Feedback Control as a Chat

What a conversation with the environment reveals about robot control.

Every few weeks, another frontier lab drops a demo of a general-purpose AI model controlling a robot. Now that models like GPT-6 Astra and Gemini 4 Argon are out, people are wiring them directly into robot arms to see how far raw multimodal reasoning can go. A recent study even put these exact frontier models to the test on the LIBERO-10 manipulation benchmark, giving them camera images and task goals to see if they could figure out the right moves on the fly [6].

When you watch these setups, it is easy to focus entirely on how smart the model is. A vision-language model looks at the scene, proposes an action, and gets a new image back. We wanted to step back and ask what this setup actually means from a systems perspective. If you look past the model weights and pay attention to the back-and-forth between the model and the physical world, the whole thing starts to look like a conversation.

Imagine a robot arm reaching for a cup. The gripper closes, and the next camera image shows the cup still sitting on the table. In these new setups, the robot does not just fail out. That image of the missed grasp gets fed right back to the model alongside what it just tried to do. The model acts, the environment pushes back, and the next observation gives the model something to respond to. While testing vision-language models on robots ourselves, we started calling this chatting with the environment.

To a control theorist, this loop is immediately familiar. It is feedback. Looking at it as a conversation is useful because it shifts our focus from building a smarter model to designing a better exchange around the model. What should the controller actually be told after a move? What should it remember from earlier turns? And what happens when the model suggests a move that is completely unsafe?

In this blog, we look at how viewing robot control as a chat connects frontier AI setups to core problems across control theory, from system identification and hybrid mode switching to safety guarantees and explainability.

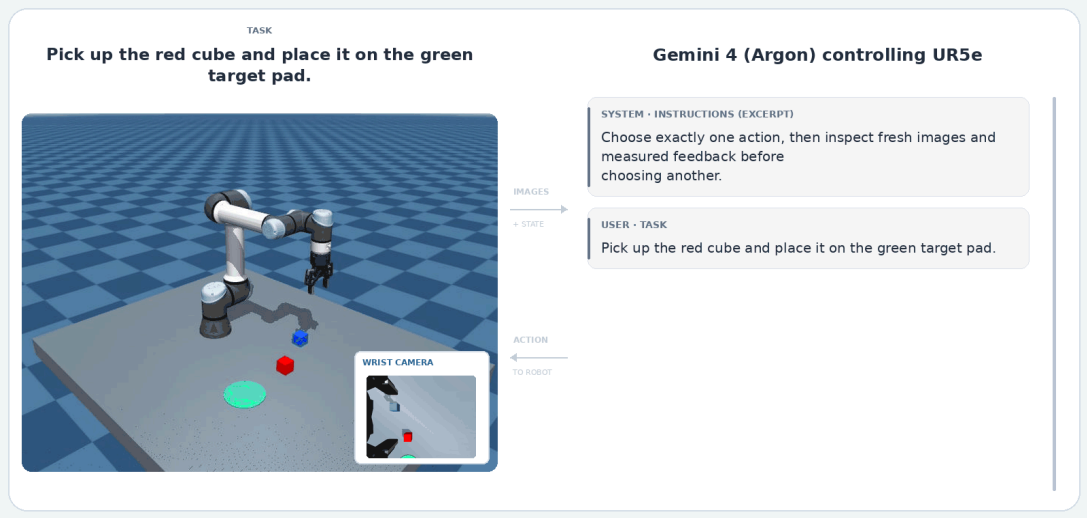

Figure 1: Gemini guides UR5e to pick and place a cube through a visual conversation—observing, acting, and correcting mistakes from feedback.

What is actually happening in this chat?

If you look at the background trace in Figure 1, the interaction is simple. The model gets a camera frame, reasons about where the object is, and outputs a command. When the robot misses or nudges the object out of place, the next image shows the mistake, and the model tries a different approach on the next turn.

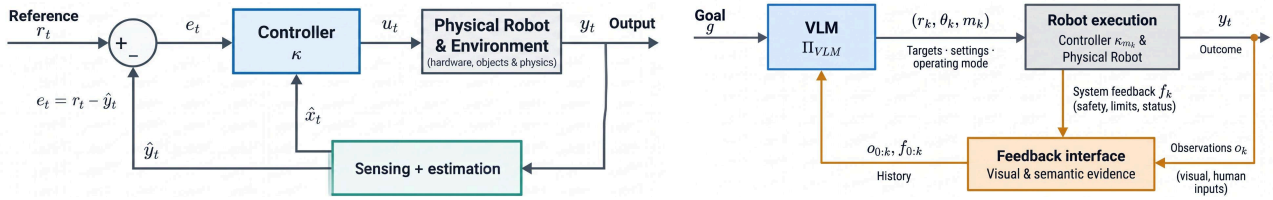

Figure 2: *From numeric tracking to conversational feedback.* ***Left:*** *(a) In a traditional feedback loop, the controller κ uses the tracking error $e_t = r_t - \hat{y}_t$ and estimated state $\hat{x}_t$ to produce the control input u_t for the physical robot and environment. Every event has to be reduced to a numeric vector before the controller can react.* ***Right:*** *(b) In a conversational feedback loop, placing a vision-language model alongside the low-level controller opens up the feedback channel. Robot execution and the environment return system feedback f_k and observations o_k through a feedback interface that collects visual and semantic evidence into the history $o_{0:k}, f_{0:k}$, allowing the VLM to update the target r_k, settings θ_k, and operating mode m_k.*

In a traditional feedback loop (Figure 2a), the system starts with a tracking error:

$$e_t = r_t - \hat{y}_t, \quad u_t = \kappa(e_t, \hat{x}_t)$$

That tracking error tells the robot how far its gripper is from the target coordinate, and the low-level controller κ drives that number to zero. Because a traditional loop only speaks in fixed-dimension numbers, every sensor reading has to be reduced to a scalar or vector before the controller can react to it. A numerical error can tell the arm that it is two millimeters off target, but it cannot explain why the task failed when the gripper reached the exact target coordinate and still came up empty. Maybe the fingers caught the tapered rim of a cup, or maybe the block rotated as the gripper closed.

That is what the chat changes (Figure 2b). Instead of restricting feedback to numbers from a traditional loop, the system can now pass back many different types of feedback across modalities and levels of abstraction. Alongside raw state estimates, a single conversational turn can return visual observations showing how the scene shifted, semantic summaries of physical contact or slip, execution reports explaining which safety constraint blocked a motion, or even a spoken warning from a human standing nearby. A turn in the chat can bundle all of this evidence together with a hypothesis about why the grasp slipped and a plan for how to recover. Recent work like TextGrad showed how textual critiques can act like gradients to update a computational graph [5]. Here, the conversation with the environment does something similar for physical control, using richer forms of feedback to update how the robot acts without changing the model's weights.

How do we add control over the conversation?

Letting a vision-language model chat directly with the hardware is exciting, but we should not expect a slow, general-purpose model to replace the fast feedback loop or guarantee physical safety on its own. The real engineering question is how to put control *around* the chat.

Instead of treating the language model as a drop-in replacement for a motor controller, we can pair it with the low-level controller so the system can act on both numerical errors and richer semantic feedback:

$$(r_k, \theta_k, m_k) = \text{VLM}(g, o_{0:k}, f_{0:k})$$
$$u_t = \kappa_{m_k}(\hat{x}_t, r_k; \theta_k)$$

Here, the low-level controller κ_{m_k} still tracks numerical state $\hat{x}_t$ at time t, while at each conversational turn k, the model VLM takes the goal g, the history of visual observations $o_{0:k}$, and structured system feedback $f_{0:k}$ to update how the robot behaves. Because $f_{0:k}$ is not limited to a tracking error vector, it can carry contact events, constraint violations, execution status, or operator instructions. Using those diverse feedback signals, the model can set a new reference target r_k, tune controller parameters θ_k like speed or stiffness, or switch the discrete operating mode m_k from free-space reaching to compliant grasping.

Looking at the loop in Figure 2b, shaping this conversation comes down to three engineering choices along the path of the signals. First, on the arrow leaving robot execution, we can decide what system feedback f_k gets reported back when an action completes, gets modified, or hits a limit. Second, inside the orange feedback interface, we can decide how raw observations o_k and system reports f_k are filtered and assembled into the history $o_{0:k}, f_{0:k}$ for the model's next turn. Third, on the command path leaving the VLM, we can govern how new targets, settings, and operating modes (r_k, θ_k, m_k) are chosen and timed, including actions taken solely to gather missing information.

The following three sub-sections look at each part of this loop in turn, covering safety feedback, memory management, and active sensing and timing.

1. Making safety filters talk back

The first design choice is what the system reports after an action is attempted or prevented. Safety interventions illustrate why this matters. A camera image may show that the robot has stopped, while an execution report can explain which constraint prevented the requested motion.

Suppose Gemini Argon looks at the scene and proposes a shortcut that drives the arm straight through a fragile obstacle or near a table limit. In a standard safe control setup, we wrap the low-level controller in a safety filter, like a Control Barrier Function based quadratic program [3], that checks the nominal command $u_{nom,t}$ when needed to satisfy specified safety constraints.

If we just clamp the command silently, the arm stops moving, the next camera image looks identical to the last one, and the language model has no idea why. It will often keep asking for the exact same blocked waypoint turn after turn.

Because our supervisory loop takes textual and structured feedback $f_{0:k}$, we can make the safety filter part of the conversation. When the barrier function or joint limit check intervenes, it logs a diagnostic report into f_k explaining which constraint became active and whether the requested motion was clipped or blocked, just as we do on the Unitree G1 humanoid in Figure 3 where joint targets are clamped to a 15° per-turn limit and the resulting tracking errors feed straight into the next prompt. The low-level math still guarantees that the physical constraint is never violated, while the explanation gives the model the context it needs to adjust its plan on the next turn.

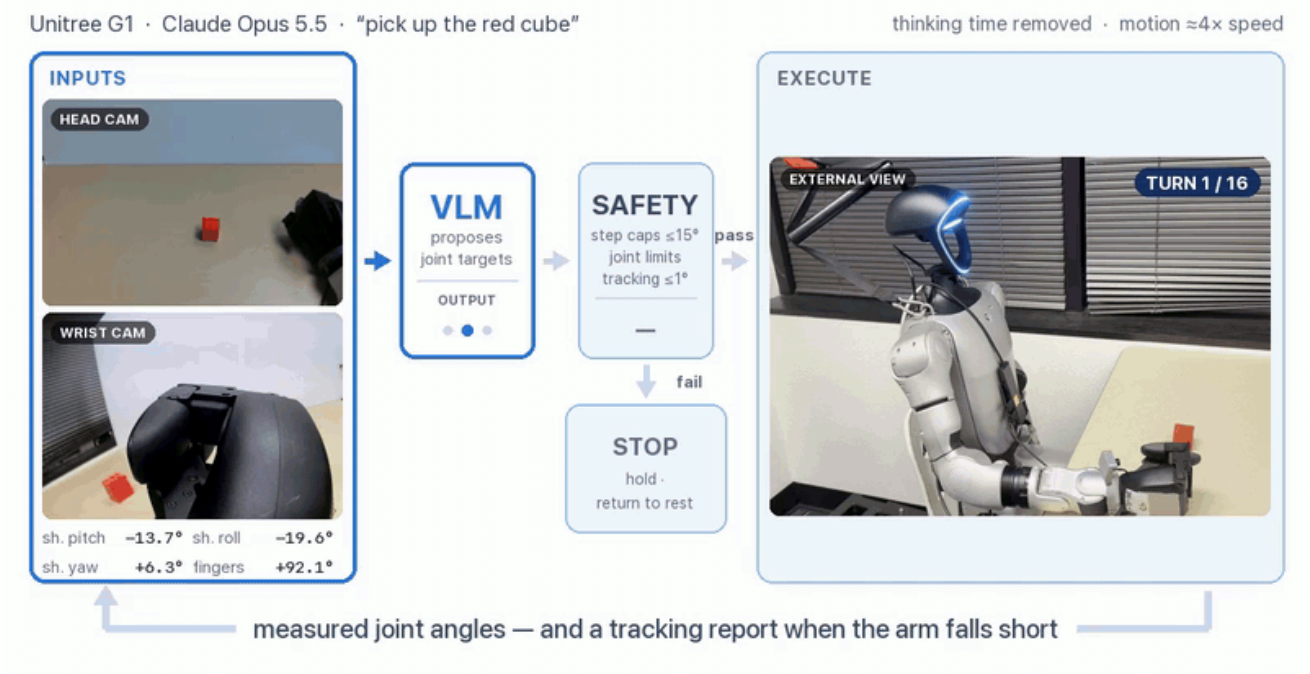

Figure 3: *Claude Opus 5.5 sets joint targets for a Unitree G1 using head and wrist images alongside measured joint states. Targets pass a $\le 15^\circ$ per-joint, per-turn limit check before 50 Hz tracking, and the actual motion and tracking errors feed into the next prompt. Thinking time is removed and motion is shown at roughly $3\times$ speed.*

2. Controlling what the robot remembers

The second design choice is how that evidence enters the model's next input. Memory becomes important as observations and reports accumulate across turns. What we retain, summarize, or discard determines whether the model can distinguish an unsuccessful action from one that was never executed.

When the safety filter blocks a move, or when a grasp slips, what gets stored in the history $o_{0:k}$ and $f_{0:k}$ shapes every decision that follows. If the robot's memory flattens these different feedback signals and summarizes a safety-blocked turn simply as a failed physical grasp, the model will draw a false conclusion about an action that never physically happened. It might spend the next five turns trying different gripper orientations when the real problem was a blocked approach path.

Controlling the chat means treating the context window like a state estimator. We cannot afford to keep every raw camera frame forever, so we have to compress the history without erasing active constraints or turning a guess into a fact. A good feedback log keeps what the model *requested* strictly separate from what the robot *actually executed*. Our Physical Agentic AI framework makes this distinction explicit by having a non-actuating planner propose actions while a deterministic orchestrator validates and authorizes each skill before execution [7]. The log should also separate sensor observations (like reaching a force limit) from hypotheses the model still needs to test (like assuming the object is too heavy).

3. Deciding when to look, when to switch, and when to act

The third design choice concerns the actions sent back to the robot and their timing. The next action may advance the task or gather evidence needed to choose a better action. Execution also requires deciding whether a proposal is still valid and how to transition between targets, settings, and operating modes.

Putting control around the chat lets us handle the physical realities that a static prompt ignores. Sometimes the best move in a conversation is to ask a clarifying question. In robotics, that is system identification and active sensing. If a handle is occluded in o_k or an object's weight is unknown, the supervisor can choose an exploratory mode m_k, moving the wrist camera for a better angle or nudging the block to generate contact feedback in f_{k+1} before committing to a grasp.

At the same time, the physical world keeps moving while the model is thinking of its reply. If a human operator steps in mid-task and warns the robot to keep a cup upright because it is full, or if the object shifts while the model is computing, the supervisor has to absorb that new spoken or visual feedback and update its parameters θ_k and modes m_k without causing jerky transients in the low-level controller. Timestamps on observations and validity windows on proposals ensure the fast controller knows what to do while waiting for the next turn of the chat.

Why this perspective matters

When we watch demos of frontier models controlling robots, a single clean grasp only shows that the model made a good initial guess. The real test of a feedback loop is what happens right after a mistake.

These questions also change how we evaluate the system. Alongside task success, we can measure recovery after disturbances, repeated requests for blocked actions, sensitivity to missing or incorrect feedback, and performance under delayed observations. Holding the model fixed while changing the types of feedback we pass back, how we structure them in memory, or how we time their execution helps reveal how much reliable behavior comes from the design of the conversation itself.

Viewing robot control as a chat gives us a clean way to bridge frontier multimodal models with the guarantees of control theory across very different robot embodiments, whether guiding a tabletop manipulator, a humanoid (Figure 3), or a quadruped coordinating its body and feet to walk toward a target (Figure 4). The language model brings visual reasoning, diagnosis, and flexible replanning, while the surrounding control architecture governs what enters the context window, how modes switch, and which commands are safe to execute. Plus, the chat log itself gives us built-in explainability, letting us trace every recovery or failure back to the exact observation, safety report, and command that produced it.

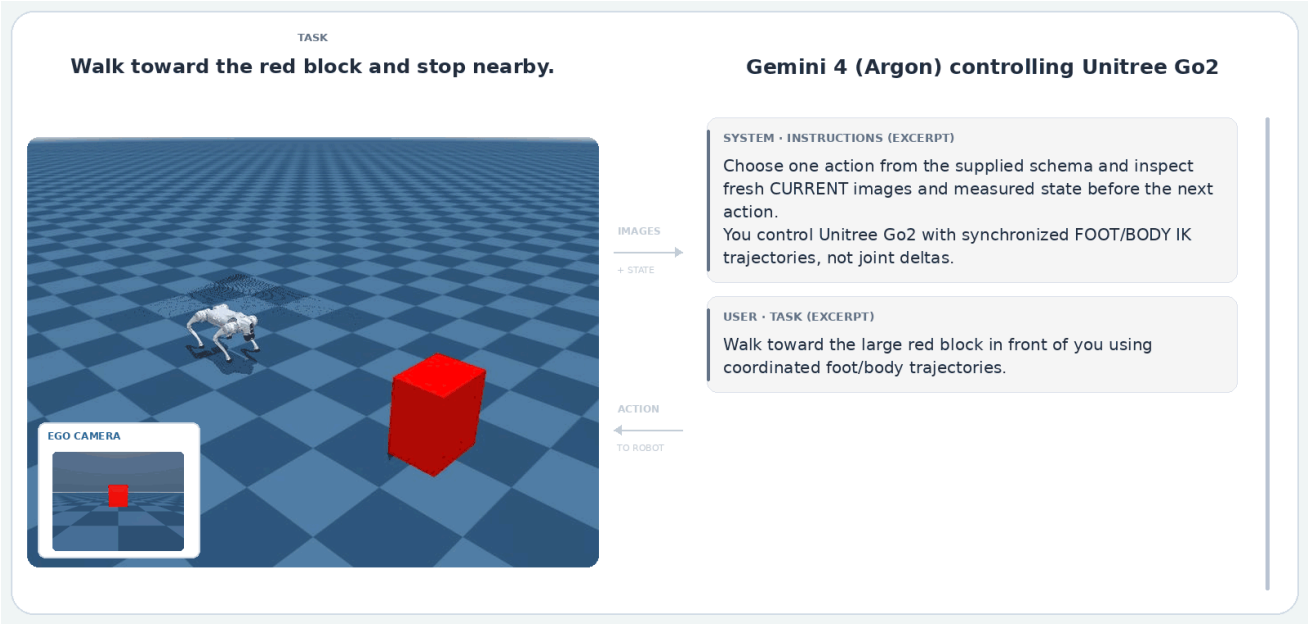

Figure 4: *Gemini guides a Unitree Go2 quadruped toward a red block, coordinating its feet and body while using visual feedback to adjust each movement.*

References

1. Ahn et al. (2022). *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*. Paper and project.
2. Huang et al. (2022). *Inner Monologue: Embodied Reasoning through Planning with Language Models*. Paper and project.
3. Ames, Xu, Grizzle, and Tabuada (2017). *Control Barrier Function Based Quadratic Programs for Safety Critical Systems*. IEEE Transactions on Automatic Control. Paper.
4. Wabersich and Zeilinger (2021). *A predictive safety filter for learning-based control of constrained nonlinear dynamical systems*. Automatica. Paper.
5. Yuksekgonul et al. (2024). *TextGrad: Automatic "Differentiation" via Text*. arXiv preprint. Paper.
6. Espinola, Zhang, and Xiang (2026). *From Reasoning to Action: Benchmarking Frontier AI Models on LIBERO-10 Robot Manipulation Tasks*. arXiv preprint. Paper.
7. Liu, Sadikoglu, Chatterjee, and Senanayake (2026). *Physical Agentic AI: An Architecture for Orchestrating a Robot Crew with LLMs*. Paper.