Prediction, Memory, and Motor Adaptation

A proposed control architecture informed by neuroscience

Ontos Research

8 November 2026

Abstract

A robot lifting a box receives camera images, joint positions, and gripper-force readings at different times. To keep moving safely, its controller needs to estimate where the box is between images and revise that estimate if contact changes. A second robot performing the same task has different joints, sensors, and actuator limits, so the first robot's motor commands cannot simply be replayed. We propose a controller that estimates the state of the task, predicts the result of candidate movements, consults records of relevant failures, and generates commands for the body performing the work. We relate these design choices to research on internal motor models, spatial memory, and replay, and specify the proposed state-estimation and control equations. The paper describes tests using delayed camera images, changed payloads, and a robot arm excluded from training. No experiments on the integrated architecture are reported; its benefits remain hypotheses.

Research contributor: Promise Oluwadare

1 How a robot controls its arm between camera images

Consider a robot arm lifting a box. Its camera estimates where the box is. Joint encoders report how far the arm has moved, and a sensor at the gripper measures the force applied to the box. The controller needs all three kinds of information to lift the box without dropping it.

Those sensors do not update together. In a robotics study on multiscale sensor fusion, Singh and colleagues discuss cameras operating at about 30 Hz, joint measurements at 100 Hz, and force-torque sensors at 500 Hz [13]. At those example rates, a camera provides a new image about every 33 milliseconds; the joint encoders update roughly every 10 milliseconds. The arm keeps moving between images. During that interval, the controller must estimate where the box has moved and whether the grasp is still secure. These rates come from the study and are not specifications for all robots.

Neuroscience researchers have studied a similar problem in biological movement. Wolpert, Miall, and Kawato reviewed evidence in 1998 that the cerebellum may use internal models to predict the effects of motor commands [1]. A forward model estimates where the body will be after a movement, before delayed sensory information arrives. Wolpert and Kawato also proposed using multiple paired forward and inverse models to handle different movement

conditions [2]. In a robot, a comparable predictor could estimate the next position of the gripper and compare it with the next sensor reading.

The problem changes when the same instruction is given to another robot arm. Both machines may be asked to move the box to the same shelf, but they can have different reach, grippers, joint limits, and actuator commands. Copying the first robot's joint movements is unlikely to place the second robot in the same position. The shared information is the task goal and what happened during earlier attempts, including any observed failure.

This paper proposes a controller with separate representations of the task and the robot's physical body. It combines an estimate of the current scene with predictions of candidate movements. It can consult recorded failures when choosing the next action, but each arm must produce commands that meet its own mechanical limits. We specify the equations for those components and experiments that would show whether they improve control compared with simpler methods. The combined system has not yet been tested.

2 Research on predicting movement and remembering past actions

2.1 Predicting movement and tracking the position of the body

Suppose the robot commands its gripper to lift the box by five centimeters. A forward model uses the current state and that command to estimate the arm's next joint angles, the force at the gripper, and the position of the box. The controller compares those values with subsequent measurements. A large disagreement means the prediction was wrong; it does not, by itself, identify the cause. Unexpected weight, a shifted grasp, sensor error, and model error can produce similar residuals.

Wolpert, Goodbody, and Husain studied patients with superior parietal lesions and examined the maintenance of internal body representations [3]. Their findings support the role of stored state information in sensorimotor integration. Rao and Ballard later proposed a predictive-coding account of visual cortex in which predictions are compared with incoming signals [9]. These studies concern different neural processes. We take the narrower lesson that control requires a maintained estimate and a way to register disagreement when fresh evidence arrives.

In a robot, a state estimator combines recent sensor readings with motor commands to estimate quantities that no sensor can measure continuously. For a partially occluded grasp, the estimate may preserve the last known object pose and a distribution over possible contact states. As the robot moves without seeing the object, uncertainty can increase. When a useful image or force reading arrives, the estimate is corrected.

2.2 How memories of earlier actions can affect a later decision

Stachenfeld, Botvinick, and Gershman proposed a predictive-map interpretation of hippocampal representations, relating spatial memory to expected future states under a policy [4]. Mat-

tar and Daw developed a computational account of prioritized replay: previously experienced transitions can be revisited according to how much that replay is expected to improve later decisions [5]. Neither paper supplies an off-the-shelf memory algorithm for a robotic arm.

A robot should retrieve a past failure only when knowing about that failure could change the action it is about to take. Suppose a gripper previously dropped a box while turning through a narrow passage. A useful record contains the estimated contact state before the turn, the command, the predicted change, the observed slip, and the task outcome. Retrieving that record may justify slowing the turn or choosing a different grasp. A recording of the same room with no link to an action or outcome is less informative for that decision.

Schultz, Dayan, and Montague found dopamine activity associated with the difference between predicted and received reward [6]. Doya's review compared learning functions often attributed to cerebellum, basal ganglia, and cortex [7]. These findings motivate examining task outcomes alongside prediction errors. Our design does not assign a software module to each brain region or claim to simulate its biological computation.

3 What existing robot-learning experiments have shown

Existing robotics experiments show which parts of the proposal already have engineering precedents. Banino and colleagues trained artificial agents whose representations developed grid-like properties and supported vector-based navigation [8]. MetaMorph conditioned a controller on the structure of modular robots and tested it on held-out morphologies [10]. The Open X-Embodiment collaboration assembled more than one million real-robot trajectories across 22 embodiments and reported benefits from training across robot datasets [11]. Those experiments support learned representations and transfer in their respective settings; they do not show that a motor command can be reused unchanged on any machine.

The RT-1 study at Google trained a robotics transformer on approximately 130,000 episodes spanning more than 700 tasks, collected with 13 robots over 17 months [14]. In one mixed-data evaluation, adding Kuka bin-picking data to the Everyday Robots dataset improved the reported bin-picking result on the Everyday Robots platform from 22% to 39%, while performance on its original classroom evaluation remained similar [14]. The figures come from that study's evaluation, not from Ontos. They illustrate the value of shared training data and the difficulty of transfer between platforms.

Hafner and colleagues trained DreamerV3 to learn a predictive world model and improve behavior through imagined rollouts, reporting experiments across more than 150 tasks [12]. The Barkour project at Google offers a different point of comparison. It measured quadruped agility on a defined course with weave poles, an incline, a jump, and end tables; its controller combined learned locomotion policies with recovery behavior [15]. The study assessed success, speed, and failures on real hardware rather than relying on the appearance of fluent motion.

None of these studies tested the full combination proposed here. We would measure whether

checking prediction errors, retrieving selected failures, and adapting a limited set of body parameters improves a predictive controller's performance. A failure to improve on a recurrent policy or an ordinary model-predictive controller would be an important finding.

4 Representing the same task on two different robot arms

Moving the box to a target shelf is a task both arms can share. The motor commands needed to do it depend on the arm. We encode that difference explicitly. Let m identify the robot and let μ_m describe its joints, actuator limits, sensors, and command interface. At time t, the physical state s_t is not directly observable. The robot receives observations o_t^m and issues commands u_t^m. We model its interaction with the environment as

$$\begin{aligned} s_{t+1} &\sim p(s_{t+1} \mid s_t, u_t^m, \mu_m), \\ o_t^m &\sim p(o_t^m \mid s_t, \mu_m). \end{aligned}$$

A single measurement rarely determines all variables needed for control. The belief state b_t^m represents the distribution over possible states given the recorded observations, previous commands, and robot description:

$$b_t^m(s) = p(s_t = s \mid o_{0:t}^m, u_{0:t-1}^m, \mu_m).$$

An exact belief distribution is usually too costly to maintain for high-dimensional sensor data. We therefore propose a compact learned representation $z_t = (z_t^{\text{task}}, z_t^{\text{body}})$ and an uncertainty estimate Σ_t. For the box task, the task portion could encode the box pose, target location, and whether the intended transfer has been completed. The body portion could encode joint positions, grasp geometry, contact estimates, and actuator response. Neither portion can be defined without regard to the other: a target pose is relevant only if the robot can reach it, and a contact estimate depends on gripper geometry.

A recurrent estimator q_ψ receives the previous estimate, uncertainty, most recent command, new observation, and robot description:

$$q_\psi(z_t, \Sigma_t \mid z_{t-1}, \Sigma_{t-1}, u_{t-1}^m, o_t^m, \mu_m).$$

The transition and observation models predict the next state and expected sensor measurements:

$$\begin{aligned} \hat{z}_{t+1} &\sim p_\theta(z_{t+1} \mid z_t, u_t^m, \mu_m), \\ \hat{o}_{t+1}^m &\sim p_\phi(o_{t+1}^m \mid \hat{z}_{t+1}, \mu_m). \end{aligned}$$

The equations define the information each model must receive and predict. They do not specify a trained network. An implementation must still choose state dimensions, sensor encoders,

probability distributions, operating frequency, and compute limits. We expect different robot families to need different input adapters and action decoders. The claim worth testing is whether some task variables and prediction parameters can remain useful when those adapters change.

5 How the controller responds when a box is heavier than expected

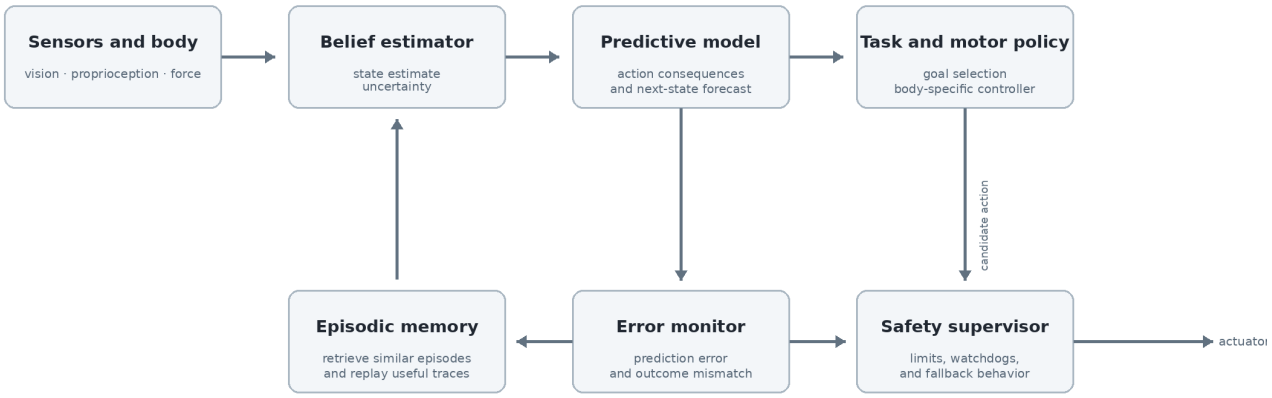


Figure 1: Proposed information flow for a predictive sensorimotor controller. Measurements enter a belief estimator; a forward model and error monitor compare predicted and observed outcomes; memory informs the next decision; and an independent safety supervisor checks the command before actuation.

The following example traces the proposed controller through a box lift. It explains the design; it is not a completed experiment. The camera estimates the box pose. The joint encoders report the arm configuration, and the gripper's force measurement helps estimate contact. Each measurement carries its own timestamp. The estimator combines them with previous commands to maintain a current state and uncertainty.

Before sending the lifting command, the controller predicts how far the joints and box will move and the force reading it expects at the gripper. If the observed displacement is smaller than predicted, the error monitor records that mismatch. The controller then updates its state estimate, considers whether the payload model should change, and generates a new candidate action. A critical sensor loss or an action outside the allowed operating envelope goes to the safety supervisor instead of an unconstrained retry.

For measured vector y_t, predicted measurement $\hat{y}_t$, and residual covariance S_t, the error monitor can compute the normalized squared residual

$$r_t = (y_t - \hat{y}_t)^\top S_t^{-1} (y_t - \hat{y}_t).$$

A large residual means the sensor reading differs from the prediction more than the model's uncertainty would suggest. The number alone cannot reveal whether the box is heavier than expected, the grasp has slipped, or a sensor is miscalibrated. Those explanations require addi-

tional measurements or a diagnostic test. Thresholds could be estimated from held-out runs; a fixed threshold chosen from a clean Gaussian model would be unreliable under the correlated errors and missing observations common on real hardware.

5.1 Converting a box-placement goal into motor commands

For the box-transfer task, the short-term goal might be to raise the box clear of the shelf before moving it sideways. The task policy π_H selects such a goal g_t from the estimated task state, instruction c_t, and retrieved episodes M_t. The lower-level controller π_L translates that goal into a command for robot m:

$$g_t \sim \pi_H(g \mid z_t^{\text{task}}, c_t, M_t),$$
$$u_t^m \sim \pi_L(u \mid z_t^{\text{body}}, z_t^{\text{task}}, g_t, \mu_m).$$

For one arm, the command may specify joint torques; for another, joint positions or end-effector targets. A model-predictive implementation could choose a command sequence over horizon H by minimizing

$$\min_{u_{t:t+H-1}} \mathbb{E}\left[\sum_{k=0}^{H-1} \ell_{\text{task}}(\hat{z}_{t+k}, g_t) + \lambda_u \|u_{t+k} - u_{t+k-1}\|^2 + \lambda_r R(\hat{z}_{t+k}, \Sigma_{t+k})\right].$$

The task loss measures progress toward the goal, the command penalty discourages abrupt changes, and R penalizes selected risks or uncertain outcomes. The solver must also honor platform-specific joint, torque, contact, and workspace constraints. This objective is a candidate formulation; its terms and weights require definition for each experimental task.

5.2 Using a previous failed grasp to plan the next attempt

We propose storing selected episodes with a task estimate, robot description, action, prediction, observed next state, goal, and outcome:

$$e_i = (z_i^{\text{task}}, \mu_m, u_i^m, \hat{z}_{i+1}, z_{i+1}, g_i, \text{outcome}_i).$$

If the arm encounters another box at a similar angle, the memory system could retrieve a previous attempt in which that grasp slipped. The task policy can use the old outcome to consider another grasp or a slower motion. The stored action cannot be executed verbatim on a second arm unless its geometry and control interface permit it. The retrieval algorithm therefore needs a check on embodiment compatibility as well as task similarity. The proposed memory ranking has not been optimized or validated.

5.3 Updating the robot's estimate of payload and friction

If the new box is heavier than the controller expected, the robot may need to revise its estimate of payload mass before the next lift. We propose adapting a limited parameter set η_m using recent observations $\mathcal{D}_m$:

$$\eta_m^* = \arg\min_{\eta_m} \sum_{i \in \mathcal{D}_m} \|y_{i+1} - \hat{y}_{i+1}(\eta_m)\|^2 + \lambda \|\eta_m - \eta_m^{(0)}\|^2.$$

The regularization term discourages large departures from the initial parameters $\eta_m^{(0)}$. It does not guarantee stable closed-loop control. An implementation would bound the permitted updates, evaluate their effect on held-out observations, and revert to a validated configuration if tracking or safety measurements deteriorate. Changes made on one robot should not silently update the shared model used by the rest of a fleet.

5.4 Checking motor commands against the robot's safety limits

The safety supervisor receives candidate commands and decides whether the physical robot may execute them. It must enforce the platform's actuator and workspace limits and respond to watchdog failures or missing critical sensors. A robot holding a heavy object may need a controlled fallback rather than an instantaneous power cut. The fallback depends on the machine's mechanical design and risk assessment.

The learned model can predict that an action is likely to succeed. The safety supervisor still has authority to reject it. Research on the proposed controller would report both task performance and every intervention by that supervisor.

6 Training with sensor recordings and task outcomes

Each training episode would record camera images, joint and force readings, the commands sent to the robot, the robot's mechanical description, and whether the task succeeded. Where available, object poses, contact state, or privileged simulation measurements can supervise parts of the latent state. Privileged simulation inputs must be identified and excluded from real-world evaluation inputs. The predictive model can learn by comparing its next-state forecasts with subsequent measurements; the task policy can learn from demonstrations or task rewards.

A candidate training objective combines prediction, state estimation, task performance, and cross-body alignment:

$$\mathcal{L} = \alpha \mathcal{L}_{\text{prediction}} + \beta \mathcal{L}_{\text{state}} + \gamma \mathcal{L}_{\text{task}} + \delta \mathcal{L}_{\text{alignment}}.$$

The expression does not identify an optimizer, a dataset split, or the exact target for each

term. Those choices belong in an implementation report. They also create a genuine design question: an image-reconstruction objective may retain texture that has little effect on control, while aggressive alignment across robots may erase contact information that a particular gripper needs. We would tune the coefficients on training and validation data and then measure whether transfer survives on a robot excluded from that process.

7 Testing delayed camera frames, changed loads, and unfamiliar robot arms

To test transfer between robots, we would train and tune the system on two arms with different joint arrangements, then evaluate it on a third arm excluded from training. Each must be physically capable of the same object-transfer tasks. We would also evaluate locomotion on its own task suite, because success on a manipulation benchmark does not establish transfer to a legged robot. Train and test sets must be separated by robot, object identity, layout, and disturbance. Frames from one episode must remain in the same split.

We propose three pre-registered comparisons.

Delayed observations. Introduce controlled camera latency, temporary occlusion, and sensor dropout during object transport. Compare the full system with a recurrent reactive controller trained on the same episodes and compute budget. The test asks whether explicit prediction improves task completion and reduces recovery time. Report uncertainty calibration and safety interventions, since a controller can appear faster by accepting more risk.

Changed load or friction. Alter a payload or contact condition without telling the controller when the change occurs. Compare bounded local adaptation with the same system's adaptation disabled. Measure the number of actions and elapsed time required to return to acceptable tracking, together with failed grasps and constraint violations. Both conditions should receive the same observation history and restart allowance.

Recovery on a held-out arm. Record selected failures during development, then ask the unseen arm to complete physically compatible tasks with and without access to those episodes. Count successful recoveries, repeated failed attempts, memory selections that did not change the action, and harmful retrievals. Task similarity is insufficient evidence of transfer if the retrieved action violates the new arm's reach or gripper constraints.

A classical state estimator with model-predictive control is an additional baseline whenever the system model and measurements support one. The learned architecture should also be ablated by disabling its memory, prediction loss, or body adaptation in turn. Model size, training interactions, inference hardware, control frequency, and sensor rates must be reported for every comparison. Median and 99th-percentile inference latencies are important because occasional slow decisions can matter during contact-rich tasks.

If a recurrent policy matches the proposed system at lower compute cost, the added machinery has no demonstrated value on that benchmark. If retrieval increases repeated failures on

the held-out arm, the memory interface needs revision. If adaptation improves tracking but increases safety interventions, it has failed the proposed criterion. These outcomes would be reported alongside successful trials; the experiments have not yet been run on the integrated architecture.

8 What the results would need to show

The neuroscience papers offer models of how predictions and stored experiences can influence movement [1, 2, 4, 5]. Robotics studies show that predictive representations and some forms of cross-robot training can work in defined experimental settings [10–15]. These are findings from different systems; the papers do not establish that combining them will produce a better controller. None has tested the full architecture specified in this paper.

The next test is specific: when camera frames are delayed, a box becomes heavier, or a new arm is used, does the complete controller finish more tasks than a recurrent policy or a classical model-predictive controller using comparable data and compute? The answer requires held-out robots, measured latency, and a record of failed actions. Until those results exist, the architecture remains a research design from Ontos Research.

9 References

1. Wolpert, D. M., Miall, R. C., & Kawato, M. (1998). Internal models in the cerebellum. *Trends in Cognitive Sciences*, 2(9), 338–347. doi:10.1016/S1364-6613(98)01221-2.
2. Wolpert, D. M., & Kawato, M. (1998). Multiple paired forward and inverse models for motor control. *Neural Networks*, 11(7–8), 1317–1329. doi:10.1016/S0893-6080(98)00066-5.
3. Wolpert, D. M., Goodbody, S. J., & Husain, M. (1998). Maintaining internal representations: The role of the human superior parietal lobe. *Nature Neuroscience*, 1, 529–533. doi:10.1038/2245.
4. Stachenfeld, K. L., Botvinick, M. M., & Gershman, S. J. (2017). The hippocampus as a predictive map. *Nature Neuroscience*, 20, 1643–1653. doi:10.1038/nn.4650.
5. Mattar, M. G., & Daw, N. D. (2018). Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21, 1609–1617. doi:10.1038/s41593-018-0232-z.
6. Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275(5306), 1593–1599. doi:10.1126/science.275.5306.1593.
7. Doya, K. (1999). What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Networks*, 12(7–8), 961–974. doi:10.1016/S0893-6080(99)00046-5.
8. Banino, A., Barry, C., Uria, B., et al. (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557, 429–433. doi:10.1038/s41586-018-0102-6.
9. Rao, R. P. N., & Ballard, D. H. (1999). Predictive coding in the visual cortex: A functional interpretation of some extra-classical receptive-field effects. *Nature Neuroscience*, 2, 79–87. doi:10.1038/4580.

10. Gupta, A., Fan, L., Ganguli, S., & Fei-Fei, L. (2022). MetaMorph: Learning universal controllers with transformers. *International Conference on Learning Representations*. Project and paper.
11. Open X-Embodiment Collaboration. (2024). Open X-Embodiment: Robotic learning datasets and RT-X models. *IEEE International Conference on Robotics and Automation*, 6892–6903. doi:10.1109/ICRA57147.2024.10611477.
12. Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2025). Mastering diverse control tasks through world models. *Nature*, 640, 647–653. doi:10.1038/s41586-025-08744-2.
13. Singh, S., McCann Ramirez, F. E., Varley, J., Zeng, A., & Sindhwani, V. (2022). Continuous control and multiscale sensor fusion with Neural CDEs. *IROS & RSS Imitation Learning Workshop*. Google Research publication.
14. Brohan, A., et al. (2022). RT-1: Robotics Transformer for real-world control at scale. Google Research technical article.
15. Caluwaerts, K., & Iscen, A. (2023). Barkour: Benchmarking animal-level agility with quadruped robots. Google Research technical article.

10 Appendix: Data and code needed to reproduce the experiments

An implementation report must provide the robot description format, joint and sensor mappings, actuator command units, timestamp synchronization, calibration procedures, coordinate-frame conventions, and per-platform safety constraints. For each model, report the architecture, checkpoint, parameter count, training loss, data sources, optimizer, random seeds, and inference device. Data partitions must identify robot embodiments, objects, environments, and disturbances held out from training.

For every real-robot trial, retain time-aligned observations, controller outputs, safety-supervisor interventions, attempted and completed tasks, and reasons for terminating a run. Report the metric definitions and confidence intervals, and provide scripts that reproduce the reported aggregates from the evaluation logs. If a dataset cannot be released, describe its collection and access restrictions and provide an independently executable simulation test with matching task criteria.