

Unavailable families are omitted, not imputed. Reject if $S > \eta$. Under certified envelopes, $\eta = 1$ has a direct bound interpretation; empirical envelopes calibrate η on held-out nominal data. The maximum is a logical OR over conjunctive conditions and retains the active cause.

Every normalizing denominator must be strictly positive. Because omission changes the meaning and distribution of S , the implementation must log the active family mask and calibrate thresholds for each permitted mask rather than compare scores from different monitor configurations as if they were identical.

What the experiment showed

The paper evaluates compact predictive interfaces on LeRobot PushT with horizon $K = 32$. Structured violations include smooth impulse, actuator lag, time warp, contact-like mode change, action-state mismatch, and action saturation. The history-conditioned model obtained rollout RMSE 0.00221 ± 0.00099 , compared with 0.01000 ± 0.00341 for the state-only ensemble. This result is consistent with insufficiency of the two-dimensional monitored state for prediction, but it is not proof that history is the only missing information. Transition-RMSE and standardized dynamics residuals reached AUC 0.982 ± 0.002 and 0.972 ± 0.003 , respectively; kinematic-only scoring reached AUC 0.592 ± 0.015 on the structured dynamic-violation study.

At a threshold calibrated to the 95th percentile of held-out nominal windows, the detection study reported 5.1% false rejection. In the separate replay intervention study, the full physical gate prevented 87.7% of invalid proposals with 8.5% false intervention, while mean retained progress remained near 0.998. Controlled direct-forecast inconsistency was detected at 100% with AUC/AP 1.000 in that constructed stress test. These are PushT results under controlled perturbations and replay, not hardware-wide guarantees.

The conservative conclusion is essential: passing does not imply task success, safety, or prediction correctness. Rejection identifies incompatibility with the specified monitored envelope. The contribution is a practical pass/reject-and-diagnose interface that can surround opaque predictive models.

12.8 Calibrating an empirical monitor

Split data by trajectory and condition. Fit bounds on nominal training/calibration data, then lock them. For residual test j , choose a bound b_j so that $\mathbb{P}(r_j > b_j) \leq \alpha_j$ under the calibration regime. If N_r tests are evaluated together and $\sum_{j=1}^{N_r} \alpha_j \leq \alpha$, the

union bound gives

$$\mathbb{P}\left(\bigcup_{j=1}^{N_r}\{r_j > b_j\}\right) \leq \sum_{j=1}^{N_r} \mathbb{P}(r_j > b_j) \leq \sum_{j=1}^{N_r} \alpha_j \leq \alpha. \quad (12.17)$$

Terms: N_r is the number of simultaneously evaluated residual tests; r_j is residual j ; b_j is its bound; and α_j is its allocated exceedance probability. The union bound does not require independence.

Temporal correlation reduces effective sample size; calibrate on episodes and environments, not frames. Use leave-site/object/robot-out validation where appropriate. Report threshold sensitivity and residual distributions.

Threshold selection is a cost and requirement decision. Let $D \in \{\text{valid}, \text{invalid}\}$, let $\pi_d = \mathbb{P}(D = d)$ denote the deployment class prior, let C_{miss} be the cost of passing an invalid proposal, and let C_{int} be the cost of rejecting a valid proposal. A two-error empirical risk is

$$\begin{aligned} R(\eta) = & C_{\text{miss}}\pi_{\text{invalid}}\mathbb{P}(S \leq \eta \mid D = \text{invalid}) \\ & + C_{\text{int}}\pi_{\text{valid}}\mathbb{P}(S > \eta \mid D = \text{valid}). \end{aligned} \quad (12.18)$$

Terms: $S \leq \eta$ means pass and $S > \eta$ means reject. The first probability is the false-pass rate; the second is the false-intervention rate.

This notation avoids ambiguous positive-class terminology. Hard safety requirements override minimum empirical risk.

EXERCISE 12.2 - Interpreting an authorization result without overstating it

Scale the PushT replay rates reported in [94] to an illustrative set containing 200 invalid and 800 valid proposals. Preventing 87.7% of invalid proposals corresponds to about 175 blocked and 25 surviving, a 12.3% false-pass rate before rounding. An 8.5% false-intervention rate corresponds to 68 valid proposals blocked and 732 accepted.

These counts are an interpretation aid, not the experiment's sample size. They expose both sides of the operating point: the monitor materially reduces invalid exposure but does not eliminate it, and it trades away some valid progress. Deployment requires domain-specific thresholds, independent constraints, and fallback behavior.

12.9 Fallback and graceful degradation

On rejection, options include:

- ▶ request a new observation or shorter chunk;
- ▶ resample the generative policy;
- ▶ ask WAM/planner for a different candidate;
- ▶ project the action into a feasible set;
- ▶ execute a trusted hold, brake, retreat, or hover controller;
- ▶ transfer control to a human;
- ▶ terminate the task in a safe state.

The active component should guide response. Stale observation requests sensing; reachability violation requests replanning; small barrier violation may be projected; schema mismatch requires stopping and configuration review. A single generic retry can loop.

Define retry budget and escalation. If n consecutive proposals are rejected or residual remains above a critical threshold, enter degraded mode. Log why.

12.10 Authorization metrics

Offline detection metrics AUC and AP are useful across thresholds; here invalid proposals must be declared as the positive class. Runtime metrics are more important:

$$\text{invalid prevented} = \frac{N_{\text{invalid,blocked}}}{N_{\text{invalid}}}, \quad (12.19)$$

$$\text{false pass} = \frac{N_{\text{invalid,passed}}}{N_{\text{invalid}}}, \quad (12.20)$$

$$\text{false intervention} = \frac{N_{\text{valid,blocked}}}{N_{\text{valid}}}, \quad (12.21)$$

$$\text{fallback success} = \frac{N_{\text{fallback,safe progress}}}{N_{\text{fallback}}}, \quad (12.22)$$

$$\text{retained progress} = \frac{G_{\text{guarded}}}{G_{\text{nominal valid}}}. \quad (12.23)$$

Terms: N_{invalid} and N_{valid} are labeled proposal counts; passed and blocked partition the evaluated proposals; N_{fallback} counts fallback executions; and G is a predeclared progress measure. Thus invalid-prevented rate plus false-pass rate equals one when every invalid proposal receives exactly one decision.

Add time-to-detection, monitor latency, cause attribution, repeated-rejection rate, and residual shift. Evaluate interaction: filtering can move the system into states the original policy did not expect, and fallback can create new observations.

12.11 Assurance case

An authorization layer should have an explicit assurance argument:

- ▶ claim: which proposal properties are checked;
- ▶ assumptions: state, timing, model, and environment envelope;
- ▶ evidence: analysis, calibration, tests, hardware trials, fault injection;
- ▶ limits: unmonitored variables and failure modes;
- ▶ response: fallback and operator procedures;
- ▶ monitoring: how assumption drift is detected.

Never label an empirical neural residual as certified. Conversely, do not discard it because it lacks proof: empirical dynamics checks can detect structured failures missed by simple kinematic bounds. State the strength of each layer.

ENGINEERING NOTE - The authorization contract

Input a decoded, timestamped proposal in physical coordinates. Return pass/reject, normalized severity, active condition, offending indices, envelope revision, and monitor latency. Preserve enough evidence for replay. Treat pass as permission under the named checks, never as a universal certificate.



Figure 12.1: Runtime authorization turns model evidence into an explicit decision.

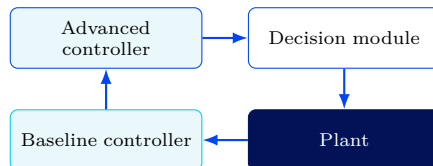


Figure 12.2: A Simplex architecture can preserve progress while retaining an assured fallback.

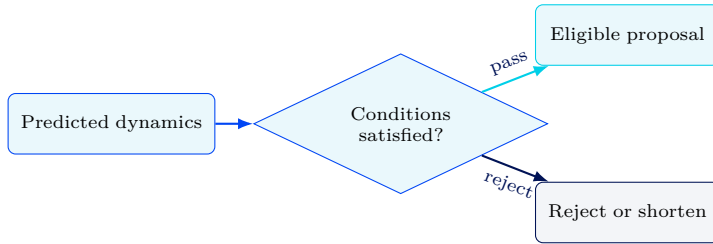


Figure 12.3: *Physical-admissibility tests are bounded proposal filters, not proofs of task success.*

Principal references

- [94] Physical-admissibility conditions and PushT replay evidence.
- [79] Runtime authorization literature synthesis.
- [91] Control barrier function theory.
- [92] Original Simplex runtime-assurance architecture.
- [93] Black-box Simplex architecture.
- [89] Reachability analysis tooling.
- [90] General safety framework for learning-based control.
- [88] Distribution-free uncertainty quantification.
- [87] Unified safety-filter framework and intervention semantics.
- [86] Safe learning foundations.

LESSON 13

Physical AI Systems Engineering

LEARNING OBJECTIVES

After this lesson you can design an end-to-end Physical AI architecture; budget latency and compute; create release gates and observability; manage dataset, model, configuration, and monitor revisions; plan guarded deployment across robot domains; and organize a team capable of training, retraining, evaluating, and operating VLM, VLA, world-model, JEPA, and WAM components.

13.1 The deployed system is the unit of intelligence

A robot does not deploy a checkpoint. It deploys a chain of sensors, clocks, transforms, state estimates, learned models, planners, controllers, monitors, networks, configurations, human procedures, and fallback behavior. Any model in that chain can be replaced; the system contract has to endure. Success belongs to the whole chain when it maintains useful behavior within stated constraints.

Define a decision transaction with immutable identifiers:

$$\mathcal{T}_t = (i_o, i_s, i_q, i_m, i_c, i_e, \tau_o, \tau_d, \tau_u, y_t, S_t, d_t), \quad (13.1)$$

Terms: i_o observation ID; i_s state-estimator revision; i_q task/instruction ID; i_m model revision; i_c robot configuration; i_e monitor envelope; τ_o, τ_d, τ_u observation, decision, execution times; y_t raw model output; S_t, d_t monitor score and decision.

This transaction lets an incident be reproduced. Without revision and timing, identical model weights can produce different physical behavior.

13.2 Reference architecture

An implementation-centered architecture has two planes.

The **real-time plane** performs sensing, state estimation, policy inference, authorization, control, and fallback. The **learning and operations plane** stores episodes, curates data, trains and evaluates models, approves releases, monitors fleets, and manages rollback. Do not let an experimental training job write directly into the active robot model location.

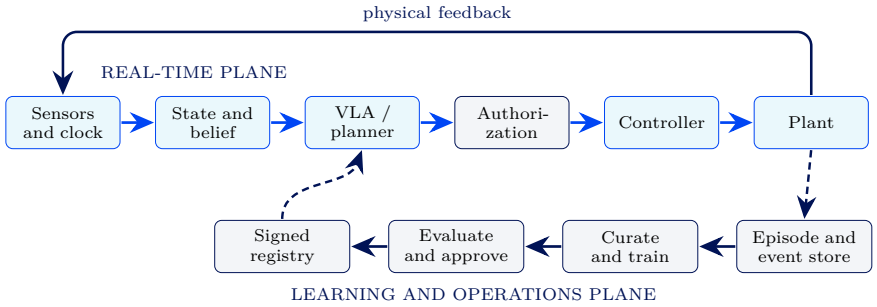


Figure 13.1: *Real-time execution is separated from learning and release control.*

WAM or world-model services can sit beside the policy for reranking; VLM services can provide grounded state; JEPA encoders can be shared. Every service returns revision, timestamps, and schema.

13.3 Latency and real-time budgets

For deadline T_D , a sequential path must satisfy

$$T_{sense} + T_{sync} + T_{pre} + T_{infer} + T_{plan} + T_{auth} + T_{comm} + T_{ctl} \leq T_D. \quad (13.2)$$

Use high-percentile and worst-case measurements. Mean latency is insufficient because queuing and memory pressure create tails.

If arrivals are approximately Poisson with rate λ and service rate μ , even an ideal M/M/1 queue has mean wait $W_q = \lambda / [\mu(\mu - \lambda)]$. As utilization $\rho = \lambda / \mu$ approaches one, wait grows rapidly. Real inference is not M/M/1, but the lesson holds: provision headroom and bound queues. Drop stale requests rather than execute them late.

End-to-end age matters more than server time. Let observation age at execution be

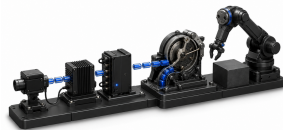
$$A_t = \tau_t^{exec} - \tau_t^{sensor}. \quad (13.3)$$

Monitor its distribution. A fast model behind a slow video decoder can be operationally slow.

Action chunks trade inference frequency for open-loop horizon. If policy period is T_p , inference time T_i , and executed prefix length h , a necessary steady-state condition for synchronous inference is $T_i \leq hT_p$. Asynchronous systems also need response freshness and buffer reserves.

PHYSICAL FAILURE MODE - Latency overrun

Correct components can form an unsafe loop when end-to-end latency exceeds the plant's tolerance. Budget capture, network, inference, monitoring, and control at p95 and p99, not only at the mean.



EXERCISE 13.1 - End-to-end success and reliability budgets

Suppose perception, planning, authorization, and control each succeed with conditional probabilities 0.97, 0.96, 0.995, and 0.99. Under a simplifying independence assumption, end-to-end success is their product: $0.97 \times 0.96 \times 0.995 \times 0.99 \approx 0.914$. Four apparently strong components yield only 91.4%. The product is not a field prediction because failures share causes such as lighting, timing, and configuration. It is a budget diagnostic: to reach 97%, the team must improve multiple stages or add recovery. Dependability engineering therefore tracks common-cause failures, degraded modes, and repair, not only component accuracy [95, 96].

13.4 Compute placement

Choose edge, local server, or cloud by deadline, bandwidth, availability, privacy, thermal power, and update needs.

- ▶ **on-device:** low network risk, strict memory/power, difficult thermal and model updates;
- ▶ **local edge server:** larger model, bounded local network, shared-resource contention;
- ▶ **cloud:** elastic compute and centralized ops, variable network and data governance;

- **hybrid**: small local fallback and control, remote semantic or planning model.

A robust hybrid keeps minimum safe behavior local. Network loss must not leave actuators holding the last learned action indefinitely. Define lease time T_{lease} ; commands older than the lease expire.

Memory optimization includes lower resolution, patch pruning, quantization, distillation, LoRA merge, cache control, and smaller action expert. Benchmark the entire processor-model-decoder pipeline with realistic cameras and concurrency. Quantize after identifying sensitive layers; action and normalization heads can remain higher precision.

EXERCISE 13.2 - Fast-path schedulability for a humanoid action expert

At 50 Hz the command deadline is 20 ms. Suppose p95 action-head inference is 10.5 ms, state packaging 1.8 ms, physical monitor 3.2 ms, communication 1.5 ms, and controller preparation 1.0 ms. Sequential execution totals 18.0 ms, leaving 2.0 ms of margin. A 4 ms scheduling disturbance misses the deadline even though average inference is acceptable.

At 5 Hz, a semantic update may take 72 ms and still meet its own 200 ms period, provided the action expert can safely use cached context. The cache needs an age limit and invalidation rule. A new person entering the workspace cannot wait for the next semantic refresh if the fast safety path already observes the event.

13.5 Model compression and numerical acceptance

Post-training quantization maps weight w to integer q :

$$q = \text{clip}(\text{round}(w/s) + z, q_{min}, q_{max}), \quad \hat{w} = s(q - z). \quad (13.4)$$

Terms: $s > 0$ is scale; z zero point; $\hat{w}$ dequantized value. Per-channel scales reduce error.

Numerical similarity must be measured at action output and behavior. Compare log-its/flows, normalized actions, physical actions, saturation decisions, monitor scores, and closed-loop trials. A small latent difference can cross a gripper threshold.

Distillation trains student p_s from teacher p_t :

$$\mathcal{L} = \lambda_{data} \mathcal{L}_{action}(p_s, A) + \lambda_{KD} T^2 D_{KL}(p_t^T \| p_s^T), \quad (13.5)$$

Terms: T is softening temperature for token distributions; continuous action experts can distill trajectories or vector fields. Teacher error is inherited, so retain real labels and physical tests.

13.6 Configuration and artifact integrity

Version at least:

- ▶ dataset and split manifests;
- ▶ model weights/adapters and code;
- ▶ tokenizer, image processor, and prompt template;
- ▶ normalization statistics and action schema;
- ▶ robot calibration, controller, firmware, and hardware revision;
- ▶ monitor envelope and thresholds;
- ▶ evaluation suite and scenario seeds;
- ▶ deployment container and drivers.

Define compatibility matrix. Model m is activatable only if

$$C(m, r, e, v) = 1, \quad (13.6)$$

where r is robot configuration, e envelope, and v runtime version. Compute compatibility in a trusted launcher. Refuse partial match.

Sign release manifests and verify hashes at startup. Restrict who can promote artifacts. Data poisoning and compromised checkpoints can induce targeted physical behavior; model supply chain is part of safety.

PHYSICAL FAILURE MODE - Configuration mismatch

Correct weights with the wrong decoder, monitor envelope, robot configuration, or calibration are a different system. Bind every release to immutable revisions and explicit compatibility checks.



13.7 Observability

Logs should answer: what did the system see, believe, propose, authorize, execute, and observe afterward? Store enough for replay without violating privacy or overwhelming bandwidth.

Core signals include:

- ▶ sensor and state freshness, missing rates, calibration revision;
- ▶ model input/output summaries and action distributions;



- ▶ inference, queue, authorization, and control latencies;
- ▶ uncertainty, OOD, monitor scores, active conditions;
- ▶ commanded, filtered, and measured action;
- ▶ task progress, fallback, intervention, collision/near-miss;
- ▶ compute utilization, temperature, memory, restarts, and network loss.

Use event-triggered full capture around anomalies and sampled capture during nominal operation. Preserve pre-event ring buffer. Privacy-sensitive images can be retained locally with derived telemetry centrally, but incident analysis may require controlled access to raw evidence.

Detect drift in feature or residual distribution. Population Stability Index for binned reference fractions p_b and current fractions q_b is

$$\text{PSI} = \sum_b (q_b - p_b) \log \frac{q_b + \epsilon}{p_b + \epsilon}. \quad (13.7)$$

PSI is a heuristic, not a universal threshold. Track domain-specific shifts and performance labels when available. A drift alert should trigger investigation, not automatic retraining.

13.8 Testing pyramid

Physical AI needs a test pyramid:

1. **unit**: transforms, normalization, action decode, masks, bounds;
2. **component**: model golden inputs, calibration, latency, OOD;
3. **contract**: sensor/model/controller schemas and version compatibility;
4. **replay**: logged episodes, counterfactual perturbations, previous incidents;
5. **simulation**: closed loop, domain randomization, faults, rare scenarios;
6. **hardware-in-the-loop**: real controllers/sensors with constrained plant;
7. **guarded hardware**: reduced energy and supervised trials;
8. **shadow/canary**: observe or limited deployment before promotion.

A release gate is a conjunction, not a weighted average:

$$G = G_{\text{schema}} \wedge G_{\text{offline}} \wedge G_{\text{closed}} \wedge G_{\text{latency}} \wedge G_{\text{physical}} \wedge G_{\text{fallback}} \wedge G_{\text{security}}. \quad (13.8)$$

One failed hard gate blocks promotion. Soft metrics rank candidates only after hard gate pass.

Use metamorphic tests when exact labels are hard. Examples: changing unrelated background should not change action substantially; transforming camera and state

consistently should transform action equivariantly; duplicating an instruction should not invert it; delaying observation should increase staleness response; swapping robot configuration should fail compatibility.

13.9 Fault injection

Inject faults into sensors, timing, models, communications, controllers, and environment:

- ▶ frozen, dropped, reordered, or corrupted frames;
- ▶ timestamp offset and jitter;
- ▶ calibration perturbation and camera swap;
- ▶ action scaling/sign/schema mismatch;
- ▶ actuator lag, saturation, and stuck joint;
- ▶ world-model mode shift and action-state mismatch;
- ▶ network timeout, duplicate response, and server overload;
- ▶ model confidence inflation and monitor unavailability.

For each fault specify expected detection, maximum detection time, safe response, and recovery. A test passes only if the system enters the intended state, not merely if an alert appears.

Failure-mode coverage is

$$C_F = \frac{\sum_i w_i \mathbf{1}\{\text{fault } i \text{ detected and handled}\}}{\sum_i w_i}, \quad (13.9)$$

with criticality weights w_i . Publish the unweighted list too.

13.10 Human roles and operating modes

Define authority across modes: development, teach, autonomous, supervised, degraded, emergency stop, maintenance. The robot should expose current mode, active model, and who can command it. Language instructions from users are untrusted input unless authenticated and authorized.

Human intervention interfaces need predictable latency and control transfer. A remote operator requires situation awareness; control handoff after network delay can conflict with autonomy. Use explicit ownership token and transition state.

An emergency stop is independent of learned software and should remove or control energy appropriately. Restart should require state reinitialization and possibly human inspection; do not resume an old action chunk.

13.11 Domain-specific differences

The same stack adapts across domains:

Manipulation: contact, calibration, gripper state, collision, object identity, and recovery dominate. Action chunks and multimodal policies are valuable.

Mobile robots: localization, dynamic obstacles, geofences, braking distance, and route-level planning dominate. A local safety controller must survive remote planner loss.

Autonomous vehicles: sensor redundancy, prediction, trajectory planning, rules, long-tail scenarios, and rigorous operational design domain are central. World models supplement but do not replace validated simulation and safety cases.

Drones: attitude control and stability run much faster than foundation-model inference. Keep flight control local; high-level Physical AI supplies waypoint/trajectory within reachable and energy envelopes.

Humanoids: whole-body balance, contact mode, high action dimension, and fall consequence require hierarchical control and rapid fallback. Semantic reasoning must not bypass balance control.

Medical and assistive systems: human physiology, consent, clinical workflow, regulation, and fail-safe interaction require stricter evidence and oversight. Educational models are not medical authorization.

13.12 Continuous learning without uncontrolled change

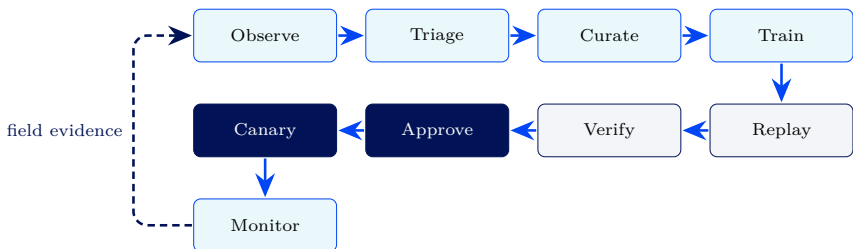


Figure 13.2: *Production learning remains observable, gated, and reversible before release.*

Do not let a deployed robot update core policy online unless the adaptation mechanism itself is bounded and verified. Most systems should collect data online and update offline under release control. Personalization can use limited adapters with norm, scope, and rollback bounds.

For new data slice D_n , mix replay D_r and target data:

$$\mathcal{L} = \mathbb{E}_{D_n} \ell + \lambda_{replay} \mathbb{E}_{D_r} \ell + \lambda_{anchor} \|\theta - \theta_0\|_2^2. \quad (13.10)$$

Terms: θ_0 is previous release; replay and anchoring reduce forgetting but cannot guarantee behavior preservation. Full regression testing is required.

Label incident causes before retraining. A controller bug should not be “fixed” by teaching the model to compensate for corrupted units.

13.13 A build roadmap

For a new Physical AI product:

1. define task, operational domain, physical limits, and success/risk metrics;
2. implement controller, deterministic safety limits, and logging before learned autonomy;
3. collect aligned demonstrations and build state-only baseline;
4. add vision/VLM grounding and validate evidence;
5. train or adapt VLA with action contract and closed-loop evaluation;
6. add world model/JEPA/WAM only for a measured need: planning, data, or diagnosis;
7. implement authorization and fallback with fault injection;
8. deploy guarded, then shadow/canary, with immutable rollback;
9. operate a curated retraining loop and update the assurance evidence.

This roadmap prevents architecture from outrunning measurement. The most advanced model should enter only after its interface can be tested.

Recommended practice: complete Exercise 4 in the companion course bundle.

13.14 Completion standard

On completing the book and laboratories, a reader should be able to train or retrain a VLM/VLA policy stack, train an action-conditioned world model or JEPA-style predictor, understand when a WAM adds value, diagnose Physical AI hallucinations,

and engineer the data, mathematics, runtime monitoring, and deployment system around them.

The final competence is integrative: explain every tensor in physical terms, every loss in behavioral terms, every benchmark in operational terms, and every pass decision in terms of its assumptions.

ENGINEERING NOTE - Final design review

Ask: Can we reproduce this action from raw evidence? Can we state why it was allowed? Can we recover when it is wrong? Can we prove which versions ran? Can we measure whether updates improved both progress and integrity? If any answer is no, the model is not yet a deployable Physical AI system.

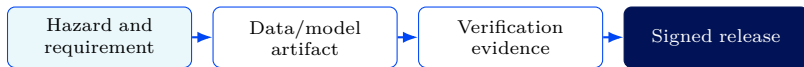


Figure 13.3: A release pipeline preserves traceability from requirement to evidence.

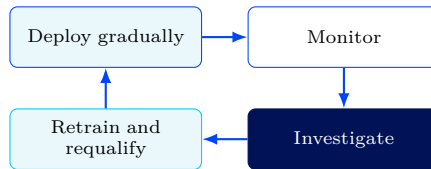


Figure 13.4: Operations close the lifecycle through observable field evidence.

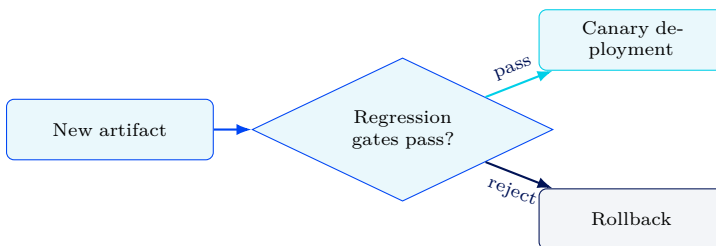


Figure 13.5: A model update is released only when the whole system case still holds.

Principal references

- [97] Organization-level AI risk management.
- [98] Current industrial robot safety requirements for robot manufacturers.
- [99] Current industrial robot application and cell integration requirements.
- [96] System-theoretic safety engineering.
- [100] Dependable computing foundations.
- [101] Runtime assurance through architectural simplicity.
- [79] Physical AI runtime authorization review.
- [102] Decision-making algorithms and uncertainty.
- [32] Open robot-learning lifecycle.
- [95] Dependability and security taxonomy.



Closing

Physical AI is not one model class. It is a contract between learned representations and a world that keeps changing after training has stopped. VLMs contribute semantic perception; VLAs turn observations and language into action proposals; world models and JEPA systems predict; WAMs couple futures with action; controllers, monitors, and fallback logic determine what is allowed to reach hardware.

If there is one principle I hope you carry from this book, it is this: do not judge the intelligence of a physical system by the fluency of its model output. Judge it by the quality of the loop. Can the system state what it observed, explain the frame and time of its decision, measure when its assumptions no longer hold, and recover without hiding the failure?

Larger models will arrive, and many of the names in this edition will change. The engineering discipline should remain: build the complete loop, measure what it actually does, and preserve useful behavior when prediction meets an untidy physical world.



ArtificialGate

APPENDIX A

Mathematical Foundations

A.1 Probability Background for Physical AI

Physical AI is built around uncertain state, incomplete observation, stochastic action, and distribution shift. Probability is therefore not an optional language for confidence scores. It is the mathematical layer that connects sensors to beliefs, demonstrations to learned policies, predicted futures to risk, and model outputs to authorization. This appendix supplies the probability needed throughout the book. It is compact enough to use as a bridge, but explicit enough that later equations do not depend on hidden notation. A reader wanting full proofs and a broader statistical treatment should use a probabilistic machine-learning text in parallel [6].

Probability spaces, events, and random variables

A probability model is a triple $(\Omega, \mathcal{F}, \mathbb{P})$. The **sample space** Ω contains possible outcomes. The **event set** $\mathcal{F}$ is a sigma-algebra: it contains Ω , is closed under complements, and is closed under countable unions. The probability measure $\mathbb{P}$ assigns every event $A \in \mathcal{F}$ a number in $[0, 1]$ such that

$$\mathbb{P}(\Omega) = 1, \quad \mathbb{P}(A) \geq 0, \quad \mathbb{P}\left(\bigcup_i A_i\right) = \sum_i \mathbb{P}(A_i) \quad (\text{A.1})$$

for pairwise disjoint events A_i . These axioms distinguish an event probability from an arbitrary model score.

A random variable $X : \Omega \rightarrow \mathcal{X}$ maps an outcome to a value. A random vector $\mathbf{X} \in \mathbb{R}^d$ maps it to d jointly distributed values. Uppercase letters denote random quantities; lowercase x denotes a realized value. The support $\mathcal{S}_X$ is the set of values that can occur under the model.

For a discrete variable, the probability mass function is $p_X(x) = \mathbb{P}(X = x)$ and $\sum_{x \in S_X} p_X(x) = 1$. For a continuous variable, the density satisfies

$$\mathbb{P}(X \in A) = \int_A p_X(x) dx, \quad \int_{S_X} p_X(x) dx = 1. \quad (\text{A.2})$$

The value of a density can exceed one; probability is obtained only after integration. The cumulative distribution function is $F_X(x) = \mathbb{P}(X \leq x)$. Its α -quantile is

$$q_\alpha = \inf\{x : F_X(x) \geq \alpha\}, \quad 0 < \alpha < 1. \quad (\text{A.3})$$

Quantiles are used for latency budgets, prediction intervals, and tail-risk thresholds. They must be estimated from the distribution that will actually be encountered.

Joint, marginal, and conditional distributions

Two variables are described jointly by $p_{X,Y}(x, y)$. Marginalization removes an unobserved variable:

$$p_X(x) = \int p_{X,Y}(x, y) dy \quad (\text{A.4})$$

for continuous Y , with a sum replacing the integral when Y is discrete. The conditional distribution is

$$p(x | y) = \frac{p(x, y)}{p(y)}, \quad p(y) > 0, \quad (\text{A.5})$$

so the product rule is $p(x, y) = p(x | y)p(y)$. Repeated application gives the chain rule

$$p(x_{1:n}) = \prod_{i=1}^n p(x_i | x_{1:i-1}), \quad (\text{A.6})$$

where $x_{1:i-1}$ is the sequence before element i . Autoregressive models are direct parameterizations of this identity.

The law of total probability and Bayes' rule are

$$p(y) = \int p(y | x)p(x) dx, \quad (\text{A.7})$$

$$p(x | y) = \frac{p(y | x)p(x)}{\int p(y | x')p(x') dx'}. \quad (\text{A.8})$$

Here $p(x)$ is the prior, $p(y | x)$ the likelihood of the observation, $p(x | y)$ the posterior, and $p(y)$ the evidence or marginal likelihood. The denominator normalizes the posterior; omitting it is acceptable only when writing proportionality in x .

Variables X and Y are independent when $p(x, y) = p(x)p(y)$. They are conditionally independent given Z , written $X \perp\!\!\!\perp Y \mid Z$, when

$$p(x, y \mid z) = p(x \mid z)p(y \mid z). \quad (\text{A.9})$$

Conditional independence is the basis of state-space and graphical models. Two sensors may be independent given the physical state but dependent marginally because both respond to the same state. Treating correlated observations as independent counts evidence twice and produces overconfident beliefs.

Expectation, variance, covariance, and conditioning

For an integrable function f , expectation is

$$\mathbb{E}[f(X)] = \int f(x)p_X(x) dx. \quad (\text{A.10})$$

Use a sum for a discrete variable. Linearity gives $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$, whether or not X and Y are independent. The mean is $\mu_X = \mathbb{E}[X]$, and the variance is

$$\text{Var}(X) = \mathbb{E}[(X - \mu_X)^2] = \mathbb{E}[X^2] - \mu_X^2. \quad (\text{A.11})$$

For a random vector, the covariance matrix is

$$\Sigma_X = \text{Cov}(X) = \mathbb{E}[(X - \mu)(X - \mu)^\top]. \quad (\text{A.12})$$

Its diagonal entries are marginal variances; entry (i, j) is the covariance between components X_i and X_j . Covariance has physical units equal to the product of component units. Correlation normalizes covariance:

$$\rho_{ij} = \frac{\text{Cov}(X_i, X_j)}{\sigma_i \sigma_j}, \quad (\text{A.13})$$

where $\sigma_i > 0$ is the standard deviation of X_i . Zero covariance does not generally imply independence.

Conditional expectation $\mathbb{E}[X \mid Y]$ is a random function of Y . Two identities are essential for belief estimation and uncertainty decomposition:

$$\mathbb{E}[X] = \mathbb{E}_Y[\mathbb{E}[X \mid Y]], \quad (\text{A.14})$$

$$\text{Var}(X) = \mathbb{E}_Y[\text{Var}(X \mid Y)] + \text{Var}_Y(\mathbb{E}[X \mid Y]). \quad (\text{A.15})$$

The first is the law of total expectation. The second is the law of total variance: total spread equals expected conditional spread plus spread of conditional means.

Distributions used repeatedly in the book

The following families cover most interfaces in Physical AI:

- ▶ **Bernoulli:** for a binary event $X \in \{0, 1\}$, $p(x) = \rho^x(1 - \rho)^{1-x}$, with success probability $\rho \in [0, 1]$.
- ▶ **Categorical:** for one of K classes, $p(X = k) = \pi_k$, where $\pi_k \geq 0$ and $\sum_{k=1}^K \pi_k = 1$. Action tokens and discrete modes use this form.
- ▶ **Gaussian:** $X \sim \mathcal{N}(\mu, \sigma^2)$ has density

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x - \mu)^2}{2\sigma^2}\right].$$

The parameters μ and $\sigma^2 > 0$ are mean and variance.

- ▶ **Multivariate Gaussian:** $X \sim \mathcal{N}(\mu, \Sigma)$ in d dimensions has density

$$p(\mathbf{x}) = \frac{\exp[-\frac{1}{2}(\mathbf{x} - \mu)^\top \Sigma^{-1}(\mathbf{x} - \mu)]}{(2\pi)^{d/2} |\Sigma|^{1/2}},$$

where $\Sigma > 0$ is covariance and $|\Sigma|$ its determinant. The quadratic term is squared Mahalanobis distance.

- ▶ **Mixture:** $p(x) = \sum_{k=1}^K \pi_k p_k(x)$, with nonnegative weights summing to one. Mixtures preserve alternative grasps, routes, and contact modes that a single mean can erase.

Beta and Dirichlet distributions place uncertainty over Bernoulli and categorical probabilities. Heavy-tailed families such as Student's t can represent outliers more honestly than a Gaussian. A distribution family is a modeling assumption, not a property established by the tensor shape.

Gaussian conditioning and sensor fusion

Partition a jointly Gaussian vector as

$$\begin{bmatrix} X \\ Y \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_X \\ \mu_Y \end{bmatrix}, \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{YX} & \Sigma_{YY} \end{bmatrix}\right). \quad (\text{A.16})$$

Then the conditional distribution of X given $Y = \mathbf{y}$ is Gaussian with

$$\mathbb{E}[X | \mathbf{y}] = \mu_X + \Sigma_{XY} \Sigma_{YY}^{-1} (\mathbf{y} - \mu_Y), \quad (\text{A.17})$$

$$\text{Cov}(X | \mathbf{y}) = \Sigma_{XX} - \Sigma_{XY} \Sigma_{YY}^{-1} \Sigma_{YX}. \quad (\text{A.18})$$

The cross-covariance Σ_{XY} determines how a measurement changes the state estimate. In software, solve a linear system rather than forming an explicit inverse. These equations produce the Kalman correction under a linear-Gaussian state-space model. They do not represent multiple contact hypotheses unless the model is extended beyond one Gaussian.

ENGINEERING NOTE - Numerical Bayesian fusion

Suppose a scalar position prior is $X \sim \mathcal{N}(1.00, 0.04)$ m and an independent measurement is $Y | X = x \sim \mathcal{N}(x, 0.09)$ m. Precision is inverse variance, so posterior variance and mean are

$$\sigma_{\text{post}}^2 = (0.04^{-1} + 0.09^{-1})^{-1} = 0.0277,$$

$$\mu_{\text{post}} = 0.0277 \left(\frac{1.00}{0.04} + \frac{1.30}{0.09} \right) = 1.092 \text{ m}.$$

The estimate moves toward the 1.30 m measurement but remains closer to the more precise prior. Independence, correct variances, and common units and frames are required for this calculation.

Likelihood, estimation, and posterior prediction

For data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, a conditional model $p_{\theta}(y | x)$ assigns likelihood

$$L(\theta; \mathcal{D}) = \prod_{i=1}^N p_{\theta}(y_i | x_i), \quad \ell(\theta; \mathcal{D}) = \sum_{i=1}^N \log p_{\theta}(y_i | x_i), \quad (\text{A.19})$$

under the stated conditional independence assumption. Here θ is a parameter, L is likelihood, and ℓ is log-likelihood. Maximum-likelihood and maximum-a-posteriori estimates are

$$\hat{\theta}_{\text{ML}} = \arg \max_{\theta} \ell(\theta; \mathcal{D}), \quad (\text{A.20})$$

$$\hat{\theta}_{\text{MAP}} = \arg \max_{\theta} \{\ell(\theta; \mathcal{D}) + \log p(\theta)\}. \quad (\text{A.21})$$

MAP adds the log prior $p(\theta)$. A regularizer often corresponds to a negative log prior, but only under an explicit probabilistic interpretation.

Full Bayesian inference retains parameter uncertainty:

$$p(\theta \mid \mathcal{D}) \propto p(\mathcal{D} \mid \theta)p(\theta), \quad p(y_* \mid x_*, \mathcal{D}) = \int p(y_* \mid x_*, \theta)p(\theta \mid \mathcal{D}) d\theta. \quad (\text{A.22})$$

The second expression is the posterior predictive distribution. A point estimate substitutes one $\hat{\theta}$ and therefore omits parameter uncertainty. For action selection under uncertainty, a Bayes decision minimizes posterior expected loss:

$$a^*(o) = \arg \min_a \mathbb{E}[L(a, X) \mid o], \quad (\text{A.23})$$

where $L(a, X)$ is the consequence of action a when the physical state is X . The most probable state need not yield the lowest-risk action.

Entropy, cross-entropy, KL divergence, and information

For a discrete distribution, entropy is

$$H(X) = - \sum_x p(x) \log p(x). \quad (\text{A.24})$$

It measures average uncertainty in nats when the natural logarithm is used. Cross-entropy and Kullback-Leibler divergence are

$$H(p, q) = -\mathbb{E}_{X \sim p}[\log q(X)], \quad (\text{A.25})$$

$$D_{\text{KL}}(p \parallel q) = \mathbb{E}_{X \sim p} \left[\log \frac{p(X)}{q(X)} \right] = H(p, q) - H(p). \quad (\text{A.26})$$

KL divergence is nonnegative but is not symmetric and is not a metric. Minimizing empirical negative log-likelihood minimizes an estimate of cross-entropy. The result depends on model class, data distribution, and optimization; a low token loss does not imply physical validity.

Mutual information is

$$I(X; Y) = D_{\text{KL}}(p(x, y) \parallel p(x)p(y)). \quad (\text{A.27})$$

It is zero exactly when the variables are independent under the modeled distribution. Conditional mutual information replaces each distribution by its version conditioned on Z . These quantities explain information bottlenecks and predictive representations, but estimating them in high dimensions is itself a statistical problem.

Sampling, Monte Carlo, dependence, and concentration

If $X_1, \dots, X_M$ are independent samples from p , the Monte Carlo estimator

$$\hat{\mu}_M = \frac{1}{M} \sum_{i=1}^M f(X_i) \quad (\text{A.28})$$

is unbiased for $\mu = \mathbb{E}_p[f(X)]$ when the expectation exists. Its variance is $\text{Var}(f(X))/M$. The law of large numbers gives $\hat{\mu}_M \rightarrow \mu$; the central limit theorem gives

$$\sqrt{M}(\hat{\mu}_M - \mu) \xrightarrow{d} \mathcal{N}(0, \sigma_f^2) \quad (\text{A.29})$$

under standard finite-variance conditions, where $\sigma_f^2 = \text{Var}(f(X))$. These asymptotic results do not make rare failures observable in a small test set.

Robot trajectories are temporally correlated. For a stationary scalar sequence with lag- k autocorrelation ρ_k , a useful approximation is

$$N_{\text{eff}} \approx \frac{N}{1 + 2 \sum_{k \geq 1} \rho_k}. \quad (\text{A.30})$$

The effective sample size can be far smaller than the number of frames. Confidence intervals that treat every video frame as independent are then too narrow.

For independent bounded observations $Z_i \in [a, b]$, Hoeffding's inequality states

$$\mathbb{P}\{|\bar{Z} - \mathbb{E}Z| \geq \epsilon\} \leq 2 \exp\left(-\frac{2N\epsilon^2}{(b-a)^2}\right). \quad (\text{A.31})$$

This is a finite-sample bound, but it requires independence and boundedness. Distribution-free does not mean assumption-free. For K successes in N independent Bernoulli trials, $\hat{p} = K/N$ estimates the success probability. A two-sided Wilson confidence interval with nominal coverage $1 - \delta$ is

$$\frac{\hat{p} + z^2/(2N) \pm z\sqrt{\hat{p}(1-\hat{p})/N + z^2/(4N^2)}}{1 + z^2/N}, \quad z = \Phi^{-1}(1 - \delta/2), \quad (\text{A.32})$$

where Φ^{-1} is the standard-normal quantile function. Unlike the elementary Wald interval $\hat{p} \pm z\sqrt{\hat{p}(1-\hat{p})/N}$, Wilson behaves better for small samples and rates near zero or one. A frequentist confidence interval describes the long-run coverage of the procedure, not the posterior probability that this particular interval contains p . When

trials share scenes, operators, or trajectories, resample or cluster at that independent unit rather than at the frame level.

When samples come from proposal $q(x)$ rather than target $p(x)$, importance sampling uses weights $w_i = p(X_i)/q(X_i)$:

$$\mathbb{E}_p[f(X)] = \mathbb{E}_q[w(X)f(X)], \quad \hat{\mu}_{\text{SN}} = \sum_{i=1}^M \bar{w}_i f(X_i), \quad (\text{A.33})$$

where $\bar{w}_i = w_i / \sum_j w_j$. The diagnostic $M_{\text{eff}} = 1 / \sum_i \bar{w}_i^2$ reveals weight collapse. Importance weighting fails when q has no support where p places mass.

Transformations, reparameterization, and uncertainty propagation

For an invertible differentiable transformation $Y = g(X)$, change of variables gives

$$p_Y(y) = p_X(g^{-1}(y)) \left| \det \frac{\partial g^{-1}(y)}{\partial y} \right|. \quad (\text{A.34})$$

The determinant accounts for local volume change. Normalizing flows use this identity; omitting it produces a quantity that is not a normalized density.

A Gaussian sample can be reparameterized as

$$\mathbf{Z} = \boldsymbol{\mu} + L\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I), \quad LL^\top = \Sigma. \quad (\text{A.35})$$

This separates parameter-free noise from differentiable parameters and enables low-variance pathway gradients in variational models.

For $\mathbf{Y} = g(\mathbf{X})$ and a distribution concentrated near $\boldsymbol{\mu}_X$, first-order covariance propagation is

$$\Sigma_Y \approx J_g \Sigma_X J_g^\top, \quad J_g = \left. \frac{\partial g}{\partial \mathbf{x}} \right|_{\boldsymbol{\mu}_X}. \quad (\text{A.36})$$

The approximation can fail for strong nonlinearity, multimodality, discontinuous contact, or large rotational uncertainty. Monte Carlo propagation or a representation respecting $SO(3)$ and $SE(3)$ may then be required.

Stochastic processes and state-space factorization

A stochastic process is a collection $\{X_t\}_{t \geq 0}$ indexed by time. A controlled Markov process assumes

$$p(x_{t+1} \mid x_{0:t}, a_{0:t}) = p(x_{t+1} \mid x_t, a_t). \quad (\text{A.37})$$

Under this assumption, a trajectory factorizes as

$$p(x_{0:T}, o_{1:T} \mid a_{0:T-1}) = p(x_0) \prod_{t=0}^{T-1} p(x_{t+1} \mid x_t, a_t) \prod_{t=1}^T p(o_t \mid x_t). \quad (\text{A.38})$$

The first factor is the initial-state distribution, the first product contains transition kernels, and the second contains observation likelihoods. A hidden-state model can be Markov even when image observations are not. History, filtering, or a learned information state is then required.

Stationarity means selected distributions do not change with absolute time; ergodicity justifies estimating certain ensemble quantities from a long trajectory. Neither property follows from logging data on one robot. Resets, wear, human intervention, and software updates can create nonstationarity that must be represented in splits and evaluation.

Aleatoric uncertainty, epistemic uncertainty, and calibration

Aleatoric uncertainty is variability retained even when model parameters are known, such as sensor noise or genuinely multiple valid actions. **Epistemic uncertainty** reflects uncertainty about the model, often from limited or shifted data. If Θ is uncertain under data $\mathcal{D}$, total predictive variance decomposes as

$$\text{Var}(Y \mid x, \mathcal{D}) = \mathbb{E}_{\Theta \mid \mathcal{D}}[\text{Var}(Y \mid x, \Theta)] + \text{Var}_{\Theta \mid \mathcal{D}}(\mathbb{E}[Y \mid x, \Theta]). \quad (\text{A.39})$$

The first term is expected conditional noise; the second is variation among model predictions. Ensembles approximate only part of this decomposition and can agree while sharing the same blind spot.

A binary confidence score S is calibrated when

$$\mathbb{P}(Y = 1 \mid S = s) = s \quad (\text{A.40})$$

for relevant score values s . Negative log-likelihood and the Brier score

$$\text{Brier} = \frac{1}{N} \sum_{i=1}^N (s_i - y_i)^2 \quad (\text{A.41})$$

are proper scoring rules for probabilistic predictions. Expected calibration error is a binned diagnostic,

$$\text{ECE} = \sum_{b=1}^B \frac{|I_b|}{N} |\text{acc}(I_b) - \text{conf}(I_b)|, \quad (\text{A.42})$$

where I_b contains examples in bin b . ECE depends on binning and can hide class- or hazard-specific error. Calibration must be checked under the deployment distribution and separately for consequential subgroups [34].

Expected risk, chance constraints, and tail risk

Expected risk is $R(a) = \mathbb{E}[L(a, X)]$, but a small mean can hide a severe tail. A chance constraint requires

$$\mathbb{P}\{g_j(X, a) \leq 0 \text{ for all } j\} \geq 1 - \alpha, \quad (\text{A.43})$$

where g_j are constraint functions and α is the tolerated violation probability. The union bound, $\mathbb{P}(\cup_j A_j) \leq \sum_j \mathbb{P}(A_j)$, permits allocation of a total failure budget across constraints, although the result can be conservative.

For a loss variable L and confidence level α , conditional value at risk has the optimization form

$$\text{CVaR}_\alpha(L) = \min_{\eta \in \mathbb{R}} \left\{ \eta + \frac{1}{1 - \alpha} \mathbb{E}[(L - \eta)_+] \right\}, \quad (\text{A.44})$$

where $(u)_+ = \max(u, 0)$. CVaR averages the upper tail beyond a learned threshold η . It is useful only when the loss scale, sampling process, and tail coverage correspond to deployment. Conformal prediction offers finite-sample marginal coverage under exchangeability, developed later in the derivation atlas [88].

ENGINEERING NOTE - Numerical Bayes check for a contact sensor

Let prior contact probability be $\mathbb{P}(C = 1) = 0.20$. A binary sensor reports high force with $\mathbb{P}(H = 1 \mid C = 1) = 0.90$ and false-positive probability $\mathbb{P}(H = 1 \mid C = 0) = 0.10$. Bayes' rule gives

$$\mathbb{P}(C = 1 \mid H = 1) = \frac{0.90(0.20)}{0.90(0.20) + 0.10(0.80)} = \frac{0.18}{0.26} = 0.692.$$

The positive reading raises belief from 0.20 to 0.692, not to 0.90. If the prior or false-positive rate changes with object, speed, or sensor wear, the posterior changes too.

Readiness checklist

Use this readiness check before Lesson 1 if probability is not already part of your working background:

- ▶ distinguish probability, density, likelihood, and confidence;
- ▶ marginalize a joint distribution and apply Bayes' rule;
- ▶ recognize independence assumptions in a factorization;
- ▶ compute mean, covariance, conditional expectation, and a Gaussian update;
- ▶ interpret NLL, entropy, cross-entropy, KL divergence, and mutual information;
- ▶ explain why correlated frames are not independent samples;
- ▶ propagate covariance through a local transformation;
- ▶ separate aleatoric from epistemic uncertainty and evaluate calibration;
- ▶ distinguish expected loss from a chance constraint or tail-risk objective.

This background supports the Bayesian filter in Lesson 2, behavioral-cloning likelihoods in Lesson 3, calibrated grounding in Lesson 4, stochastic VLA objectives in Lessons 5 and 6, latent world models in Lesson 7, predictive representations in Lesson 8, action-conditioned futures in Lesson 9, hallucination evidence in Lesson 11, and authorization guarantees in Lesson 12. State estimation and robotics-specific inference are treated in greater depth by Thrun et al. and Barfoot [2, 20].

A.2 Mathematical Derivation Atlas

This atlas collects the derivations that connect the model families used throughout the book. It is intentionally compact, but it does not hide assumptions. Each result states what is exact, what is a bound, and what depends on an empirical model.

1. Closed-loop learning as a finite-horizon control problem

Let physical state be $x_t \in \mathcal{X}$, observation $o_t \sim G(\cdot \mid x_t)$, action $a_t \sim \pi(\cdot \mid I_t)$, and transition $x_{t+1} \sim P(\cdot \mid x_t, a_t)$. The information state I_t may be a filtered state, belief, or observation-action history. For horizon H , define

$$J_H(\pi) = \mathbb{E}_{\tau \sim p_\pi} \left[\sum_{t=0}^{H-1} c_t(x_t, a_t) + c_H(x_H) \right], \quad (\text{A.45})$$

where $p_\pi(\tau)$ is the trajectory distribution induced jointly by P, G, π , the initial distribution, and the timing/interface implementation.

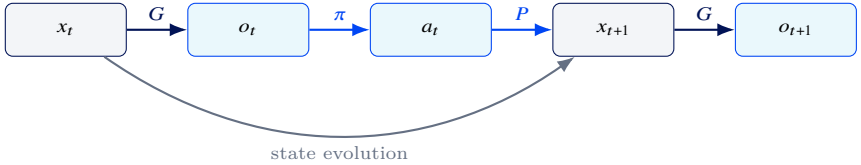


Figure A.1: The learned policy changes the distribution on which its next decision is evaluated.

Let d_π^t be the state occupancy at time t , and let $A_t^{\pi_E}(x, a) = Q_t^{\pi_E}(x, a) - V_t^{\pi_E}(x)$ be the finite-horizon advantage relative to expert π_E . The performance-difference identity is

$$J_H(\pi) - J_H(\pi_E) = \sum_{t=0}^{H-1} \mathbb{E}_{x \sim d_\pi^t, a \sim \pi(\cdot \mid x)} A_t^{\pi_E}(x, a). \quad (\text{A.46})$$

Derivation. Insert and subtract $V_{t+1}^{\pi_E}(x_{t+1})$ in every stage cost, take conditional expectation, and telescope from $V_0^{\pi_E}$ to the terminal value. The identity is exact; the difficulty is that expectations use learner occupancy d_π^t , not expert occupancy.

If $|A_t^{\pi_E}| \leq A_{\max}$ and the probability of a consequential deviation under d_{π}^t is ϵ_t , then

$$|J_H(\pi) - J_H(\pi_E)| \leq A_{\max} \sum_{t=0}^{H-1} \epsilon_t. \quad (\text{A.47})$$

Behavioral cloning estimates error mainly under $d_{\pi_E}^t$. A deviation can change all later occupancies, producing the familiar worst-case $O(H^2\epsilon)$ scaling; interactive imitation reduces the mismatch by labeling learner-visited states [26].

For a precise visual comparison, the figure plots the reference curves $b_{\text{linear}}(H) = H$ and $b_{\text{quadratic}}(H) = H(H+1)/2$. They show growth rates only; constants and realized losses depend on the policy, state distribution, and task.

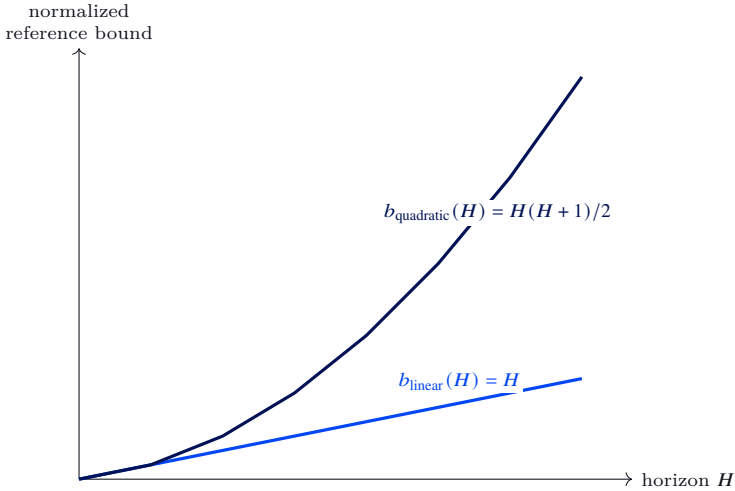


Figure A.2: Linear and quadratic reference growth. These are explicit scaling curves, not measured performance.

2. Kalman filtering as a linear-Gaussian belief update

Consider the linear stochastic system

$$\mathbf{x}_t = A\mathbf{x}_{t-1} + B\mathbf{u}_{t-1} + \mathbf{w}_{t-1}, \quad \mathbf{w}_{t-1} \sim \mathcal{N}(\mathbf{0}, Q), \quad (\text{A.48})$$

$$\mathbf{o}_t = H\mathbf{x}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, R). \quad (\text{A.49})$$



Assume the posterior at $t-1$ is Gaussian with mean $\hat{\mathbf{x}}_{t-1|t-1}$ and covariance $P_{t-1|t-1}$. The prediction step is

$$\hat{\mathbf{x}}_{t|t-1} = A\hat{\mathbf{x}}_{t-1|t-1} + B\mathbf{u}_{t-1}, \quad (\text{A.50})$$

$$P_{t|t-1} = AP_{t-1|t-1}A^\top + Q. \quad (\text{A.51})$$

The innovation, its covariance, and the Kalman gain are

$$\mathbf{r}_t = \mathbf{o}_t - H\hat{\mathbf{x}}_{t|t-1}, \quad (\text{A.52})$$

$$S_t = HP_{t|t-1}H^\top + R, \quad (\text{A.53})$$

$$K_t = P_{t|t-1}H^\top S_t^{-1}. \quad (\text{A.54})$$

The correction step is

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + K_t\mathbf{r}_t, \quad (\text{A.55})$$

$$P_{t|t} = (I - K_tH)P_{t|t-1}. \quad (\text{A.56})$$

Here A is the state-transition matrix; B maps control into state; H maps state into measurement space; Q and R are process- and measurement-noise covariances; P is estimation-error covariance; $\mathbf{r}_t$ is the measurement innovation; S_t is innovation covariance; and K_t weights the correction. The covariance expression above is exact algebraically; the Joseph form

$$P_{t|t} = (I - K_tH)P_{t|t-1}(I - K_tH)^\top + K_tRK_t^\top$$

is often preferable in finite-precision software because it better preserves symmetry and positive semidefiniteness.

Scalar numerical check. Let $A = H = 1$, $\hat{x}_{t-1|t-1} = 1.00$, $P_{t-1|t-1} = 0.04$, $Q = 0.01$, $R = 0.09$, and measurement $o_t = 1.30$, with no control input. Prediction gives $\hat{x}_{t|t-1} = 1.00$ and $P_{t|t-1} = 0.05$. Then $r_t = 0.30$, $S_t = 0.14$, and $K_t = 0.05/0.14 = 0.357$. Correction gives

$$\hat{x}_{t|t} = 1.00 + 0.357(0.30) = 1.107, \quad P_{t|t} = (1 - 0.357)(0.05) = 0.0321.$$

The estimate moves toward the measurement but does not copy it, because measurement variance exceeds predicted state variance. The posterior variance is smaller than either the predicted variance or the measurement variance.

Guarantee boundary. These equations are the exact Bayesian recursion for the declared linear-Gaussian model. They are not exact for nonlinear, multimodal, delayed, miscalibrated, or data-associated observations. Extended, unscented, ensemble, particle, and learned filters modify the approximation, but none removes the need to validate timestamps, observability, noise assumptions, and outliers [2, 20].

3. Variational world models and the sequence ELBO

Consider latent dynamics

$$p_{\theta}(z_{1:T}, o_{1:T} \mid a_{1:T-1}) = p(z_1) \prod_{t=1}^T p_{\theta}(o_t \mid z_t) \prod_{t=1}^{T-1} p_{\theta}(z_{t+1} \mid z_t, a_t). \quad (\text{A.57})$$

Introduce an inference model $q_{\phi}(z_{1:T} \mid o_{1:T}, a_{1:T-1})$. Jensen's inequality gives

$$\begin{aligned} \log p_{\theta}(o_{1:T} \mid a_{1:T-1}) &= \log \mathbb{E}_{q_{\phi}} \left[\frac{p_{\theta}(z, o \mid a)}{q_{\phi}(z \mid o, a)} \right] \\ &\geq \mathbb{E}_{q_{\phi}} [\log p_{\theta}(z, o \mid a) - \log q_{\phi}(z \mid o, a)] = \mathcal{L}_{\text{ELBO}}. \end{aligned} \quad (\text{A.58})$$

(A.59)

For a filtering factorization, the bound separates into reconstruction/prediction and transition regularization:

$$\mathcal{L}_{\text{ELBO}} = \sum_{t=1}^T \mathbb{E}_q \log p_{\theta}(o_t \mid z_t) - D_{\text{KL}}(q_{\phi}(z_1 \mid o_1) \parallel p(z_1)) \quad (\text{A.60})$$

$$- \sum_{t=1}^{T-1} \mathbb{E}_q D_{\text{KL}}(q_{\phi}(z_{t+1} \mid z_t, o_{t+1}, a_t) \parallel p_{\theta}(z_{t+1} \mid z_t, a_t)). \quad (\text{A.61})$$

The gap is exactly $D_{\text{KL}}(q_{\phi}(z \mid o, a) \parallel p_{\theta}(z \mid o, a))$. Maximizing the ELBO therefore maximizes likelihood only when the inference family can approach the true posterior and optimization succeeds [103]. A high ELBO does not establish controllability: the latent state may preserve pixels while discarding a small contact variable that determines action outcome.

ENGINEERING NOTE - A control-aware augmentation

Add reward, termination, constraint, inverse-action, or multi-step consistency heads only when their labels and semantics are explicit. The augmented objective changes which information the latent state is encouraged to retain; it does not prove that the representation is sufficient for every downstream controller.

4. Flow matching for continuous action generation

Let context be c , data action chunk $A_1 \sim p_{\text{data}}(\cdot \mid c)$, and base noise $A_0 \sim p_0$. Along the linear probability path

$$A_s = (1-s)A_0 + sA_1, \quad s \in [0, 1], \quad (\text{A.62})$$

the conditional target velocity is $U_s = A_1 - A_0$. Flow matching trains

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{c, A_0, A_1, s} \|v_\theta(A_s, s, c) - U_s\|_2^2. \quad (\text{A.63})$$

At inference, solve the ordinary differential equation

$$\frac{dA_s}{ds} = v_\theta(A_s, s, c), \quad A_{s=0} \sim p_0. \quad (\text{A.64})$$

Under a Lipschitz velocity field, an Euler solver with step $\Delta s = 1/K$ has global discretization error $O(\Delta s)$; higher-order solvers reduce numerical error but not model error [43]. The end-to-end action error can be decomposed conceptually as

$$\epsilon_{\text{action}} \leq \epsilon_{\text{representation}} + \epsilon_{\text{field}} + \epsilon_{\text{solver}} + \epsilon_{\text{decoder}}. \quad (\text{A.65})$$

This is a diagnostic decomposition, not a claim that the terms are statistically independent.

5. JEPA collapse and what the asymmetries accomplish

For context encoder f_θ , target encoder f_ξ , predictor g_θ , and mask m ,

$$\mathcal{L}_{\text{JEPA}} = \mathbb{E} \|g_\theta(f_\theta(C_m(x)), m) - \text{sg}(f_\xi(T_m(x)))\|_2^2. \quad (\text{A.66})$$

If both encoders and predictor output the same constant, the unconstrained objective admits a zero-loss collapsed solution. Stop-gradient removes a direct optimization path into the target; exponential-moving-average updates slow target motion; multi-block masking prevents trivial local copying; variance/covariance controls can explicitly penalize dimensional collapse. These mechanisms explain why training can avoid collapse, but they do not prove that the learned coordinates retain task-relevant dynamics. That must be tested with action prediction, future discrimination, intervention response, and closed-loop control [14, 57, 59].

6. Calibrated authorization and finite-sample coverage

Let nonconformity score $r_i = R(x_i, y_i)$ be computed on n exchangeable calibration examples. For target miscoverage α , choose

$$k = \lceil (n+1)(1-\alpha) \rceil, \quad q_\alpha = r_{(\min\{k, n\})}, \quad (\text{A.67})$$

where $r_{(j)}$ is the j -th order statistic. The conformal prediction set $C(x) = \{y : R(x, y) \leq q_\alpha\}$ satisfies marginal coverage

$$\mathbb{P}\{Y_{n+1} \in C(X_{n+1})\} \geq 1 - \alpha \quad (\text{A.68})$$

under exchangeability [88]. The guarantee is marginal, not conditional on every scene, robot, or hazard class. Distribution shift breaks the premise unless addressed by new calibration or a justified weighted method.

For constraints $g_j(\zeta) \leq 0$, define the admissible event $\mathcal{E}_j = \{\zeta : g_j(\zeta) \leq 0\}$. A joint chance constraint is

$$\mathbb{P}\left(\bigcap_{j=1}^m \mathcal{E}_j\right) \geq 1 - \alpha. \quad (\text{A.69})$$

Allocating α_j with $\sum_j \alpha_j \leq \alpha$ and enforcing each marginal violation probability below α_j is sufficient by the union bound, but can be conservative.

7. Why a control barrier inequality preserves a safe set

Let safe set $C = \{x : h(x) \geq 0\}$. Suppose closed-loop dynamics satisfy

$$\dot{h}(x(t)) \geq -\kappa h(x(t)), \quad \kappa > 0. \quad (\text{A.70})$$

Multiplying by $e^{\kappa t}$ gives

$$\frac{d}{dt} (e^{\kappa t} h(x(t))) = e^{\kappa t} (\dot{h} + \kappa h) \geq 0. \quad (\text{A.71})$$

Therefore $h(x(t)) \geq e^{-\kappa t} h(x(0)) \geq 0$ whenever $h(x(0)) \geq 0$. Forward invariance follows under the assumed model, regularity, feasible control, and continuous enforcement [91]. Discretization, delay, state-estimation error, and actuator saturation require explicit margins.

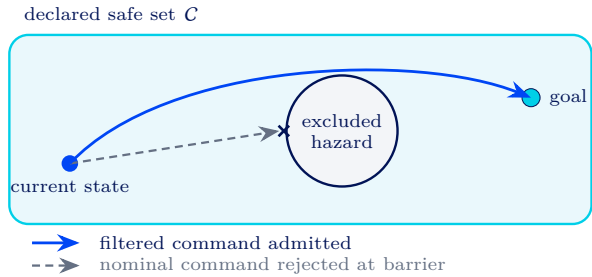


Figure A.3: A barrier filter redirects a nominal proposal around an excluded region while remaining inside the declared safe set.

Proof-status ledger

Result	Status	Main boundary
Performance difference	exact identity	Requires the true learner occupancy for evaluation.
Kalman recursion	exact Bayesian update under stated assumptions	Requires linear dynamics, Gaussian noise, correct timing, and known associations.
Sequence ELBO	rigorous lower bound	Tightness depends on posterior family and optimization.
Flow solver order	numerical-analysis result	Assumes regular learned velocity; excludes model bias.
JEPA anti-collapse	mechanism and empirical evidence	Does not certify semantic or controllable sufficiency.
Conformal coverage	finite-sample marginal guarantee	Requires exchangeability; not per-hazard conditional coverage.
Barrier invariance	theorem under stated dynamics	Requires feasible control, correct state, timing, and margins.

Table A.1: The book distinguishes theorem, bound, mechanism, and empirical claim.

APPENDIX B

Notation and Terminology

B.1 Mathematical Notation and Symbols

This guide collects the notation used in the lessons and mathematical appendices. It is designed as a lookup reference rather than a prerequisite reading assignment. Symbols are also defined at their first operational use, so readers can begin Lesson 1 and return here whenever a compact reminder is useful.

A letter can carry different local meanings in different model families. In particular, $A, B, C, H, K, M, S, T, r$, and η are intentionally overloaded in standard robotics and machine-learning notation. The tables below state the scope explicitly. Subscripts identify time, component, horizon, coordinate, loss, or gate; superscripts identify frame, domain, episode, candidate, model, or revision.

ENGINEERING NOTE - Reading rule

Interpret every expression together with its conditioning variables, units, coordinate frame, time index, and stated assumptions. Bold symbols denote vectors unless a table states otherwise. Uppercase calligraphic symbols usually denote sets, datasets, losses, or operators. A hat denotes an estimate or prediction, a bar an average, a tilde a transformed quantity, and a star an optimizer or selected candidate.

Typesetting, indices, and modifiers

Symbol	Meaning and scope
x, a, r, ℓ	Scalar quantity. The local text supplies units and meaning.
$\mathbf{x}, \mathbf{a}, \mathbf{o}$	Vector, printed in bold. Components use x_i , a_j , or o_k .
A, B, W, K, Σ	Matrix unless the local table identifies a count, set, sequence, or event.
$\mathcal{X}, \mathcal{A}, \mathcal{D}$	Set, action space, or dataset.
X, Y, Θ	Random quantity or random parameter; lowercase x, y, θ denotes a realization.
t, τ	Discrete decision index t , or a timestamp τ when used with a clock label.
i, j, k, m, b	Generic sample, coordinate, horizon, model, constraint, or bin index. Scope is local.
$x_{a:b}, x_{\leq t}$	Inclusive subsequence from a through b , or all elements through time t .
$\hat{x}, \bar{x}, \tilde{x}, x^\star$	Estimate or prediction, average, transformed/normalized value, and optimizer or selected value.
$x^\top, A^{-1}, A^H$	Transpose, inverse, and matrix power.
$\dot{x}, \ddot{x}, \dddot{x}$	First, second, and third derivatives with respect to physical time.
$\Delta x, \Delta^p x$	Increment or finite difference; p is difference order when present.

Common sets, operators, and relations

Symbol	Meaning and scope
$\mathbb{R}, \mathbb{R}^d$	Real numbers and d -dimensional real vector space.
$\ \mathbf{x}\ _p, \langle \mathbf{x}, \mathbf{y} \rangle$	p -norm and inner product; an omitted subscript is defined locally.
$ A , \#A$	Set cardinality or count. Absolute value is used when the argument is scalar.
$\Sigma, \Pi, \int$	Sum, product, and integral over the stated index or domain.
$\cup, \cap$	Union and intersection of events or sets.
$\in, \subseteq, \cap, \cup$	Membership, subset, intersection, and union.
$\wedge$	Logical conjunction. Every joined condition must hold.
$\leq, \geq, <, >$	Ordered comparison. Units must match on both sides.
$\sim, \approx, \propto$	Distributed as, approximately equal, and proportional to.
$\arg \min, \arg \max$	Argument that minimizes or maximizes an objective over the stated feasible set.
$\min, \max, \inf$	Minimum, maximum, and infimum values over the stated domain.
$\mathbf{1}\{E\}$	Indicator: one when event E occurs and zero otherwise.
$[x]_+ = \max(x, 0)$	Positive part.
$\text{diag}, \text{tr}, \det$	Diagonal matrix, matrix trace, and determinant.
$\nabla, \partial/\partial x, J_g$	Gradient, partial derivative, and Jacobian of mapping g .
$\text{clip}, \text{round}$	Clipping to declared bounds and numerical rounding.

Probability spaces and distributions

Symbol	Meaning and scope
$(\Omega, \mathcal{F}, \mathbb{P})$	Sample space, sigma-algebra of events, and probability measure.
$\mathbb{P}(E), \Pr(E)$	Probability of event E . Both forms appear in cited standard notation.
$p_X(x), p(x, y)$	Probability mass function, density, or joint density as stated.
$p(x y)$	Conditional distribution of x given y .
$p(x), p(y x),$ $p(x y), p(y)$	Prior, likelihood, posterior, and evidence in Bayes' rule.
$X \perp\!\!\!\perp Y Z$	Conditional independence of X and Y given Z .
$\mathcal{S}_X$	Support of random variable X .
$F_X(x), q_\alpha$	Cumulative distribution function and its α -quantile.
$\mathcal{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ .
$\mathbb{E}[X], \mathbb{E}_p[f(X)]$	Expectation, optionally under distribution p .
$\text{Var}(X), \text{Cov}(X, Y)$	Variance and covariance.
$\Sigma, \sigma, \rho_{ij}$	Covariance matrix, standard deviation, and correlation coefficient.
Φ, Φ^{-1}	Standard-normal cumulative distribution and quantile function.

Information, estimation, calibration, and risk

Symbol	Meaning and scope
$H(X), H(p, q)$	Entropy of X and cross-entropy between p and q .
$D_{\text{KL}}(p \ q)$	Kullback-Leibler divergence from q to p .
$I(X; Y)$	Mutual information.
$L(\theta; \mathcal{D}), \ell(\theta; \mathcal{D})$	Likelihood and log-likelihood of parameter θ on data $\mathcal{D}$.
$\hat{\theta}_{\text{ML}}, \hat{\theta}_{\text{MAP}}$	Maximum-likelihood and maximum-a-posteriori estimators.
N, M, B	Sample count, Monte Carlo/model count, or batch size according to the local expression.
N_{eff}	Effective sample size for correlated or importance-weighted samples.
$w_i, \bar{w}_i = p/q$	Importance weight and its normalized form.
Brier, BS	Brier score for probabilistic binary predictions.
ECE, I_b, S_b	Expected calibration error and the samples in calibration bin b .
$g_j(X, a) \leq 0$	Constraint j in a chance-constrained decision.
$\text{CVaR}_\alpha(L)$	Conditional value at risk of loss L at tail level α .
η in CVaR	Auxiliary loss threshold optimized in the CVaR representation.

Learning and optimization

Symbol	Meaning and scope
$\theta, \phi, \psi, \xi, \omega$	Parameter sets. Typical roles are main model, encoder/inference model, policy/inverse model, target model, and value model.
$f_\theta, g_\theta, h, \pi_\theta$	Learned encoder/dynamics function, predictor/decoder, observation mapping, and policy.
$\mathcal{L}, \ell, J$	Training objective, per-example loss, and performance/cost objective.
$\theta^\star$	Parameter value minimizing or maximizing the stated objective.
λ, λ_k	Nonnegative loss, regularization, risk, or constraint weights.
$\alpha, \beta, \gamma, \kappa$	Step/scale, regularization or momentum, discount, and decay/rate parameter as scoped locally.
$\epsilon > 0$	Numerical stabilizer, tolerance, approximation error, or perturbation radius.
$\mathbf{g}_k, \mathbf{m}_k, \mathbf{v}_k$	Minibatch gradient and first/second moment estimates in AdamW.
$\beta_1, \beta_2, \alpha_k$	AdamW moment decays and learning rate at update k .
$\odot$	Elementwise product.
$W_0, \Delta W, A, B, r$	Frozen weight, low-rank update, LoRA factors, and LoRA rank.
$N_{\text{LoRA}}, \mathcal{S}$	Number of trainable LoRA parameters and adapted layer set.
$\mathcal{L}_{\text{MSE}}, \mathcal{L}_{\text{NLL}}$	Mean-squared-error and negative-log-likelihood objectives.

Physical state, observations, actions, and time

Symbol	Meaning and scope
x_t, s_t	Physical/Markov state and recorded robot-environment state at decision t .
$o_t, I_{\leq t}$	Sensor observation and image/history context.
z_t, h_t	Learned latent state and deterministic recurrent state.
a_t, u_t	Learned/proposed action and executed low-level control input.
q, g	Language prompt/instruction or goal. Joint coordinates also use q only in explicitly robotic equations.
r, e, c	Proprioceptive history, embodiment descriptor, and optional context.
H, L, T, T_e	Action/prediction horizon, history length, total horizon, and episode length.
$\Delta t, f_p, T_D$	Physical time step, required policy frequency, and deadline.
$\tau_t, \tau_o, \tau_d, \tau_u$	Timestamp, or observation, decision, and execution timestamps.
$T_{\text{inf}}, T_{\text{net}}, T_{\text{mon}}, T_{\text{ctl}}$	Inference, network, monitor, and controller latency.
$T_{\text{sense}}, T_{\text{sync}}, T_{\text{pre}}$	Sensing, synchronization, and preprocessing latency.
$T_{\text{plan}}, T_{\text{auth}}, T_{\text{comm}}$	Planning, authorization, and communication latency.
$A_t = \tau_t^{\text{exec}} - \tau_t^{\text{sensor}}$	Observation age when an action is executed.

Frames, geometry, camera, and rigid-body dynamics

Symbol	Meaning and scope
${}^A T_B, {}^A R_B, {}^A t_B$	Transform from frame B to A , its rotation, and translation.
$SO(3), SE(3)$	Three-dimensional rotation group and rigid-transform group.
${}^A \mathbf{p}$	Coordinates of point $\mathbf{p}$ expressed in frame A .
$d_R(R_1, R_2)$	Geodesic orientation error.
K in camera equations	Camera intrinsic matrix.
f_x, f_y, c_x, c_y, Z	Focal lengths, principal point, and depth.
$D(u, v), \tilde{\mathbf{p}}$	Depth at pixel (u, v) and homogeneous image point.
$\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}}$	Joint position, velocity, and acceleration.
$F(\mathbf{q}), J(\mathbf{q}), V_e, V_d$	Forward kinematics, Jacobian, actual end-effector twist, and desired twist.
$M, C, \mathbf{g}, \boldsymbol{\tau}_f, \boldsymbol{\tau}$	Inertia, Coriolis/centrifugal term, gravity torque, friction torque, and commanded torque.
$\mathbf{f}_{\text{ext}}$	External wrench.
$\phi(\mathbf{q}), \lambda_n$	Contact gap function and normal contact force.

State-space models and belief estimation

Symbol	Meaning and scope
$f, h, \mathbf{w}_t, \mathbf{v}_t$	State transition, observation model, process disturbance, and measurement noise.
$\mathcal{H}_t, \bar{b}_t(\mathbf{x}), b_t(\mathbf{x}), \eta_t$	Observation-action history, predicted belief, corrected belief, and posterior normalizer.
ψ	Mapping from observation-action history to an information state.
A, B, H in linear models	State-transition, input, and observation matrices.
Q, R in Kalman filtering	Process-noise and measurement-noise covariance matrices.
$\hat{\mathbf{x}}_{t t-1}, \hat{\mathbf{x}}_{t t}$	Predicted and measurement-updated state estimates.
$P_{t t-1}, P_{t t}$	Predicted and updated state covariance.
$\mathbf{r}_t, S_t, K_t$ in Kalman filtering	Innovation, innovation covariance, and Kalman gain.
$\delta t, \delta \mathbf{x}$	Timing error and its first-order state effect.

Demonstrations, action chunks, and VLA training

Symbol	Meaning and scope
$\mathcal{D}^{(e)}$	Demonstration episode e .
$r_t, d_t, \mathbf{m}_t$ in a dataset	Optional reward, termination flag, and validity mask.
$N, \pi_\theta, \mu_\theta$	Supervised decision count, policy density, and policy mean.
$K, \alpha_k, \mu_k, \Sigma_k$	Mixture count, component weight, mean, and covariance.
$d_a, y_j, y_{<j}$	Action dimension, action token, and preceding action tokens.
$\hat{A}_t, \hat{\mathbf{a}}_\tau^{(t)}$	Predicted action chunk and the action for absolute time τ predicted at t .
$w_j = \exp(-\kappa j)$	Temporal-ensemble weight for chunk position j .
$m_{t,j}$	Validity mask for chunk element j at time t .
$\mu_j, \sigma_j, q_j^\ell, q_j^u$	Training mean, standard deviation, lower quantile, and upper quantile for component j .
D_e, η_e, κ_e	Embodiment-specific action decoder, interface metadata, and low-level controller.
B, l_j, u_j in tokenization	Number of bins and physical lower/upper action bounds.
$A^0, A^k, \tilde{\alpha}_k, \epsilon_\theta$	Clean action, noisy diffusion action, noise schedule, and noise predictor.
A_s, v_θ, p_0	Flow path, learned vector field, and initial noise distribution.
$J(\theta), r, \gamma, c_k, d_k$	Return objective, reward, discount, constraint cost, and allowed cost budget.

VLM and multimodal perception

Symbol	Meaning and scope
$E, \mathbf{x}_i, \mathbf{p}_i, d, P, C$	Patch projection, flattened patch, positional embedding, embedding width, patch size, and channel count.
$W_e, w_j, \mathbf{r}_j$	Token embedding matrix, token identity, and token positional embedding.
Q, K, V, W_Q, W_K, W_V	Query, key, value matrices and their learned projections. Here K means keys, not camera intrinsics or a count.
$M \in \mathbb{R}^{n \times n}$ in attention	Optional logit mask: $M_{ij} = 0$ for a permitted query-key pair and $M_{ij} = -\infty$ for a blocked pair.
H_i, h, W_O	Attention head i , number of heads, and output projection.
X_L, X_V	Language-token and visual-token matrices.
$\mathbf{v}_i, \mathbf{t}_i, \tau$	Normalized image/text embeddings and contrastive temperature.
$\mathcal{L}_{I \rightarrow T}, \mathcal{L}_{T \rightarrow I}, \mathcal{L}_{\text{CLIP}}$	Image-to-text, text-to-image, and symmetric contrastive losses.
$y_t, y_{<t}$	Output token and its autoregressive prefix.
$\text{IoU}(A, B)$	Intersection over union of regions A and B .
$\Sigma_{uvd}, J_g, \Sigma_p$	Pixel-depth covariance, geometric Jacobian, and propagated point covariance.
p_i, y_i, BS	Predicted probability, binary label, and Brier score.

World models and predictive dynamics

Symbol	Meaning and scope
$p_{\theta}(\mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t)$	Action-conditioned state dynamics.
$W, \mathcal{L}_1, \mathcal{L}_H$	State metric/weighting matrix, one-step loss, and rollout loss.
λ_k in rollout loss	Weight assigned to horizon k .
$\boldsymbol{\mu}_{\theta}, \boldsymbol{\sigma}_{\theta}$	Predicted state mean and standard deviation.
$M, \Sigma_m, \Sigma_{\text{pred}}$	Ensemble size, member covariance, and total predictive covariance.
$q_{\phi}, p_{\theta}, \mathcal{L}_{\text{ELBO}}$	Inference posterior, generative model, and evidence lower bound.
$\mathbf{h}_t, \mathbf{z}_t$	Deterministic and stochastic latent state in a recurrent state-space model.
$\hat{r}_t, c_t, V_{\omega}, G_t$	Predicted reward, continuation probability, value model, and imagined return.
$A, A_t^{\star}, J(A), U(A)$	Candidate action sequence, selected sequence, planning objective, and uncertainty penalty.
$I_{t+1:t+H}$	Predicted image sequence.
S_a	Action-sensitivity score comparing predicted effects of different actions.
$f_{\text{phys}}, r_{\theta}, \eta$	Analytical/simulator step, learned residual, and physical parameters.
$\mathbf{m}_{ij}, \phi_e, \phi_v$	Object-pair message, edge model, and object update model.
$\text{RMSE}(k), d_x$	Root-mean-square rollout error at horizon k and state dimension.
$\mathcal{M}, \widehat{\mathcal{M}}, P_t, \widehat{P}_t$	True and learned finite-horizon MDPs and their transition kernels.
$r_t, \widehat{r}_t, \epsilon_r, \epsilon_p, R_{\max}$	True/learned reward, uniform reward/transition errors, and reward bound in the simulation lemma.
$\text{TV}(P, Q)$	Total-variation distance, defined locally with the one-half convention.

JEPA and predictive representations

Symbol	Meaning and scope
C_m, T_m, m	Context masking operator, target masking operator, and target-location descriptor.
$f_\theta, f_\xi, g_\theta$	Context encoder, target encoder, and predictor.
$\text{sg}(\cdot)$	Stop-gradient operator.
$\mathbf{z}_c, \mathbf{z}_t, \hat{\mathbf{z}}_t$	Context representation, target representation, and predicted target representation.
B, M_i, m_{ij}	Batch size, targets for sample i , and target block j .
$\xi \leftarrow \mu\xi + (1 - \mu)\theta$	Exponential-moving-average target update; μ is momentum.
$\mathcal{K}, w_k, g_{\theta,k}$	Future-offset set, horizon weight, and offset-specific predictor.
$\mathcal{L}_{\text{var}}, \mathcal{L}_{\text{cov}}$	Anti-collapse variance and covariance losses.
$\gamma, \mathbf{Z}_{:,j}$	Target standard deviation and component j across a batch.
$r_{\text{eff}}, p_j, \lambda_j$	Effective representation rank, normalized eigenvalue, and covariance eigenvalue.
$\Delta_z(A, A')$	Representation change induced by changing the action sequence.
$d_g, \mathbf{z}_g, U(A)$	Goal distance, goal representation, and uncertainty penalty.
c_{ij}	Cosine similarity between gradients of objectives i and j .
$d_{\text{cos}}, E_{\text{effect}}$	Cosine distance and action-effect prediction error.

World-action models and planning

Symbol	Meaning and scope
$C = (z_{\leq t}, q)$	Current latent and language context.
A, Z^+	Candidate action sequence and predicted future representation sequence.
$p_\phi(Z^+ C, A)$	Forward dynamics model.
$\pi_\psi(A C)$	Action prior or policy.
$\rho_\psi(A C, Z^+)$	Inverse action model conditioned on a desired future.
$\mathcal{L}_F, \mathcal{L}_I, \mathcal{L}_{\text{cycle}}$	Forward, inverse, and cycle-consistency loss.
$\mathcal{L}_{\text{dir/roll}}$	Direct-versus-rollout future consistency loss.
$i^\star, J_{\text{task}}, U, C_k$	Selected candidate, predicted task cost, uncertainty, and soft constraint penalty.
$\lambda_g, \lambda_V, d, z_g$	Goal/value weights, terminal distance, and goal embedding.
$V_\omega, q, \gamma, \hat{r}_t$	Terminal-value model, task context, discount factor, and predicted reward.
$S, S_j, S_{<j}$	Interleaved action-state token sequence, token j , and its prefix.
$\mathbf{h}_k, A_k, Z_k^+$	High-level subgoal, lower-level action chunk, and predicted subgoal result.
$\xi^\star, G_\theta(\xi, C)$	Optimized plan latent and action generator.
$s, \sigma_k, \nabla_{A^k} J$	Guidance strength, diffusion noise scale, and objective gradient.
$\mathbb{P}(C_k C, A) \leq \delta_k$	Chance constraint with allowed failure probability δ_k .
E_{AF}, Regret	Action-future consistency error and real-world planning regret.

Simulation, twins, and transfer

Symbol	Meaning and scope
$(\hat{\mathbf{x}}_t^{(j)}, \hat{\eta}_t^{(j)})$	Estimated state and parameters of physical asset j .
$\mathcal{U}$	Synchronization/update procedure that maintains a twin.
p_s, p_r, J_s, J_r	Simulation and real distributions or objectives.
$\Delta_J(\pi) = J_r(\pi) - J_s(\pi)$	Sim-to-real performance gap for policy π .
$\eta, \hat{\eta}, R(\eta)$	Simulator parameters, identified parameters, and parameter regularizer/prior.
$\mathcal{E}$	Admissible parameter set in robust optimization.
D_ϕ	Domain discriminator.
$\lambda_r, \lambda_s, w(\eta, \tau)$	Real/synthetic loss weights and sample-quality weight.
$\mathbf{r}_t$ in twin monitoring	Observation residual against the synchronized twin.
$C_{\text{scenario}}, n_j, n_j^{\min}, w_j$	Scenario coverage, trial count, required count, and criticality weight.
$\xi_i, p(\xi)/q(\xi)$	Scenario sample and importance ratio.
$\hat{p}_F, F(\xi)$	Estimated failure probability and failure event.

Hallucination, feasibility, and detector notation

Symbol	Meaning and scope
$\mathcal{R}(a \mid o)$	Expected consequence risk of action a given observation o .
$L(a, s), p(s \mid o)$	Consequence in possible state s and belief over that state.
$F(a \mid o, e, g, t)$	Binary feasibility predicate under observation, embodiment, goal, and time.
r_q, r_v	Joint-limit residual and rate-limit residual.
$q_j^{\min}, q_j^{\max}, v_j^{\max}$	Joint lower bound, upper bound, and maximum rate.
r_t^D	Standardized dynamics residual.
r^{CF}	Counterfactual action-effect residual.
$\mathcal{H}, c_i, \hat{h}_i$	True hallucination set, consequence cost, and detector decision.
$j, N_r, r_j, b_j, \alpha_j, \eta$	Residual index, number of simultaneous tests, residual, bound, allocated exceedance probability, and operating threshold.
$S = \max_{1 \leq j \leq N_r} r_j / b_j$	Composite detector severity; reject when $S > \eta$.
$\mathcal{A}^*, d(\hat{A}, \mathcal{A}^*)$	Admissible action set and distance of a proposal from it.
$F_t, S_{t-1}, h_t, F_{\leq H}$	Stepwise failure event, survival through step $t-1$, conditional hazard, and failure-by-horizon event.
$a_s, C(a_s), H(b_t)$	Sensing action, sensing cost, and belief entropy.
$P_{\text{prevent}}, P_{\text{false}}, R_{\text{progress}}$	Prevention, false-blocking, and retained-progress metrics.

Authorization, reachability, and barriers

Symbol	Meaning and scope
$M(\hat{\xi}_{t:t+H}, \Theta_t)$	Runtime monitor applied to predicted quantities under monitor configuration Θ_t .
(d_t, c_t, I_t, S_t)	PASS/REJECT decision, active condition, offending indices, and severity score.
$M, a^{(m)}, \bar{a}, \Sigma_a$	Ensemble size, member action, ensemble-mean action, and sample action covariance.
$W, S_U, \lambda_{\max}$	Declared uncertainty weighting map, task-sensitive uncertainty score, and largest eigenvalue.
$q_\alpha, \Gamma_\alpha(x), X_{\text{new}}, Y_{\text{new}}$	Split-conformal threshold, prediction set, and new exchangeable input/label pair.
$D_M(z)$	Mahalanobis distance from reference feature statistics.
$\mathcal{R}_T(\mathbf{x}_0), \mathcal{U}, \mathcal{U}_T$	Reachable set after time T , pointwise input set, and admissible input-signal class.
A, B in reachability	Linear state-transition and control matrices.
$h(\mathbf{x}), C = \{\mathbf{x} : h(\mathbf{x}) \geq 0\}$	Barrier function and declared safe set.
$L_f h, L_g h, \alpha, \gamma$	Lie derivatives, extended class- $\mathcal{K}$ function, and discrete-time barrier contraction parameter.
$\mathbf{u}_{\text{nom}}, \mathbf{u}^\star$	Nominal and barrier-filtered control.
$\eta_{\text{low}}, \eta_{\text{high}}$	Hysteresis thresholds for return and fallback.
$\hat{\phi}_h, \Delta_{h,k}^{F,u}$	h -step predictor and forward-composition inconsistency.
$\Delta_{h,k}^R, \epsilon_{h,k}^R$	Reachability residual and its allowed bound.
$D_{\Delta t}^p, E_p^\Delta(H), B_p(H)$	p -order finite-difference operator, smoothness energy, and reference bound.
Δ_i^D, ϵ^D	Standardized dynamics residual at step i and its limit.
$\Delta_{h,k}^F, \epsilon_{h,k}^F$	Forward-consistency residual and limit.
$\mathbb{P}(\bigcup_{j=1}^{N_r} \{r_j > b_j\}) \leq \sum_{j=1}^{N_r} \mathbb{P}(r_j > b_j)$	Union bound for N_r residual events; no independence assumption is required.
$R(\eta), D, \pi_d, C_{\text{miss}}, C_{\text{int}}$	Threshold risk, validity label, deployment prior, invalid-pass cost, and valid-rejection cost.
$N_{\text{invalid}}, N_{\text{valid}}, N_{\text{fallback}}$	Labeled proposal counts, fallback count, and declared task-progress measure.

Systems, release, and operations

Symbol	Meaning and scope
$\mathcal{T}_t$	Traceability tuple for observation, revisions, configuration, clocks, output, score, and decision.
$i_o, i_s, i_q, i_m, i_c, i_e$	Observation, state-estimator, instruction, model, configuration, and monitor-envelope identifiers.
y_t, S_t, d_t	Raw model output, monitor severity, and monitor decision.
$q, w, s, z, \hat{w}$ in quantization	Integer code, real weight, positive scale, zero point, and dequantized weight.
$p_t^T, p_s^T, T, \lambda_{\text{KD}}$	Teacher/student softened distributions, temperature, and distillation weight.
$C(m, r, e, v) = 1$	Compatibility predicate for model, robot, envelope, and interface version.
p_b, q_b, PSI	Reference bin fraction, current bin fraction, and Population Stability Index.
$G_{\text{schema}}, G_{\text{offline}}, G_{\text{closed-loop}}, G_{\text{latency}}$	Schema, offline, closed-loop, and latency release gates.
$G_{\text{physical}}, G_{\text{fallback}}, G_{\text{security}}$	Physical, fallback, and security release gates.
C_F, w_i	Weighted fault-injection coverage and fault criticality.
$D_n, D_r, \lambda_{\text{replay}}, \lambda_{\text{anchor}}$	New/replay data, replay weight, and anchoring weight.
θ_0	Previous released parameter vector.
$\mathcal{R} =$	Reproducible training record: code, data, splits, model, objective,
(C, D, S, M, O, H, E, A)	hyperparameters, environment, and assessment.

Derivation-atlas notation

Symbol	Meaning and scope
$J_H(\pi), p_\pi(\tau)$	Finite-horizon policy cost and the induced trajectory distribution.
c_t, c_H, d_π^t	Stage cost, terminal cost, and learner state occupancy at time t .
$A_t^{\pi_E}(x, a)$	Finite-horizon advantage relative to expert policy π_E .
$A_{\max}, \epsilon_t$	Advantage bound and per-step learner/expert discrepancy.
$\mathcal{L}_{\text{ELBO}}$	Evidence lower bound for a latent sequence model.
$A_0, A_1, A_s, U_s, v_\theta$	Flow source, target, interpolation, target velocity, and learned velocity.
$\epsilon_{\text{representation}}, \epsilon_{\text{field}}, \epsilon_{\text{solver}}, \epsilon_{\text{decoder}}$	Representation, vector-field, numerical-solver, and decoder error terms.
$r_i, r(k), n, \alpha, q_\alpha$	Conformal nonconformity score, ordered score, calibration count, miscoverage, and calibrated quantile.
$C(X), \mathcal{E}_j$	Conformal prediction set and guarantee event j .
$h(x), \dot{h}, \kappa$	Barrier value, time derivative, and exponential barrier rate.

ENGINEERING NOTE - Overloaded symbols

The guide resolves overload by context. K can be camera intrinsics, mixture count, key matrix, Kalman gain K_t , or number of checked events. M can be inertia, an attention mask, ensemble size, or target count. A can be an event, system matrix, action sequence, LoRA factor, or admissible set element. S can be a token sequence, innovation covariance, monitor score, or set of calibration samples. η can be simulator parameters, a threshold, embodiment metadata, or the auxiliary CVaR variable. Always read the nearest definition and units.



B.2 Glossary and Acronym Map

ACT - Action Chunking with Transformers

A conditional latent-variable policy that predicts temporally coherent action chunks.

Action proposal

A model output expressed in an agreed action schema; it is not an actuator command until authorized and transformed by downstream control.

Belief state

A distribution over latent physical states conditioned on the available observation and action history.

CBF - Control barrier function

A scalar function used to define a safe set and impose forward-invariance constraints on a controller.

Digital twin

A virtual asset whose state and parameters are synchronized with a particular physical asset for a defined operational purpose.

Embodiment

The sensors, kinematics, dynamics, action interface, timing, morphology, and operating environment that give a policy physical meaning.

Hallucination, Physical AI

A semantically or statistically plausible internal claim, predicted future, plan, or action that is materially inconsistent with observed state, embodiment, physics, task contract, or safety constraints.

Information state

A finite representation of history used as the input to prediction or decision making.

JEPA - Joint-embedding predictive architecture

A model that predicts target representations from context representations rather than reconstructing every observed sample.

MPC - Model-predictive control

Receding-horizon optimization that repeatedly selects a sequence under a model, executes a prefix, and replans.

OOD - Out of distribution

Inputs or operating conditions not adequately represented by the validated data distribution.

Physical admissibility

Compatibility of a predicted state transition with specified physical constraints and uncertainty tolerances.

RFM - Robot foundation model

A broadly pretrained robot model designed for adaptation across tasks, scenes, or embodiments.

Runtime assurance

An architecture that monitors an advanced controller or learned policy and switches, rejects, or modifies behavior when an assurance condition is violated.

Simulator

A computational process that evolves a parameterized environment model under actions; fi-

delity is always relative to a use case.

VLA - Vision-language-action model

A conditional policy or action generator that maps visual observations, language, and possibly robot state/history to actions or action distributions.

VLM - Vision-language model

A model that aligns or generates over visual and linguistic representations; by itself it does not define an executable robot policy.

WAM - World-action model

A model family that couples action generation with explicit prediction of intervention-conditioned futures, often through cascaded or joint modeling.

World foundation model

A large, broadly pretrained predictive model intended to transfer across environments or physical domains.

World model

A learned or specified model of how task-relevant state or observation variables evolve, usually conditioned on actions.



APPENDIX C

Engineering Reference

C.1 Deployment Casebook

These case files study complete systems rather than isolated checkpoints. Published evidence is separated from the engineering reconstruction added here. The corresponding numbered studies are integrated at their point of use in Lessons 3, 11, 12, and 13, where each calculation follows the concept it tests.

Case 1: Open-vocabulary mobile manipulation in homes

Published evidence. OK-Robot combines open-vocabulary visual representations, navigation, and grasping primitives without task-specific training. The authors evaluated it in ten real-world homes and reported 58.5% success on open-ended pick-and-drop tasks; under cleaner, less adversarial conditions the reported rate rose to about 82% [104]. The important result is not only the score. Integration details and physical limitations accounted for a substantial part of failure.

The system is modular: a scanned environment supports semantic memory; a language query retrieves a target; navigation moves the platform; grasping executes the local interaction. This makes failure attribution possible. A monolithic success metric should therefore be accompanied by stage outcomes: target retrieval, reachability, navigation arrival, grasp generation, pickup, transport, and placement.

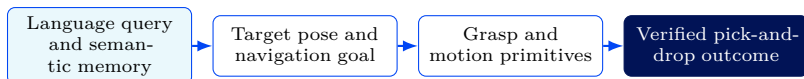


Figure C.1: An open-vocabulary mobile-manipulation case exposes the interfaces between foundation models and classical robotics.

Release lesson. Preserve modular traces even when replacing the stack with an end-to-end VLA. Record the retrieved entity, metric target, navigation termination reason, grasp proposal, controller state, and final task outcome. Otherwise a better aggregate score can hide a newly concentrated failure mode.

Case 2: Whole-body bimanual learning and open-world generalization

Published evidence. Mobile ALOHA demonstrated low-cost whole-body teleoperation and imitation learning for mobile bimanual tasks. The study used 50 demonstrations per task and reported that co-training with static ALOHA data could increase success rates by up to 90% [105]. The later $\pi_{0.5}$ system used heterogeneous robot, semantic, and web data and demonstrated long-horizon manipulation in novel homes [106]. These are evidence for data diversity and hierarchical behavior, not proof of universal household autonomy.

Whole-body tasks couple base motion, two arms, grippers, scene memory, and long-horizon subgoals. A useful architecture separates semantic subtask state from fast continuous action while preserving a single incident timeline. Demonstration count alone is insufficient: report duration, intervention, reset, failure coverage, operator, home, object, and action bandwidth.

Release lesson. Split by home and object instance, not random frames. Long-horizon claims require completion, intervention count, recovery, constraint violations, and time-to-completion. A policy that completes only after repeated human resets has not demonstrated autonomous task execution.

Case 3: Dual-system humanoid and generalist robot policies

Published evidence. GR00T N1 uses a vision-language system for interpretation and a diffusion-transformer action system, trained from a mixture of real-robot trajectories, human video, and synthetic data; the report includes deployment on a Fourier GR-1 humanoid for language-conditioned bimanual manipulation [107]. Gemini Robotics reports a generalist VLA plus an embodied-reasoning model, transfer across tasks and embodiments, and specialization experiments using as few as 100 demonstrations for some new short-horizon skills [108].

Dual-system language is useful only when rate and authority are concrete. The slower semantic path may update task interpretation and spatial context; the faster ac-

tion path must meet the physical control deadline. An independent monitor remains necessary because both paths can share the same unsupported premise.

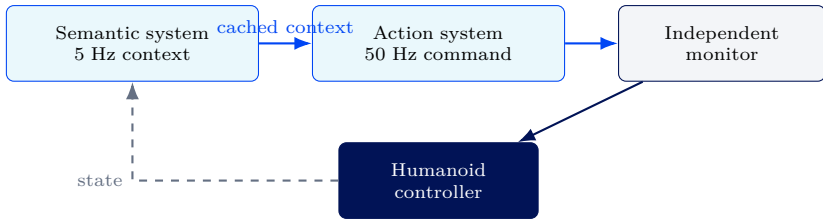


Figure C.2: A dual-rate interpretation/action architecture must state cache age, deadlines, and monitor authority.

Release lesson. Report system-level timing and authority, not only policy frequency. The fast model, semantic model, controller, and monitor should have separate version identifiers so a regression can be localized.

Case 4: Predictive intelligence for road scenes

Generative driving world models such as GAIA-1 model video, action, and language context, while CARLA provides repeatable physics, traffic, and sensor simulation [55, 76]. The useful system question is not whether a generated future looks realistic. It is whether the future preserves the variables that govern braking, collision, right-of-way, visibility, and uncertainty.



Figure C.3: Illustrative stopping-distance audit: a plausible but biased gap prediction can reverse an authorization decision.

Release lesson. Evaluate the predicted state needed for decisions: occupancy, relative motion, road geometry, signal state, and uncertainty. Video quality is supporting evidence, never the authorization metric.

Case 5: Brownfield industrial-cell retrofit

This final case is an engineering reference design, not a claim about a reported installation. Consider adding a learned vision and action-proposal layer to an existing industrial cell. ISO 10218-1:2025 addresses industrial robots, while ISO 10218-2:2025 addresses applications and cells [98,99]. The learned model enters an already governed system; it does not replace the risk assessment, safeguarding, protective devices, or validated control functions.



Figure C.4: *Illustrative invalid-proposal funnel. A small residual count still requires severity analysis and safe fallback.*

Release lesson. Deploy first in shadow mode, then constrained canary operation. Sign the model, decoder, monitor envelope, robot configuration, and controller as one release set. A model update that preserves offline accuracy but changes intervention rate is a system change.

Cross-case comparison

Case	Learned core	Physical bottleneck	Evidence required beyond task success
Open-vocabulary home	VLM plus modular skills	retrieval-to-pose and graspability	stage failures, home/object split, recovery trace
Bimanual mobile	imitation/VLA hierarchy	whole-body coordination and drift	intervention, duration, reset, unseen-home performance
Humanoid generalist	dual-system VLA	fast-path deadline and authority	p95/p99 latency, cache age, monitor and controller logs
Road prediction	world model/WAM	braking, occupancy, rare events	calibrated state error, counterfactual consistency, HIL timing
Industrial retrofit	learned proposal layer	safeguarding and valid envelope	shadow exposure, false intervention, fallback, signed release

Table C.1: *The same model score has different meaning across embodiments and operating envelopes.*

A useful case study reports the published result, reconstructs the system boundary, performs at least one checkable calculation, identifies missing evidence, and ends with a release implication. Architecture screenshots without these elements are examples, not engineering cases.

C.2 Physical AI Engineering Desk Reference

This section is designed for use during design reviews, training runs, and release meetings. It compresses the book into decisions and records without replacing the derivations, limitations, or case evidence behind them.

Choose the model by the required query

Model class	Query it must answer	Minimum acceptance evidence
VLM	What is present, referred to, or semantically related?	existence/identity, grounding, metric handoff, state, abstraction, calibration
VLA	What action chunk should this embodiment propose now?	action semantics, closed-loop success, shift, latency, constraints, recovery
World model	What follows from this state/history and candidate action?	multi-step error, action effect, uncertainty, consistency, task-state preservation
JEPA	Which target representation follows from context?	anti-collapse diagnostics, probes, intervention response, downstream control
WAM	Which futures and actions are jointly compatible with the goal?	forward/inverse queries, candidate coverage, ranking, latency, physical checks
Simulator	What occurs under encoded dynamics, geometry, sensors, and agents?	validity envelope, real residuals, step/timing, known omitted mechanisms
Digital twin	What is the state and forecast of this identified asset?	identity, synchronization, parameter/state update, drift and audit trail

Table C.2: *Model names are secondary to the conditional query and evidence contract.*

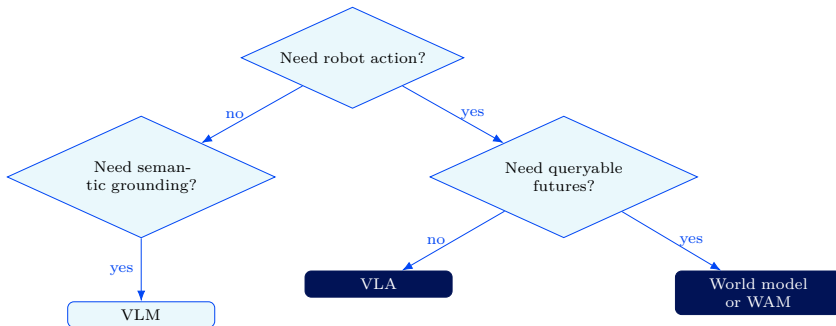


Figure C.5: *A first-pass selection tree. Architecture choice still depends on data, embodiment, timing, and assurance.*

The interface card

Complete one row for every observation, state estimate, model input, model output, decoded action, and monitor signal.

Field	Required entry
Meaning	physical or semantic variable; distinction between measured, estimated, commanded, and executed
Representation	tensor shape, dtype, encoding, ordering, mask, missing-value behavior
Units and frame	SI or declared unit, coordinate frame, orientation convention, reference point
Time	source clock, timestamp meaning, sampling rate, resampling, maximum age, deadline
Distribution	collection policy, task/site/robot coverage, normalization, known exclusions
Decoder	transformation from model value to physical request, clipping and saturation
Validity	finite checks, range, consistency, uncertainty, OOD and calibration evidence
Failure response	reject, shorten, reobserve, replan, fallback, controlled stop, human request
Traceability	schema, calibration, dataset, model, configuration, and envelope revisions

Table C.3: *A tensor is not an interface until its physical meaning and failure behavior are defined.*

Training-run record

The minimum reproducible record is

$$\mathcal{R} = (C, D, S, M, O, H, E, A), \quad (\text{C.1})$$

where C is code revision and environment; D immutable dataset revision; S split manifest; M model and initialization; O objective and optimizer; H hyperparameters and seed; E evaluation protocol; and A produced artifacts. Store hashes, not only filenames.

- ▶ **Before training:** validate schema, frames, timestamps, leakage, normalization, and a tiny-set overfit.
- ▶ **During training:** log component losses, gradient/activation health, throughput, memory, examples, and checkpoints.
- ▶ **After training:** rerun frozen evaluation, record hardware and latency distributions, export decoder and normalization, and sign the candidate bundle.
- ▶ **Before hardware:** replay, simulation, monitor fault injection, fallback test, shadow mode, and explicit authorization.

Formula sheet with decision meaning

Quantity	Expression	Decision it supports
Action likelihood	$-\log \pi_{\theta}(a \mid o, q)$	supervised fit; not closed-loop competence
Rollout error	$\sum_{k=1}^H \lambda_k \ \hat{x}_{t+k} - x_{t+k}\ ^2$	horizon choice and model-bias diagnosis
Expected calibration error	$\sum_b \frac{ B_b }{N} \text{acc}(B_b) - \text{conf}(B_b) $	confidence threshold review
Chance constraint	$\Pr(g_j(\zeta) \leq 0, \forall j) \geq 1 - \alpha$	risk allocation under a stated distribution
Barrier condition	$\dot{h}(x, u) \geq -\kappa h(x)$	forward-invariance filter under model assumptions
Latency margin	$T_{\text{deadline}} - Q_p(T_{\text{path}})$	schedulability at chosen percentile p
Retained progress	$G_{\text{guarded}}/G_{\text{nominal}}$	whether monitoring preserves useful behavior
Residual exposure	$N_{\text{invalid, accepted}}/T_{\text{operation}}$	severity-weighted operational risk tracking

Table C.4: Every reported quantity should be tied to a decision and a boundary.

Evaluation ladder

Do not skip a rung because a higher-level demo looks convincing.

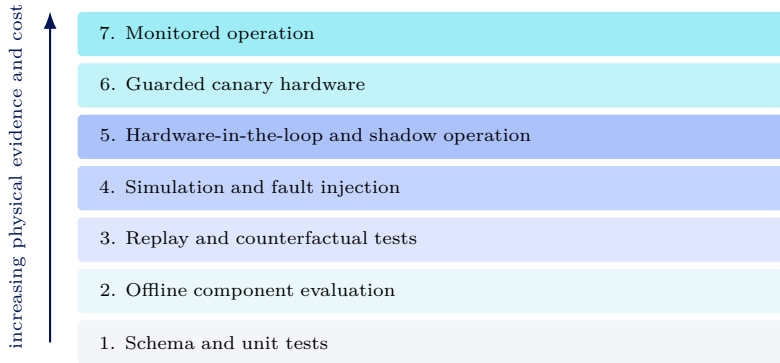


Figure C.6: Evidence becomes more physically representative and more costly up the ladder.

At every rung record numerator and denominator, scenario distribution, intervention policy, confidence interval, and failure severity. Promotion requires both capability and integrity: task progress, timing, physical bounds, calibration, fallback, and traceability.

Release gate

Gate	Owner	Required evidence
Interface	systems	schema, units, frames, clock, decoder, finite/range tests
Data	data lead	immutable revision, license, provenance, split/leakage and coverage report
Model	ML lead	objective, seed, environment, checkpoint, ablations, calibration and limitations
Closed loop	robotics	repeated trials, shift factors, recovery, latency, controller and HIL evidence
Authorization	safety/control	constraints, thresholds, false intervention, residual exposure, fall-back
Operations	release owner	signed bundle, canary scope, observability, rollback and incident procedure

Table C.5: *No single metric or team owns a Physical AI release.*

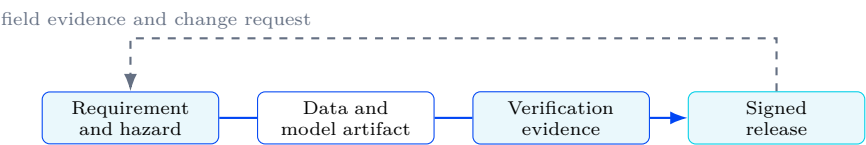


Figure C.7: *Requirements, artifacts, verification evidence, and releases remain linked through a controlled feedback cycle.*

Change classification and retraining

Classify every update before deciding how much evidence to repeat.

- ▶ **Class A - editorial/configuration only:** no tensor, decoder, monitor, or timing effect. Rebuild, review, and checksum.
- ▶ **Class B - data or calibration:** repeat leakage, distribution, calibration, offline, replay, and affected closed-loop tests.
- ▶ **Class C - model or action interface:** repeat the complete evaluation ladder, monitor calibration, timing, and canary approval.
- ▶ **Class D - embodiment, controller, or safety envelope:** treat as a system re-design and renew the applicable risk assessment and assurance argument.

Trigger retraining from diagnosed coverage gaps, not from elapsed time alone. A field incident should produce a trace, taxonomy label, root-cause hypothesis, corrected dataset slice, regression test, and release decision. Preserve the old model and evaluation environment for rollback and comparison.

Source hierarchy for a living field

When updating this book or a model card, prefer: standards and regulator/manufacture requirements for applicable obligations; peer-reviewed papers and official technical reports for methods and results; official code, model cards, dataset cards, and release notes for executable facts; and dated company material only for claims made by that organization. Secondary surveys are useful maps, but a numerical or architectural claim should resolve to its primary source.

ENGINEERING NOTE - The one-page review

Before approval, ask: What is the conditional interface? Which physical variables and times does it represent? What data support the operating envelope? Which calculation governs the deadline or constraint? Which failures remain? What independent mechanism blocks or recovers? Which exact versions produced the evidence? If one answer is missing, the review is not complete.



APPENDIX D

Literature Landscape

No single source tradition owns Physical AI. The field is assembled from mechanics, control, estimation, vision, probabilistic inference, planning, reinforcement learning, deep representation learning, robot learning, dependable systems, and a rapidly changing foundation-model literature. This survey distinguishes the books that supply durable foundations from the papers and software that define the 2026 frontier.

Robotics, mechanics, planning, and control

Source tradition	What it contributes
Lynch and Park [1]	Screw theory, rigid motion, kinematics, dynamics, motion planning, and control in a compact computational sequence.
Siciliano and Khatib [109]	Broad reference coverage across manipulation, mobility, sensing, human-robot interaction, and applications.
Murray, Li, and Sastry [22]	Geometric and mathematical foundations for robotic manipulation.
Craig [23]; Spong et al. [110]	Classical modeling and control pathways used in engineering curricula.
LaValle [111]; Choset et al. [112]	Configuration spaces, search, sampling-based planning, and motion algorithms.
Kelly [25]	Integrated mathematics and models for mobile robotics.
Rawlings, Mayne, and Diehl [113]	Optimization-based receding-horizon control and constraint handling.

These works explain what a robot *is allowed and able* to do. They generally do not organize the complete VLM-VLA-world-model-WAM-JEPA stack or the new failure

semantics created by generative action models.

Estimation, vision, and uncertainty

Source tradition	What it contributes
Thrun, Burgard, and Fox [2]	Bayes filters, localization, mapping, and decision-making under uncertain state.
Barfoot [20]	Modern estimation on manifolds with careful derivations and implementation structure.
Hartley and Zisserman [24]	Projective and multi-view geometry for calibrated visual reasoning.
Szeliski [114]	Broad computer-vision algorithms, representations, and evaluation.
Corke et al. [21]	Executable bridges between robot geometry, vision, and control.
Murphy [6]	Probabilistic machine-learning language for uncertainty, latent variables, and prediction.

This tradition provides belief and geometry. Modern VLM confidence, open-vocabulary grounding, latent world prediction, and physical hallucination require additional evaluation and system interfaces.

Learning, decision making, and dependable systems

Source tradition	What it contributes
Goodfellow, Bengio, and Courville [4] Barak Or [3]	Representation learning, optimization, regularization, and sequence-model foundations. An engineering-first route from neural-network fundamentals through CNNs, sequence models, transformers, training practice, transfer learning, framework workflows, and deployment. It is the practical deep-learning companion to this book.
Sutton and Barto [5] Kochenderfer et al. [102]	Value functions, policy learning, temporal difference methods, and exploration. A unified computational view of decision making under uncertainty.
Alpaydin [77]	Model selection, generalization, and applied machine-learning foundations.
Leveson [96]; Knight [100]	System safety, dependability, hazards, faults, and life-cycle evidence.

Why frontier papers and software are indispensable

Books stabilize knowledge slowly; the architecture frontier moves quickly. RT-2, Open X-Embodiment, OpenVLA, π_0 , SmolVLA, LeRobot, Cosmos, V-JEPA, and the explicit WAM literature are therefore treated through primary sources [8, 9, 11–16, 32].

Coverage gap this book addresses

Source family		Embodied math	Foundation mod-els	Runtime physi-cal assurance
Classical	robotics	core	limited	control/safety specific
books				
Deep-learning	books	supporting	general	limited
Robot-learning		task specific	core	emerging
papers				
Safety/dependability		system specific	limited	core
books				
This book		integrated	VLM/VLA JEPa/WAM	hallucination, au- thorization, fall- back, and release



Bibliography

- [1] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. Cambridge University Press, 2017.
- [2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.
- [3] B. Or, *Modern Deep Learning Foundations: From Fundamentals to Deployment - A Practical Guide for Engineers*, 1st ed., 2025, course notes.
- [4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: <https://www.deeplearningbook.org/>
- [5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018. [Online]. Available: <http://incompleteideas.net/book/the-book-2nd.html>
- [6] K. P. Murphy, *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.
- [7] A. Brohan, N. Brown, J. Carbajal *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *arXiv preprint arXiv:2212.06817*, 2022. [Online]. Available: <https://arxiv.org/abs/2212.06817>
- [8] A. Brohan *et al.*, “Rt-2: Vision-language-action models transfer web knowledge to robotic control,” *arXiv preprint arXiv:2307.15818*, 2023. [Online]. Available: <https://arxiv.org/abs/2307.15818>
- [9] Open X-Embodiment Collaboration *et al.*, “Open x-embodiment: Robotic learning datasets and rt-x models,” *arXiv preprint arXiv:2310.08864*, 2023. [Online]. Available: <https://arxiv.org/abs/2310.08864>
- [10] Octo Model Team *et al.*, “Octo: An open-source generalist robot policy,” *arXiv preprint arXiv:2405.12213*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.12213>

- [11] M. J. Kim *et al.*, “Openvla: An open-source vision-language-action model,” *arXiv preprint arXiv:2406.09246*, 2024. [Online]. Available: <https://arxiv.org/abs/2406.09246>
- [12] K. Black, N. Brown, D. Driess *et al.*, “ π_0 : A vision-language-action flow model for general robot control,” *arXiv preprint arXiv:2410.24164*, 2024. [Online]. Available: <https://arxiv.org/abs/2410.24164>
- [13] M. Shukor *et al.*, “Smolvla: A vision-language-action model for affordable and efficient robotics,” *arXiv preprint arXiv:2506.01844*, 2025. [Online]. Available: <https://arxiv.org/abs/2506.01844>
- [14] A. Bardes *et al.*, “V-jepa 2: Self-supervised video models enable understanding, prediction and planning,” *arXiv preprint arXiv:2506.09985*, 2025. [Online]. Available: <https://arxiv.org/abs/2506.09985>
- [15] NVIDIA *et al.*, “Cosmos world foundation model platform for physical ai,” *arXiv preprint arXiv:2501.03575*, 2025. [Online]. Available: <https://arxiv.org/abs/2501.03575>
- [16] S. Wang, J. Shi, Z. Fu *et al.*, “World action models: The next frontier in embodied ai,” *arXiv preprint arXiv:2605.12090*, 2026. [Online]. Available: <https://arxiv.org/abs/2605.12090>
- [17] Y. Liang, X. Zhan, H. Huang, C. Zhang, and X. Li, “From world action models to embodied brains: A roadmap for open-world physical intelligence,” *arXiv preprint arXiv:2607.11689*, 2026. [Online]. Available: <https://arxiv.org/abs/2607.11689>
- [18] D. Ha and J. Schmidhuber, “World models,” *arXiv preprint arXiv:1803.10122*, 2018. [Online]. Available: <https://arxiv.org/abs/1803.10122>
- [19] Y. LeCun, “A path towards autonomous machine intelligence,” *OpenReview preprint*, 2022. [Online]. Available: <https://openreview.net/forum?id=BZ5a1r-kVsF>
- [20] T. D. Barfoot, *State Estimation for Robotics*, 2nd ed. Cambridge University Press, 2024.
- [21] P. Corke, W. Jachimczyk, and R. Pillat, *Robotics, Vision and Control*, 3rd ed. Springer, 2023.
- [22] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. [Online]. Available: <http://www.cds.caltech.edu/~murray/books/MLS/pdf/mlswiki.pdf>

-
- [23] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 4th ed. Pearson, 2017.
- [24] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.
- [25] A. Kelly, *Mobile Robotics: Mathematics, Models, and Methods*. Cambridge University Press, 2013.
- [26] S. Ross, G. Gordon, and D. Bagnell, “A reduction of imitation learning and structured prediction to no-regret online learning,” in *Proceedings of AISTATS*, 2011. [Online]. Available: <https://proceedings.mlr.press/v15/ross11a.html>
- [27] C. Chi, S. Feng, Y. Du *et al.*, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.04137>
- [28] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” *arXiv preprint arXiv:2304.13705*, 2023. [Online]. Available: <https://arxiv.org/abs/2304.13705>
- [29] A. Khazatsky *et al.*, “Droid: A large-scale in-the-wild robot manipulation dataset,” *arXiv preprint arXiv:2403.12945*, 2024. [Online]. Available: <https://arxiv.org/abs/2403.12945>
- [30] H. Walke, K. Black, A. Lee *et al.*, “Bridgedata v2: A dataset for robot learning at scale,” *arXiv preprint arXiv:2308.12952*, 2023. [Online]. Available: <https://arxiv.org/abs/2308.12952>
- [31] B. Liu, Y. Zhu, C. Gao *et al.*, “Libero: Benchmarking knowledge transfer for lifelong robot learning,” *arXiv preprint arXiv:2306.03310*, 2023. [Online]. Available: <https://arxiv.org/abs/2306.03310>
- [32] R. Cadene, S. Aliberts, F. Capuano *et al.*, “Lerobot: An open-source library for end-to-end robot learning,” *arXiv preprint arXiv:2602.22818*, 2026. [Online]. Available: <https://arxiv.org/abs/2602.22818>
- [33] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, “Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection,” in *International Symposium on Experimental Robotics*, 2016. [Online]. Available: <https://arxiv.org/abs/1603.02199>

- [34] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Proceedings of ICML*, 2017. [Online]. Available: <https://proceedings.mlr.press/v70/guo17a.html>
- [35] A. Radford, J. W. Kim, C. Hallacy *et al.*, “Learning transferable visual models from natural language supervision,” in *Proceedings of ICML*, 2021. [Online]. Available: <https://arxiv.org/abs/2103.00020>
- [36] C. Jia, Y. Yang, Y. Xia *et al.*, “Scaling up visual and vision-language representation learning with noisy text supervision,” in *Proceedings of ICML*, 2021. [Online]. Available: <https://arxiv.org/abs/2102.05918>
- [37] J.-B. Alayrac, J. Donahue, P. Luc *et al.*, “Flemingo: A visual language model for few-shot learning,” in *Advances in Neural Information Processing Systems*, 2022. [Online]. Available: <https://arxiv.org/abs/2204.14198>
- [38] D. Driess, F. Xia, M. S. M. Sajjadi *et al.*, “Palm-e: An embodied multimodal language model,” in *Proceedings of ICML*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.03378>
- [39] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, “Sigmoid loss for language image pre-training,” in *Proceedings of ICCV*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.15343>
- [40] A. Kirillov, E. Mintun, N. Ravi *et al.*, “Segment anything,” in *Proceedings of ICCV*, 2023. [Online]. Available: <https://arxiv.org/abs/2304.02643>
- [41] E. J. Hu, Y. Shen, P. Wallis *et al.*, “Lora: Low-rank adaptation of large language models,” in *Proceedings of ICLR*, 2022. [Online]. Available: <https://arxiv.org/abs/2106.09685>
- [42] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: <https://arxiv.org/abs/1705.07874>
- [43] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, “Flow matching for generative modeling,” in *Proceedings of ICLR*, 2023. [Online]. Available: <https://arxiv.org/abs/2210.02747>
- [44] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” in *Advances in Neural Information Processing Systems*, 2023. [Online]. Available: <https://arxiv.org/abs/2305.14314>
- [45] Hugging Face, “Lerobot documentation, version 0.6,” 2026, accessed 31 July 2026. [Online]. Available: <https://huggingface.co/docs/lerobot/>

-
- [46] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proceedings of ICLR*, 2019. [Online]. Available: <https://arxiv.org/abs/1711.05101>
- [47] A. Paszke, S. Gross, F. Massa *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, 2019. [Online]. Available: <https://arxiv.org/abs/1912.01703>
- [48] M. Abadi, P. Barham, J. Chen *et al.*, “Tensorflow: A system for large-scale machine learning,” in *Proceedings of OSDI*, 2016. [Online]. Available: <https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi>
- [49] M. J. Kearns and S. Singh, “Near-optimal reinforcement learning in polynomial time,” *Machine Learning*, vol. 49, no. 2-3, pp. 209-232, 2002. [Online]. Available: <https://www.cis.upenn.edu/~mkearns/papers/KearnsSinghE3.pdf>
- [50] K. Chua, R. Calandra, R. McAllister, and S. Levine, “Deep reinforcement learning in a handful of trials using probabilistic dynamics models,” in *Advances in Neural Information Processing Systems*, 2018. [Online]. Available: <https://arxiv.org/abs/1805.12114>
- [51] D. Hafner, T. Lillicrap, I. Fischer *et al.*, “Learning latent dynamics for planning from pixels,” in *Proceedings of ICML*, 2019. [Online]. Available: <https://arxiv.org/abs/1811.04551>
- [52] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, “Dream to control: Learning behaviors by latent imagination,” in *Proceedings of ICLR*, 2020. [Online]. Available: <https://arxiv.org/abs/1912.01603>
- [53] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap, “Mastering diverse domains through world models,” *arXiv preprint arXiv:2301.04104*, 2023. [Online]. Available: <https://arxiv.org/abs/2301.04104>
- [54] M. Janner, J. Fu, M. Zhang, and S. Levine, “When to trust your model: Model-based policy optimization,” in *Advances in Neural Information Processing Systems*, 2019. [Online]. Available: <https://arxiv.org/abs/1906.08253>
- [55] A. Hu, L. Russell, H. Yeo *et al.*, “Gaia-1: A generative world model for autonomous driving,” *arXiv preprint arXiv:2309.17080*, 2023. [Online]. Available: <https://arxiv.org/abs/2309.17080>

- [56] M. Deisenroth and C. E. Rasmussen, “Pilco: A model-based and data-efficient approach to policy search,” in *Proceedings of ICML*, 2011. [Online]. Available: <https://proceedings.mlr.press/v15/deisenroth11a.html>
- [57] M. Assran, Q. Duval, I. Misra *et al.*, “Self-supervised learning from images with a joint-embedding predictive architecture,” in *Proceedings of CVPR*, 2023. [Online]. Available: <https://arxiv.org/abs/2301.08243>
- [58] A. Bardes, J. Ponce, and Y. LeCun, “Vicreg: Variance-invariance-covariance regularization for self-supervised learning,” in *Proceedings of ICLR*, 2022. [Online]. Available: <https://arxiv.org/abs/2105.04906>
- [59] A. Bardes, Q. Garrido, J. Ponce *et al.*, “V-jepa: Latent video prediction for visual representation learning,” *arXiv preprint arXiv:2404.08471*, 2024. [Online]. Available: <https://arxiv.org/abs/2404.08471>
- [60] Meta AI *et al.*, “V-jepa 2.1: Unlocking dense features in video self-supervised learning,” *arXiv preprint arXiv:2603.14482*, 2026. [Online]. Available: <https://arxiv.org/abs/2603.14482>
- [61] A. Zou *et al.*, “Vla-jepa: Enhancing vision-language-action models with latent world modeling,” *arXiv preprint arXiv:2602.10098*, 2026. [Online]. Available: <https://arxiv.org/abs/2602.10098>
- [62] K. He, X. Chen, S. Xie *et al.*, “Masked autoencoders are scalable vision learners,” in *Proceedings of CVPR*, 2022. [Online]. Available: <https://arxiv.org/abs/2111.06377>
- [63] J.-B. Grill, F. Strub, F. Altché *et al.*, “Bootstrap your own latent: A new approach to self-supervised learning,” in *Advances in Neural Information Processing Systems*, 2020. [Online]. Available: <https://arxiv.org/abs/2006.07733>
- [64] T. Yuan, Z. Dong, Y. Liu, and H. Zhao, “Fast-wam: Do world action models need test-time future imagination?” *arXiv preprint arXiv:2603.16666*, 2026. [Online]. Available: <https://arxiv.org/abs/2603.16666>
- [65] X. Zhang, X. Zeng, and W. Zhang, “From world models to world action models: A concise tutorial for robotics,” *arXiv preprint arXiv:2607.00836*, 2026. [Online]. Available: <https://arxiv.org/abs/2607.00836>
- [66] Z. Zhu *et al.*, “Irasim: Learning interactive real-robot action simulators,” in *Proceedings of ICCV*, 2025. [Online]. Available: <https://arxiv.org/abs/2406.14540>

- [67] S. Du *et al.*, “Dynamic world action models for generalizable robot manipulation,” in *Proceedings of ICCV*, 2025. [Online]. Available: <https://arxiv.org/abs/2502.15660>
- [68] DAWN Team *et al.*, “Dawn: A world-action pretrained model for embodied agents,” *arXiv preprint arXiv:2605.11550*, 2026. [Online]. Available: <https://arxiv.org/abs/2605.11550>
- [69] N. P. Garg *et al.*, “World-language-action models,” *arXiv preprint arXiv:2606.05979*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.05979>
- [70] J. Tobin, R. Fong, A. Ray *et al.*, “Domain randomization for transferring deep neural networks from simulation to the real world,” in *Proceedings of IROS*, 2017. [Online]. Available: <https://arxiv.org/abs/1703.06907>
- [71] S. James, A. J. Davison, and E. Johns, “Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task,” in *Proceedings of CoRL*, 2017. [Online]. Available: <https://arxiv.org/abs/1707.02267>
- [72] J. Collins, S. Chand, A. Vanderkop, and D. Howard, “A review of physics simulators for robotic applications,” *IEEE Access*, 2021. [Online]. Available: <https://arxiv.org/abs/2103.04616>
- [73] V. Makoviychuk, L. Wawrzyniak, Y. Guo *et al.*, “Isaac gym: High performance gpu-based physics simulation for robot learning,” *arXiv preprint arXiv:2108.10470*, 2021. [Online]. Available: <https://arxiv.org/abs/2108.10470>
- [74] J. Gu, F. Xiang, X. Li *et al.*, “Maniskill2: A unified benchmark for generalizable manipulation skills,” in *Proceedings of ICLR*, 2023. [Online]. Available: <https://arxiv.org/abs/2302.04659>
- [75] X. Puig, E. Undersander, A. Szot *et al.*, “Habitat 3.0: A co-habitat for humans, avatars and robots,” in *Proceedings of ICLR*, 2024. [Online]. Available: <https://arxiv.org/abs/2310.13724>
- [76] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun, “Carla: An open urban driving simulator,” in *Proceedings of CoRL*, 2017. [Online]. Available: <https://arxiv.org/abs/1711.03938>
- [77] E. Alpaydin, *Introduction to Machine Learning*, 4th ed. MIT Press, 2020.

- [78] M. Lyu, Y. Sun, Y. Jia *et al.*, “Foresightsafety-vla: A unified diagnostic safety benchmark for vision-language-action models,” *arXiv preprint arXiv:2606.27079*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.27079>
- [79] B. Or, “Silent failures in physical ai: A literature review of runtime action authorization for autonomous systems,” *arXiv preprint arXiv:2606.00090*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.00090>
- [80] H. Soh and E. Lim, “Action hallucination in generative visual-language-action models,” *arXiv preprint arXiv:2602.06339*, 2026. [Online]. Available: <https://arxiv.org/abs/2602.06339>
- [81] Q. Li, B. Yin, W. Huang *et al.*, “Vision-language-action safety: Threats, challenges, evaluations, and mechanisms,” *arXiv preprint arXiv:2604.23775*, 2026. [Online]. Available: <https://arxiv.org/abs/2604.23775>
- [82] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?” in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: <https://arxiv.org/abs/1703.04977>
- [83] B. Lakshminarayanan, A. Pritzel, and C. Blundell, “Simple and scalable predictive uncertainty estimation using deep ensembles,” in *Advances in Neural Information Processing Systems*, 2017. [Online]. Available: <https://arxiv.org/abs/1612.01474>
- [84] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” in *Proceedings of ICLR*, 2017. [Online]. Available: <https://arxiv.org/abs/1610.02136>
- [85] D. Amodei, C. Olah, J. Steinhardt *et al.*, “Concrete problems in ai safety,” *arXiv preprint arXiv:1606.06565*, 2016. [Online]. Available: <https://arxiv.org/abs/1606.06565>
- [86] J. García and F. Fernández, “A comprehensive survey on safe reinforcement learning,” *Journal of Machine Learning Research*, vol. 16, 2015. [Online]. Available: <https://jmlr.org/papers/v16/garcia15a.html>
- [87] K.-C. Hsu, H. Hu, and J. F. Fisac, “The safety filter: A unified view of safety-critical control in autonomous systems,” *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, pp. 47-72, 2024.
- [88] A. N. Angelopoulos and S. Bates, “A gentle introduction to conformal prediction and distribution-free uncertainty quantification,” *arXiv preprint*

- arXiv:2107.07511*, 2022. [Online]. Available: <https://arxiv.org/abs/2107.07511>
- [89] M. Althoff, “An introduction to cora 2015,” in *Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.
- [90] J. F. Fisac, K. K. Akametalu, M. N. Zeilinger *et al.*, “A general safety framework for learning-based control in uncertain robotic systems,” *IEEE Transactions on Automatic Control*, 2019. [Online]. Available: <https://arxiv.org/abs/1705.01292>
- [91] A. D. Ames, S. Coogan, M. Egerstedt *et al.*, “Control barrier functions: Theory and applications,” in *Proceedings of the European Control Conference*, 2019. [Online]. Available: <https://arxiv.org/abs/1903.11199>
- [92] D. Seto, B. H. Krogh, L. Sha, and A. Chutinan, “The simplex architecture for safe online control system upgrades,” in *Proceedings of the 1998 American Control Conference*, 1998, pp. 3504-3508.
- [93] M. Phan *et al.*, “The black-box simplex architecture for runtime assurance of autonomous systems,” *arXiv preprint arXiv:2102.12981*, 2021. [Online]. Available: <https://arxiv.org/abs/2102.12981>
- [94] B. Or, “Can predicted dynamics exist in the physical world?” *arXiv preprint arXiv:2606.00089*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.00089>
- [95] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic concepts and taxonomy of dependable and secure computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, 2004.
- [96] N. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press, 2011.
- [97] E. Tabassi, “Artificial intelligence risk management framework (ai rmf 1.0),” National Institute of Standards and Technology, Tech. Rep. NIST AI 100-1, 2023.
- [98] International Organization for Standardization, *ISO 10218-1:2025 Robotics-Safety Requirements-Part 1: Industrial Robots*, 2025. [Online]. Available: <https://www.iso.org/standard/73933.html>
- [99] -, *ISO 10218-2:2025 Robotics-Safety Requirements-Part 2: Industrial Robot Applications and Robot Cells*, 2025. [Online]. Available: <https://www.iso.org/standard/73934.html>

- [100] J. Knight, *Fundamentals of Dependable Computing for Software Engineers*. CRC Press, 2012.
- [101] L. Sha *et al.*, “Using simplicity to control complexity,” *IEEE Software*, vol. 18, no. 4, pp. 20-28, 2001.
- [102] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. MIT Press, 2022. [Online]. Available: <https://algorithmsbook.com/>
- [103] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *International Conference on Learning Representations*, 2014. [Online]. Available: <https://arxiv.org/abs/1312.6114>
- [104] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. M. M. Shafiullah, and L. Pinto, “OK-Robot: What really matters in integrating open-knowledge models for robotics,” *Robotics: Science and Systems*, 2024. [Online]. Available: <https://arxiv.org/abs/2401.12202>
- [105] Z. Fu, T. Z. Zhao, and C. Finn, “Mobile ALOHA: Learning bimanual mobile manipulation with low-cost whole-body teleoperation,” *arXiv preprint arXiv:2401.02117*, 2024. [Online]. Available: <https://arxiv.org/abs/2401.02117>
- [106] K. Black, N. Brown, J. Darpinian *et al.*, “ $\pi_{0.5}$: A vision-language-action model with open-world generalization,” *arXiv preprint arXiv:2504.16054*, 2025. [Online]. Available: <https://arxiv.org/abs/2504.16054>
- [107] J. Bjorck, F. Castañeda, N. Cherniadev *et al.*, “GR00T N1: An open foundation model for generalist humanoid robots,” *arXiv preprint arXiv:2503.14734*, 2025. [Online]. Available: <https://arxiv.org/abs/2503.14734>
- [108] Gemini Robotics Team, “Gemini robotics: Bringing AI into the physical world,” *arXiv preprint arXiv:2503.20020*, 2025. [Online]. Available: <https://arxiv.org/abs/2503.20020>
- [109] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*, 2nd ed. Springer, 2016.
- [110] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*, 2nd ed. Wiley, 2020.
- [111] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006. [Online]. Available: <https://lavalle.pl/planning/>



-
- [112] H. Choset, K. M. Lynch, S. Hutchinson *et al.*, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
 - [113] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*, 2nd ed. Nob Hill Publishing, 2017.
 - [114] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2nd ed. Springer, 2022.

About the author

ArtificialGate



Dr. Barak Or, Ph.D. is an artificial intelligence researcher, entrepreneur, author, and educator working at the intersection of deep learning, autonomous systems, navigation, control, and reliable real-world deployment.

He is an external lecturer at the Technion - Israel Institute of Technology and at Reichman University, and serves as Academic Director at the Google and Reichman AI Tech School. He founded ArtificialGate to connect rigorous AI education with practical engineering capability, and leads applied research on learning-based systems, uncertainty, and robust autonomy.

Dr. Or earned a Ph.D. in machine-learning-based inertial navigation from the University of Haifa, an M.Sc. and B.Sc. in Aerospace Engineering from the Technion, and a B.A. in Economics and Management with honors from the Technion. His published teaching work includes *Modern Deep Learning Foundations*. He has authored more than 30 publications and patents in artificial intelligence, navigation, and autonomous systems.

His entrepreneurial work includes a previous startup in GPS-denied navigation. At the time of writing, he is the founder of STATE16, which develops a reliability layer for Physical AI models. His research spans predictive dynamics, runtime authorization, inertial sensing, foundation models, and the engineering boundaries between model outputs and physical action.

This book reflects a recurring concern in that work: intelligence is not complete when a model can describe or predict a world. For Physical AI, it must produce useful action while respecting time, embodiment, uncertainty, and physical constraints.