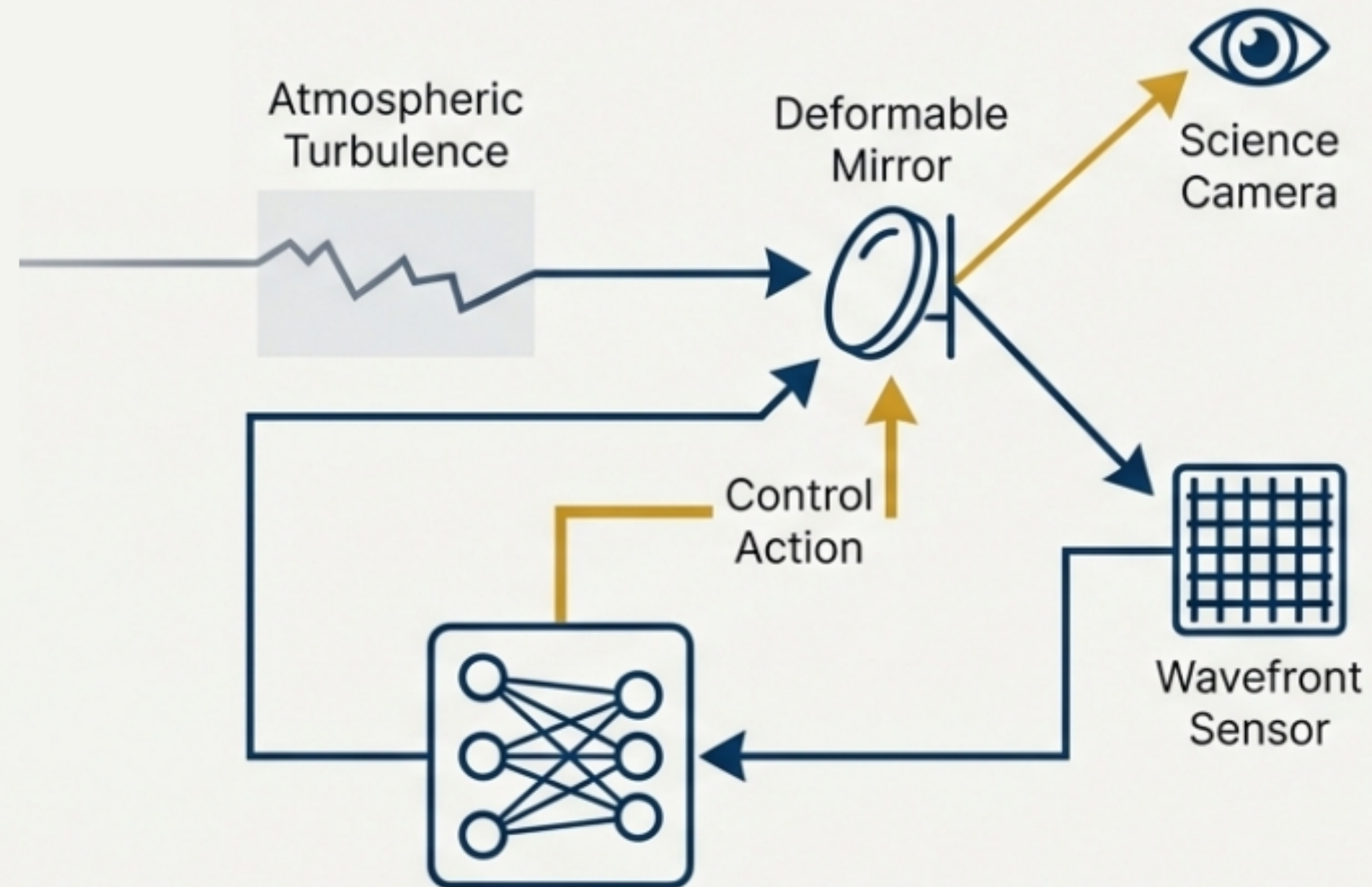


Reimagining Adaptive Optics Control with Reinforcement Learning

A direct path from system performance to control policy



Classical Adaptive Optics faces persistent, real-world limitations.

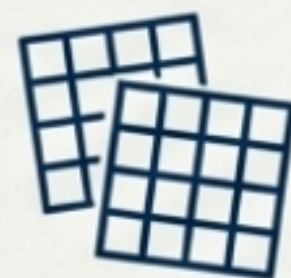
Even with meticulous modeling and calibration, traditional AO controllers struggle with a set of well-known issues that degrade performance.



Temporal Error: Inherent system delays (WFS read-out, computation, DM response) mean we are always correcting a past state of the atmosphere.



Wind-Driven Halo: High-wind conditions



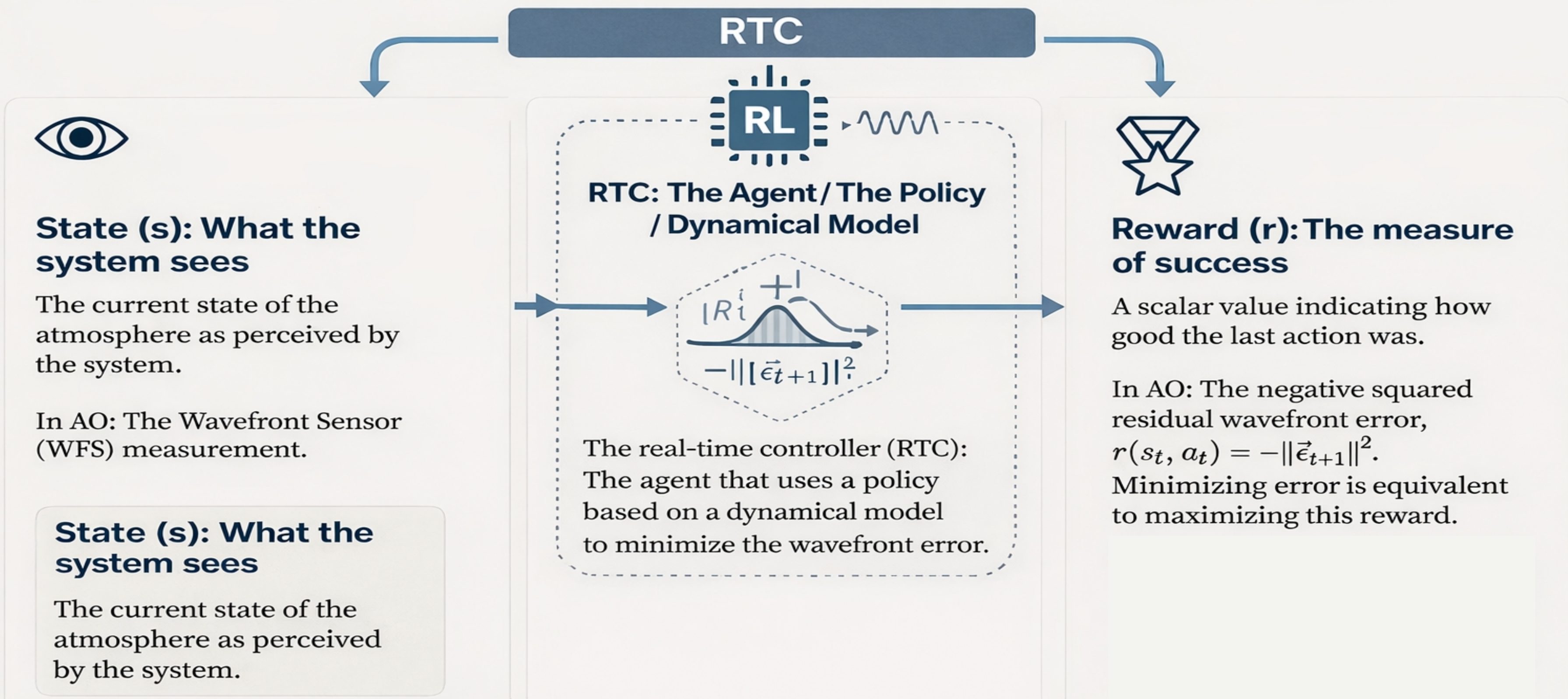
Misregistration: Minor, unavoidable misalignments between the WFS and the DM introduce non-linear control errors.



Noise: Detector noise and photon-starved scenarios corrupt measurements and impact stability.

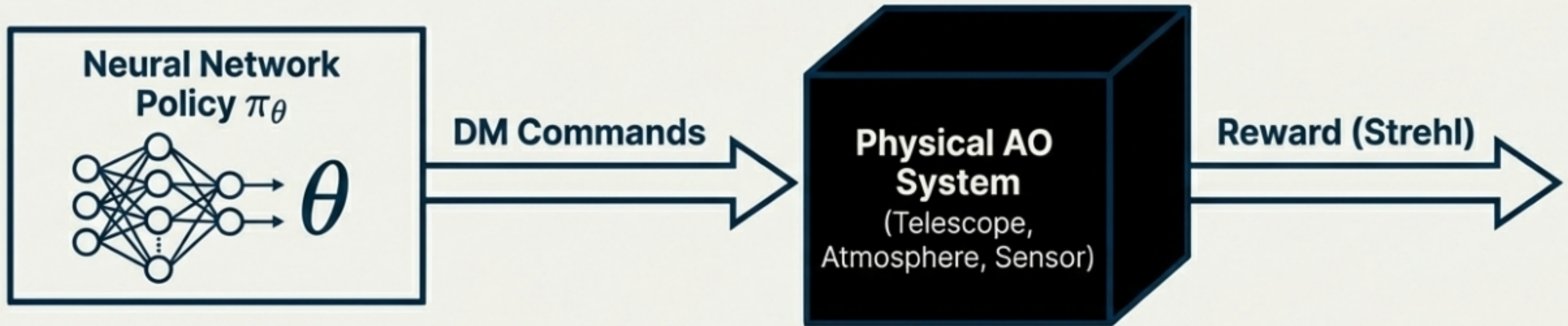
Translating Adaptive Optics into the language of Reinforcement Learning

The core components of an RL problem map directly onto the physical elements of an AO system.



The critical challenge: The reward function is a “black box”.

To learn, the algorithm needs to know how to adjust its parameters (θ) to get more reward. This requires calculating a gradient. But the reward—the Strehl ratio or residual error—comes from a physical sensor, not a clean mathematical function.



We have no analytical gradient.

$$\frac{\partial \text{Strehl}}{\partial \theta} = \text{???}$$

Is it possible to maximize this reward without a model?

Yes.

The Policy Gradient Theorem unlocks model-free learning.

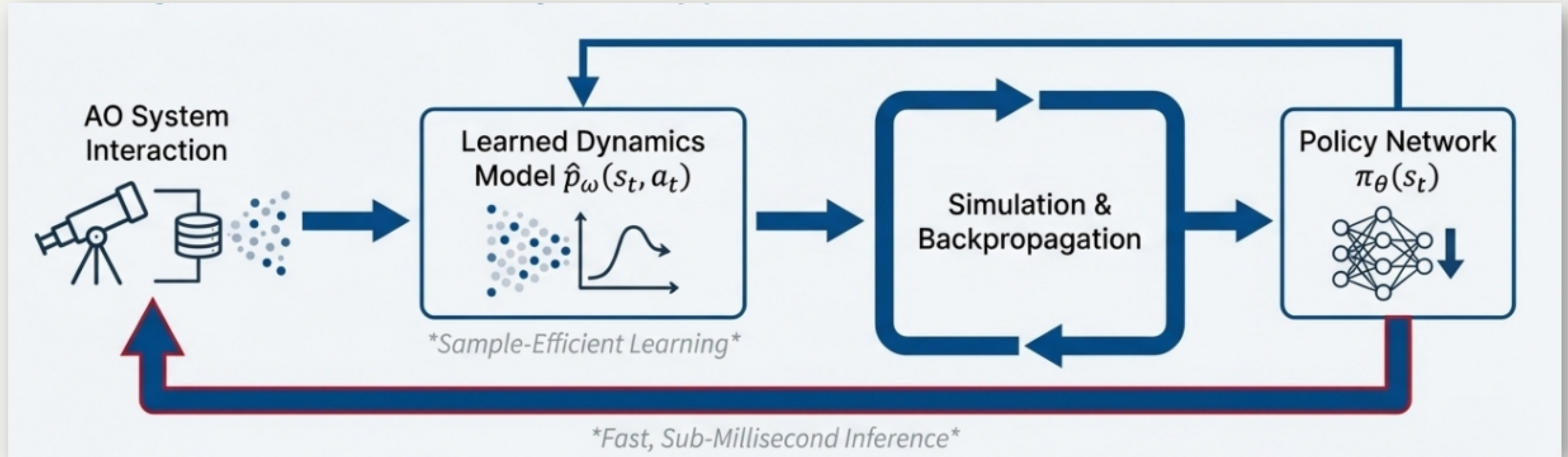
This theorem is the mathematical foundation of modern model-free RL. It provides a way to estimate the gradient of the long-term reward with respect to the policy's parameters, using only the outcomes of actions taken by the policy. We don't need to differentiate through the black box.

****Key Insight**:** The Policy Gradient Theorem allows us to update our controller by correlating actions with the rewards that followed, effectively telling the network: “Do more of what led to good results, and less of what led to bad results.”

$$\underbrace{\nabla_{\theta} J(\pi_{\theta})}_{\text{Desired Gradient}} = \mathbb{E}_{\tau \sim \pi_{\theta}} \left(\underbrace{\text{Action}}_{\text{Hand icon}} \times \underbrace{\text{Reward}}_{\text{Star icon}} \right)$$

Estimated from experienced actions and rewards.

But Model-Based is a good way to do it:



PO4AO is a model-based policy optimization algorithm :

- Uses a learned dynamics model to generate simulated trajectories.
- It directly trains a policy network by backpropagating gradients through these simulated trajectories.

The goal is not instantaneous correction, but long-term stability.

A successful controller must optimize its performance over the entire observation, not just for the next timestep. This is the fundamental premise of maximizing the long-term cumulative reward in RL.

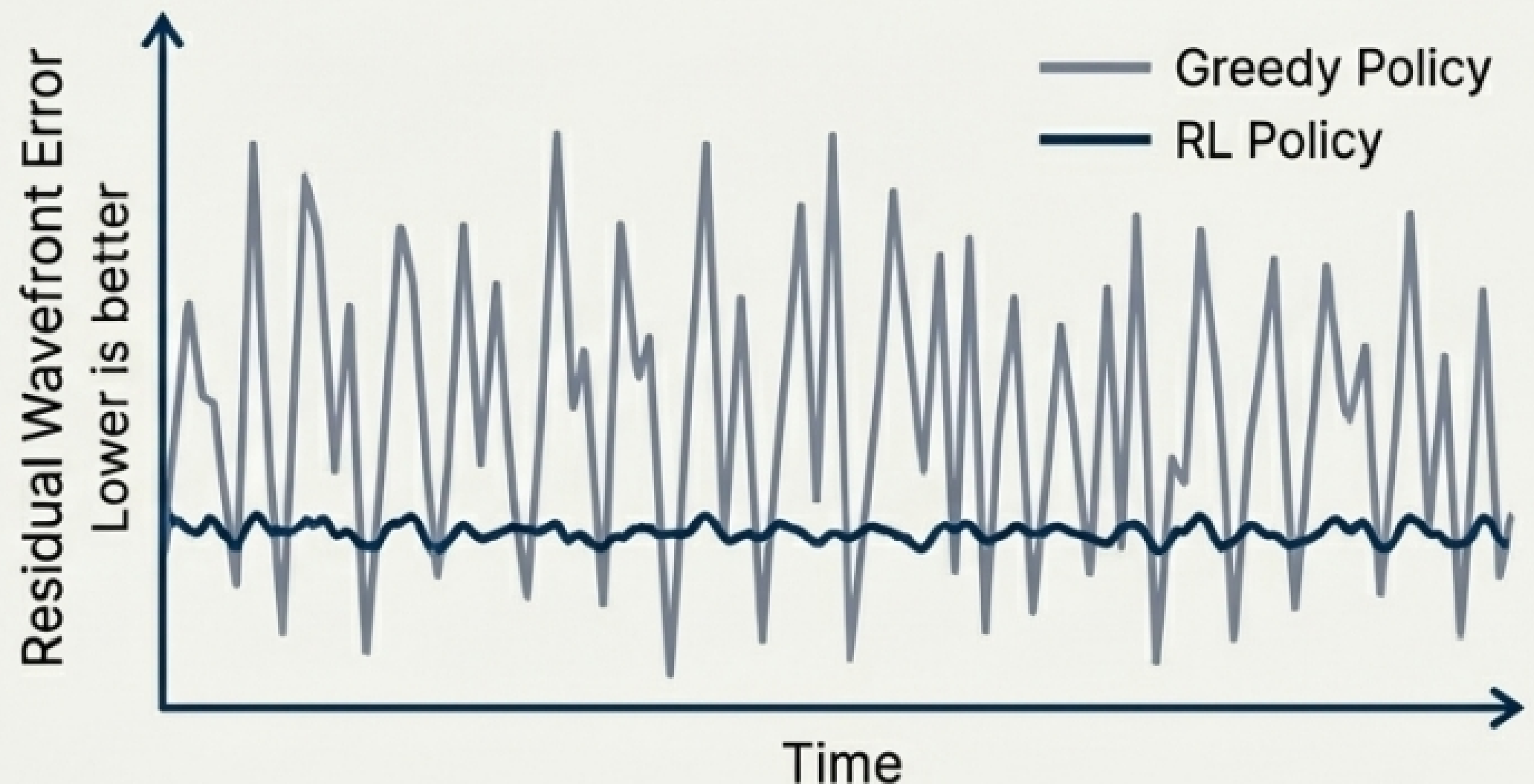
Step 1: Define the goal.

“I want a controller that minimizes the residual wavefront error (which is equivalent to maximizing the Strehl ratio).”

Step 2: Define “minimizes” over time.

“Minimizes the error throughout the entire observation, not just instantaneously.”

Controller Performance Over Time

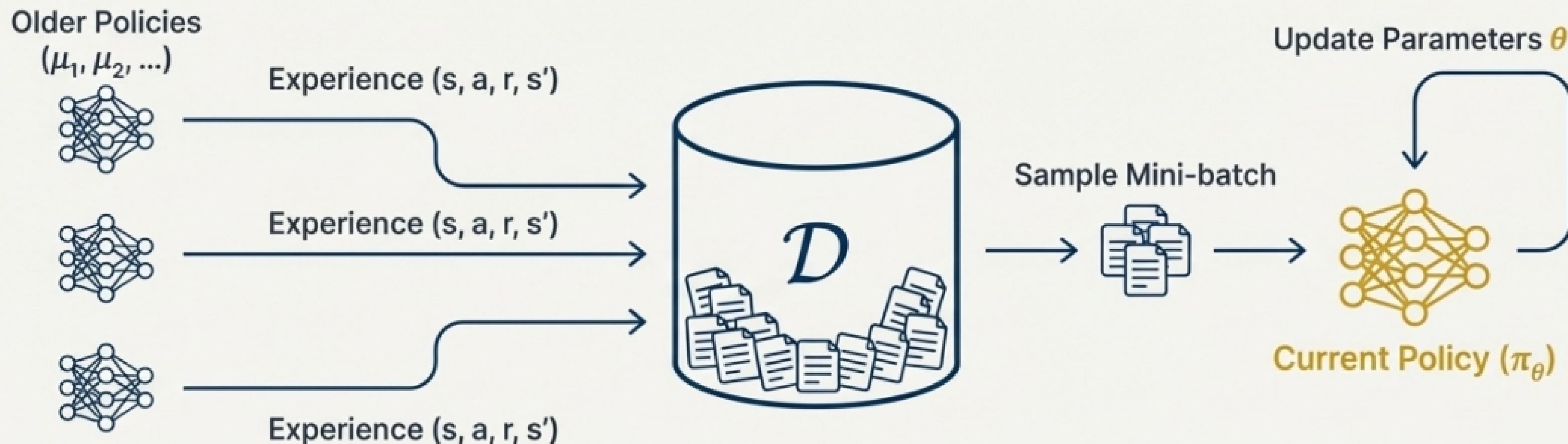


Learning efficiently from past experience: Off-Policy RL.

To maximize data efficiency, we don't discard old experiences. An off-policy agent can learn from data generated by older versions of its own policy.

Definition: In reinforcement learning, an off-policy method learns about a **target policy** (π) while using data generated by a different **behavior policy** (μ).

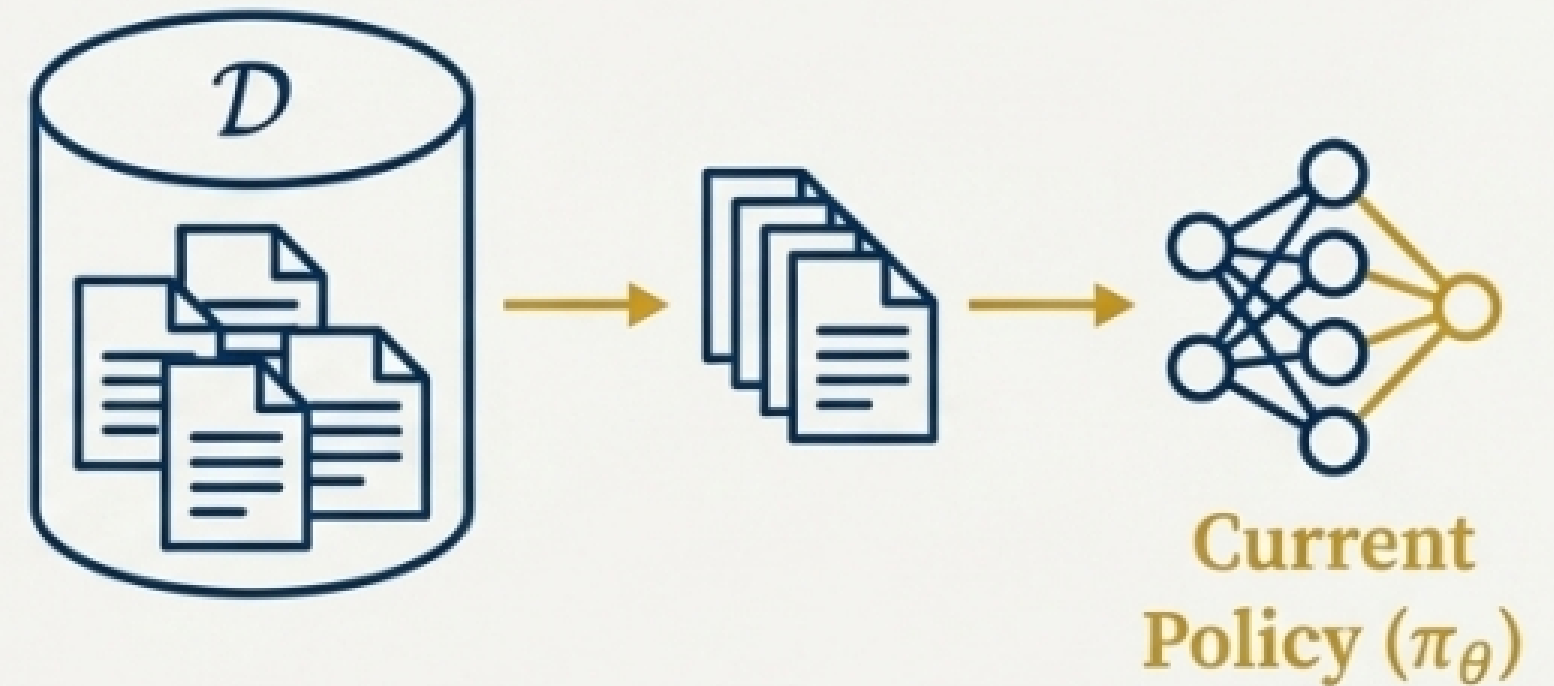
In practice: The actions actually taken by the agent (under older policies, μ) are stored in a “replay buffer” and used to optimize the current policy (π). This stabilizes and accelerates learning.



PO4AO is an off-policy algorithm.

PO4AO leverages off-policy learning to make the training process practical and robust.

1. **Experience Storage:** PO4AO uses a replay buffer ($\mathcal{D}$) to store past transitions (state, action, reward) from previous control episodes.
2. **Sampled Updates:** It updates the current control policy (π_θ) by sampling mini-batches from this buffer.



This means the transitions used for learning were generated by earlier, less-optimal versions of the policy ($\mu \neq \pi$). This is the definition of off-policy learning.

The KL Filter: Bridging abstract AI commands and real-world mirror movements.

The neural network's output is a set of abstract numerical values. The KL filter serves two essential functions to translate these into valid, stable commands for the Deformable Mirror.



Function 1: Physical Projection

Transforms the abstract values into a physically realizable mirror shape by projecting them onto the basis of controllable mirror modes.

Function 2: Regularization

Removes non-controllable or poorly-sensed modes (like waffle, tip/tilt if handled separately) to ensure the commands are stable and effective.