

Google DeepMind publishes Gemini 4 Argon for robotics

Source copy kept by wheresthe.ai for the story "Google DeepMind publishes Gemini 4 Argon for robotics" (<https://wheresthe.ai/d/intl-google-deepmind-publishes-gemini-4-argon-for-robotics-2026-10/>).

Source: <https://deepmind.google/blog/gemini-4-argon-for-robotics>

Text saved 18 Oct 2026 (the page as published).

SHA-256 of the saved text: feaad7e97080e5554f324bdf7de19d0449aa119b3e7385a1af4ba0f55245f497

Gemini 4 Argon for Robotics

- Google Developers Blog

Gemini 4 Argon for Robotics

OCT. 13, 2026

Kanishka Rao

Principal Software Engineer

Share

Facebook

Twitter

LinkedIn

Mail

Gemini 4 Argon marks a step change in physical understanding, code generation, and visual-semantic reasoning —unlocking new workflows across the robotics stack and accelerating progress toward physical AGI. You can see this leap in capability on the Embodied Reasoning (ERQA) benchmark , which tests a model's ability to understand and solve physical, spatial problems in 3D environments. Using pure reasoning, Argon scores 87%, outperforming every other frontier model.

As a general-purpose physical reasoning model, Argon can be applied across the entire robotics lifecycle. In this post, we share representative prompts for Argon across four key areas:

Robot control → Mobile manipulation with MuJoCo practice, zero-shot robot control, and closed-loop steering of video world models. Real2Sim → Reconstructing physically calibrated simulation twins from robot camera feeds, egocentric human video, or hand-drawn paper sketches. Physical understanding → Agentic multimodal hindsight labeling including detailed understanding of dexterity, dense value functions and progress estimation. Research and self-improvement → Action improvement via simulation and iterative self-improvement on hardware.

Unlocking robot control with Argon 1. Spot mobile manipulation: Putting in the trash bin with MuJoCo practice Argon controls a Boston Dynamics Spot® quadruped (12-DoF legs + 6-DoF arm) to learn a new skill: stepping on a foot-pedal trash bin to throw away soda cans. Argon creates an articulated MuJoCo digital twin of a foot-pedal trash bin to practice its moves before stepping on the physical pedal and dropping soda cans into the open bin.

Sorry, your browser doesn't support playback for this video

Spot Mobile Manipulation — Real-to-Sim-to-Real foot-pedal trash bin disposal: Spot picking up the soda can, checking its wrist camera, practicing 3-leg tripod balance in MuJoCo, stepping on the foot pedal, verifying hand centering in the open bin, and dropping the can inside.

Prompt:

There's a Diet Coke can behind you and another on the black table on your left. Use the spot-mobile-manipulation skill to pick them up and trash them in the black foot-pedal bin. Before stepping down on the pedal, verify in the front cameras that your front-right foot target is centered on the pedal, and use a 2-agent verification swarm (foot-verifier and hand-verifier) to verify both the foot-pedal step and the 3D hand-in-bin alignment before releasing.

Plain text

Copied

2. Zero-shot control Argon controls a simple robotic arm to place a fallen mug upright and put the crumpled napkin inside. Argon reasons with a visual coordinate harness and action primitives to perform this task over several turns.

Sorry, your browser doesn't support playback for this video

Left: 3rd-person real robot view; Right: calibrated wrist camera with projected 2 cm world coordinates and jaw-closing reticle.

Prompt:

[Attached: wrist_cam.png, obs.json]

Task: Get the robot arm to reorient the mug upright and put the napkin inside the mug.

Execution Discipline - Simply output 1 sentence observation followed immediately by your `curl` command (with `# Plan:` on Turn 1 and `# Thought:` on every turn). Everything needed is in `obs.json` and the calibrated reticle/grid drawn on `wrist\_cam.png`.

Visual Coordinate Reference (Annotated on wrist_cam.png) - `+X` (UP in image): Away from robot base | `-X` (DOWN): Towards robot base - `+Y` (LEFT in image): World left | `-Y` (RIGHT): World right - `+Z`: Height above table | `Yellow numbers`: World (X, Y) in centimetres (2 cm grid at `grid\_z`) - `Cyan + crosshair`: Current 3D pinch center - `Cyan [ + ]` brackets: Current jaw opening and closing line - `Green line`: Current object long-axis orientation (`yaw\_deg`: 0° = vertical along +X, 90° = horizontal along +Y) - `pitch\_deg = 90°`: Straight down (-Z) top-down grasp/place; `pitch\_deg < 60°`: Tilted coaxial approach along finger axis

Closed-Loop Turn Workflow 1. Inspect current state & view `wrist\_cam.png`: `curl -s http://127.0.0.1:8081/obs` (returns `ee\_pos`, `holding`, `status`, `workspace`, `wrist\_cam\_path`). 2. Select VLM call granularity upfront on Turn 1: - Coarse (2–4 calls): `look\_at` -> `grasp` -> `place` -> `done` - Pre-Grasp Hover Verify (4–7 calls): `look\_at` -> `goto(z=0.04)` (verify cyan `[ + ]` brackets straddle target) -> `grasp` -> `place\_until\_contact` -> `done` - Fine Visual Servoing (8–20 calls): `look\_at` -> `grasp` -> pre-pose `goto` -> closed-loop `move\_by` / `move\_until\_contact` -> `done` 3. Send motion primitives (`POST /plan`) and verify visually after every turn in the updated `wrist\_cam.png`. If `status == "grasp\_failed"`, call `robot.look\_at(...)` to re-observe and recover closed-loop.

Code-as-Policies Primitives (`POST http://127.0.0.1:8081/plan`) robot.look_at(x=0.35, y=0.13, z=0.14, grid_z=0.015) x, y = robot.deproject(u=0.52, v=0.48, z_plane=0.015) robot.grasp(x=x, y=y, z=0.015, yaw_deg=-35, pitch_deg=90) robot.place(x=0.24, y=0.06, z=0.075) robot.place_until_contact(x=0.24, y=0.06, z=0.015, max_lag=0.012, relief=0.002) robot.goto(x=0.28, y=0.05, z=0.04, arc_z=0.14, pitch_deg=25, azimuth_deg=90, roll_deg=0, speed=0.10) robot.move_by(forward=0.025, dz=-0.002, dpitch_deg=-3, droll_deg=5, speed=0.04) robot.move_until_contact(dz=-0.06, speed=0.03, max_lag=0.012, relief=0.002) robot.open_gripper() / robot.close_gripper() / robot.home() / robot.done()

Plain text

Copied

3. Omni control Argon steers robots in the Gemini Omni World Model to rearrange five scattered Scrabble tiles (G, R, A, N, O) into a clean row spelling A - R - G - O - N . Instead of specifying one static prompt, Argon steers in a closed visual loop. It writes timestamped action prompts, inspecting intermediate rollout keyframes, and refining its prompts.

Sorry, your browser doesn't support playback for this video

Iterative Video World Model Steering Dexterous Hands: 33-Step Steered Video Rollout Spelling A-R-G-O-N.

Prompt:

Goal: {task_goal}

You are steering a video world model (Gemini Omni) in an iterative visual feedback loop. Inspect the attached

frames from the latest rollout and write the next timestamped action prompt for the video world model: - Break complex motions into one simple, single-direction action at a time. - Keep inactive hands retracted and avoid blocking the camera's view of objects in the workspace. - Explicitly state which areas of the workspace become empty after an object moves. - Use [MM:SS.mmm] timestamps for each action beat and wrap your output in <omni_prompt> tags.

Example format (sample timestamps only, not the full prompt): <omni_prompt> [00:00.000] Character A's left hand rests at the side while the right hand reaches toward the target object. [00:02.500] Character A's right fingertips push the object smoothly into the center of the table, leaving its starting area empty. [00:05.000] Character A's right hand opens and retracts to the side so all placed objects remain in clear view of the camera. </omni_prompt>

Plain text

Copied

Unlocking Real2Sim with Argon 1. Robot view to simulation Argon views a real robot teleoperation episode and develops a digital twin; it obtains or creates simulation assets that look and behave like the real objects upon contact with the dexterous hands. Argon tunes the assets and scenes to match sim visuals and contact events with those in the real recorded data in an iterative visual-reasoning loop.

Sorry, your browser doesn't support playback for this video

Left (Real View): synchronized real robot camera feeds; Right (Simulated): 3D MuJoCo scene reconstructed and calibrated by Argon.

Prompt:

[Attached: robot_rollout_episode]

Build a MuJoCo scene of the robot and reconstructed task objects. When each episode is replayed frame by frame, by driving the robot joints toward the logged observation.state with joint-space PD.

Iterate using two verification objectives: (1) Visual match: MuJoCo renders from the head_left, left_wrist, and right_wrist cameras line up with, and look like, those real observations. (2) Physical match: the simulated contact force/torque on each Sharpa fingertip, in the sensor frame, matches the logged tactile_force in timing, magnitude and direction.

What you can tune happens before simulation starts: (1) Static object assets: realistic geometry, collision shapes, mass, inertia, friction and joints, with appearance close to the real objects, plus the initial object poses at $t = 0$. (2) Offline-planned robot commands, changed only in frames where the real fingertips are near an object. Each command must stay within 0.55 rad (arm joints) and 0.65 rad (hand joints) of the logged teleop trajectory. Everywhere else, replay the logged trajectory unchanged.

No sim cheating: task objects may move only through real MuJoCo contact with the robot hands and the environment (mj_step). Not allowed after $t = 0$: changing MjModel parameters; applying external forces or torques to objects; welds or equality constraints; feedforward torques beyond joint-space PD; setting or rolling back object qpos/qvel; disabling or excluding collisions; reporting contact forces that don't come from the MuJoCo solver (for example, copying the logged tactile readings).

Immediately present the reconstructed scene and rollout results as an interactive HTML session viewer with one page per episode, showing: - At the bottom: A synced timeline (play, scrub, speed control) that drives every panel. - On the left panel: An interactive 3D Blender scene showing the simulated robot and objects, showing contact points and force arrows as replay progresses. - On the right panel: Real vs. sim camera views side by side for all 3 cameras. Real vs. sim finger force visualizations.

Plain text

Copied

2. Human view to simulation Given an egocentric human video of a manipulation task, Argon identifies the task-relevant objects, replaces crude bounding boxes with compound convex-decomposed meshes for concave containers, and retargets the human hand trajectory onto a 76-DoF bimanual humanoid with 22-DoF five-fingered hands in MuJoCo contact physics.

Sorry, your browser doesn't support playback for this video

Human View to Simulation — Egocentric Human Video to Dexterous Humanoid Twin: Left shows the input egocentric human RGB video; Right shows the synchronized 500 Hz MuJoCo + Blender Cycles multi-view simulation rollout.

Prompt:

[Attached: egocentric_human_video.mp4, candidate_sim_views.png, hand_3d_keypoints.json]

You are building a physically solvable 500 Hz MuJoCo manipulation environment (`scene.xml`) and studio Blender render that faithfully reproduces the tabletop task shown in `egocentric\_human\_video.mp4` on a bimanual humanoid robot with 22-DoF dexterous five-fingered hands:

1. Scene Reconstruction & Camera Calibration: - Identify only the manipulation-relevant objects (the grasped target object and the receiving container or landmark) and triangulate their 3D tabletop coordinates from the egocentric camera intrinsics/extrinsics. - Replace crude bounding boxes with compound convex-decomposed 3D meshes so concave containers (nested bowls, recessed wooden trays, open bins) support physical interior placement.
2. 500 Hz Dexterous Motion-Fitting Stress Test: - Retarget the 3D human wrist and finger keypoints to the humanoid's 7-DoF arm and 22-DoF hand joints, and step the trajectory in MuJoCo at 500 Hz ($dt = 0.002s$) with zero kinematic weld constraints. - Verify opposing thumb-and-finger force closure, clean object lift ($> 2\text{ cm}$), and stable final placement inside or beside the target landmark.

Plain text

Copied

3. Sketch to simulation With Argon, a developer can snap a photo of a rough pen-and-paper sketch—showing an Apollo 2 humanoid throwing a ball along a parabolic arc to topple a 3-2-1 stack of boxes on a table—and have Argon automatically construct the GPU-accelerated mujoco_warp scene, synthesize a balanced 76-DoF whole-body and five-finger throw trajectory, verify 6/6 target knockdown in closed-loop physics, and export a Blender 4.5 Cycles render.

Sorry, your browser doesn't support playback for this video

Sketch to Simulation — Hand-Drawn Sketch to Dexterous Humanoid Trajectory & Cycles Render.

Prompt:

[Attached: hand_drawn_throw_sketch.jpg]

Translate this hand-drawn sketch into a physically verified humanoid simulation in `mujoco\_warp` and a raytraced Blender Cycles video:

1. Scene & Kinematic Setup from Sketch: - Parse the spatial layout from `hand\_drawn\_throw\_sketch.jpg`: place the 76-DoF humanoid (32-DoF body equipped with two 22-DoF five-finger dexterous hands) at the origin facing a table holding a 3-2-1 pyramid of 6 rigid target boxes, with a projectile sphere grasped in the right five-finger hand.
2. Whole-Body Balance & Dexterous Throw Synthesis: - Synthesize a 50 Hz whole-body throwing trajectory (140 steps, 2.80 s) across four phases: (1) stable stance & 5-finger power grasp closure, (2) torso/shoulder wind-up with ankle-hip Center-of-Mass compensation, (3) forward whipping acceleration, and (4) synchronized 22-DoF five-finger opening at peak hand velocity so the ball launches on a ballistic arc into the pyramid's second-tier center of mass. - Verify in `mujoco\_warp` contact physics that the humanoid maintains upright balance ($< 3\text{ cm}$ pelvis drift) and knocks down all 6 target boxes.

Plain text

Copied

Unlocking physical understanding with Argon 1. Hindsight labeling and data quality Argon has precise physical understanding of dexterous manipulation and can provide detailed segmentation and text labeling from video. Argon is able to identify poor manipulation segments such as pauses, unstable grasps and fumbles. Given an overhead video of a bimanual ALOHA 2 robot folding a square sheet of paper into an origami fox over 136

seconds, Argon produces a time-aligned 4-level hindsight annotation hierarchy: Episode Goal (G1), 6 contiguous Semantic Subtasks (G2), 17 contact-level Atomic Primitives (G3), and automated Quality Control (QC) reweighting that holds task progress flat during fumbles and retries.

Sorry, your browser doesn't support playback for this video

Argon Multi-Granularity Hindsight & Telemetry HUD.

Prompt:

[Attached: Aloha2_Origami_20250309.mp4]

Produce a time-aligned multi-granularity hindsight annotation JSON with four levels: 1. G1_episode_instruction: Concise imperative sentence describing the overall manipulation goal. 2. G2_subtasks: Contiguous semantic subtask phases specifying which arm(s) anchor vs. fold/rotate the object. 3. G3_atomic_primitives: Fine-grained contact-level primitives (3–15s) aligned with grasp closure, crease sweeps, release, or retry attempts. 4. quality_control_and_reweighting: Hold task_progress_pct flat during non-progressing slip/delamination retry loops and downweight them (0.15x–0.25x) for Behavior Cloning.

Plain text

Copied

2. Find the odd ball To recreate the classic 12-ball balance scale puzzle with robots, Argon builds a bimanual Franka FR3 MuJoCo environment with 12 visually identical pool balls (one secretly heavier or lighter) and an articulated Roberval two-pan balance scale. It then writes a vision-only closed-loop solver that triangulates the balls from stereo RGB cameras, visually reads the scale tilt after each weighing, and isolates the odd ball into the Heavy (Red) or Light (Blue) tray within 3 weighings.

Sorry, your browser doesn't support playback for this video

Find the Odd Ball (Vision-Only Closed-Loop Solver): Dual-arm Franka FR3 weighing candidate balls on the Gold/Teal pans, reading stereo scale tilt, and isolating the confirmed Heavy odd ball into the Red tray in 3 weighings.

Prompt:

Prompt 1 — Physics-Accurate MuJoCo Environment & Benchmark Generation Make a MuJoCo environment where a dual-arm robot (two 7-DoF Franka FR3 arms with parallel-jaw grippers on a shared tabletop) is tasked to solve the classic 12-ball balance scale weight puzzle from https://www.mathsisfun.com/pool_balls_solution.html. 1. Create 12 visually indistinguishable pool balls in a 2x6 starting rack (11 standard 150 g, 1 odd ball either 270 g or 30 g). 2. Build a functional two-pan Roberval balance scale at the center of the table with parallelogram equality constraints (`mjEQ\_JOINT`) so the Left (Gold) and Right (Teal) pans stay horizontal, plus Red (Heavy) and Blue (Light) submission trays. 3. Track distinct settled weighings (`weighing\_count`) and reward `1.00` for isolating the true odd ball in the correct Heavy/Light tray within ≤ 3 weighings.

Prompt 2 — Closed-Loop Vision-Only Solver Implement an independent closed-loop policy relying strictly on stereo RGB camera inputs (`head\_camera\_left\_rgb\_image`, `head\_camera\_right\_rgb\_image`) and robot proprioception (`add\_privileged\_task\_observations=False`): 1. Segment the 12 balls, Gold/Teal pans, and Red/Blue trays in both stereo views and triangulate their 3D coordinates. 2. Visually read the balance scale tilt after each weighing from stereo vertical pixel disparity (+1 = left heavier, -1 = right heavier, 0 = balanced). 3. Rotate the gripper wrist yaw by 90 degrees when grasping/placing inside the narrow scale pans to clear the raised rims, and isolate the odd ball into the Red or Blue tray within ≤ 3 weighings.

Plain text

Copied

3. Value function Argon is a general value function for robotics tasks. Evaluated on a 60-second bimanual ring-sorting episode from a human UMI dataset, Argon iteratively inspects synchronized head, left-gripper, and right-gripper camera feeds around contact transitions to generate dense value function / task progress predictions and decompose the trajectory into 28 segments with positive, neutral, and negative progress slopes.

Sorry, your browser doesn't support playback for this video

Value Function: Synchronized 3-camera UMI playback (head, left gripper, right gripper) locked to animated task progress curve and bimanual segment annotations.

Prompt:

[Attached: umi_head.mp4, umi_left_gripper.mp4, umi_right_gripper.mp4]

Look at 3 videos of human wearing a gripper performing activities from head, left, right gripper views. Look at the videos thoroughly and give a detailed plot of progress at the task (with positive, neutral, negative slope), identifying important segments in the video and labeling the activities. Look at individual frames or subsegments of the video if necessary to get a better understanding.

Plain text

Copied

Unlocking research and self-improvement with Argon 1. Action improvement via simulation Argon can generate new data in simulation to improve performance of generalist vision-language-action (VLA) models. Starting from a single RGB photo of the real tabletop, Argon autonomously builds a calibrated MuJoCo simulation twin, defines new tasks, and synthesizes a closed-loop motion generator to collect 4,500+ domain-randomized episodes in simulation—boosting real-world physical robot Success Rate from 56% to 75% and reducing completion time from 35s to 24s .

Sorry, your browser doesn't support playback for this video

Action Improvement via Simulation: improves the real-world evaluation benchmark (56% → 75% Success Rate and 35s → 24s Completion Time).

Prompt:

[Attached: real_tabletop_photo.jpg]

You are given a single overhead RGB camera photo (`real\_tabletop\_photo.jpg`) of a bimanual robot workspace (two 7-DoF Franka FR3 arms with parallel-jaw grippers and 4 RGB cameras) facing a table arranged with photobooth party props. 1. Reconstruct procedural 3D meshes and compound MuJoCo collision primitives for all 9 party props matching their real-world dimensions, colors, and resting poses, including spring-hinged deformable crown panels for the soft plush cap. 2. Define a 77-task single-arm and bimanual curriculum (pick-and-place, uprighting, directional pushing, mid-air handovers, bimanual scoop-and-push, and pipelined table clearing). 3. Synthesize a closed-loop IK trajectory generator across ± 10 cm pose randomization, verify `reward == 1.0` in MuJoCo physics, and export 4,500+ episodes for VLA fine-tuning.

Plain text

Copied

2. Self-Improvement for visual pick and place On a simple 6-DoF arm, Argon iteratively self-improves both execution speed and placement accuracy on a pick and place task via trial and self-reflection.

Sorry, your browser doesn't support playback for this video

SO-101 Vision-Guided Pick-and-Place — Iterative Hardware Self-Improvement Across Iterations 1 → 2 → 3: Iteration 1 (58.9s), Iteration 2 (22.9s, grazed box rim on drop), and Iteration 3 (17.4s, dead-center drop).

Prompt:

Stage 1 — Measure Objects & Dry-Run Plan (No Hardware Motion) Stage 1: measure objects — find the objects and plan the whole motion. The arm does not move in this stage.

A. I put a LEGO brick and a box on the table in front of the arm, and the wooden box to one side. Find the brick and the wooden box in the image. And also in the real world. Show this in an annotated image.

B. A dry-run plan (no hardware). Write the planner to pick and place the lego block. Print the plan table, draw all poses onto the overhead image, and stop. Do not move the robot. Report with evidence.

Stage 2 — Pre-Pick Hover & Wrist-Camera Verification Stage 2: Put the gripper above the brick without touching it, and verify with the wrist camera that the arm and the vision agree. Please record executions with both

cameras (compressed video is OK).

Stage 3 (Autonomous Pick-and-Place → Self-Improvement → Optimization): 1. Execute autonomously: Turn the Stage 1 dry run into a real grasp, lift, and place into the wooden box. Run it yourself subject to safety rules from AGENTS.md.

2. Self-improve: When an attempt fails, diagnose it yourself. Review your own logs and wrist/overhead footage, find the root cause, check the fix in a dry run, and retry.

3. Optimize on feedback: Once it succeeds, make it at least 2× faster within safety limits (rule R5) unless I authorize more, or give feedback.

Plain text

Copied

3. One-prompt robotics paper reproductions in MuJoCo WebAssembly With a single prompt per paper, Argon reads a robotics control manuscript and any released checkpoints, ports the physics model and closed-loop controller (from 7-network hierarchical locomotion in DeepLoco and 50 Hz PPO actors in HUSKY to Web Worker iLQR in Whole-Body MPC) into self-contained MuJoCo WebAssembly browser labs, and verifies them in headless Chromium.

Sorry, your browser doesn't support playback for this video

Five Landmark Legged & Humanoid Papers Running Live in Browser (MuJoCo WASM): Robotic Soccer OP3 1v1, DeepLoco, Massively Parallel Deep RL ANYmal C, Whole-Body MPC Go2, and HUSKY G1 Skateboarding (from 68 interactive browser sandboxes reproduced by Argon).

Prompt:

[Attached: paper.pdf, optional author_checkpoints/ or urdf_assets/]

Reproduce the paper "<Paper Title & arXiv ID>" from scratch in MuJoCo WebAssembly and deploy a self-contained browser lab + visual essay: 1. Match the paper's exact physics timestep, actuator gains (kp/kv), torque limits, and kinematic constraints in MuJoCo XML, and run real neural inference or Web Worker iLQR in JavaScript at the paper's exact control rates. 2. Build `index.html` (Spatial UI simulator with camera presets, live telemetry plots, and disturbance injection) and `scrollytelling.html` (interactive visual essay). 3. Run a headless Chromium (Playwright) verification loop asserting 0 console errors and non-zero task progress over 300+ frames.

Plain text

Copied

Advancing embodied intelligence with Argon Argon bridges coding, visual perception, physics simulation, and embodied robot control. By reasoning natively over multi-view video, 3D geometry, contact mechanics, and proprioceptive telemetry, Argon understands not just what objects are in a scene, but how they physically interact with agents to accomplish whole-body dexterous tasks. Gemini 4 Argon marks an important milestone accelerating the path toward physical AGI. By building this core intelligence, our goal is to enable robotics that can work alongside humans to solve complex challenges. Getting started You can start building with Gemini 4 Argon today across Google's developer platforms:

Google AI Studio : Available to Google AI Ultra subscribers. Gemini API: Available with the paid tier of the Gemini API in AI Studio. Use gemini-4-argon in the Interactions API with day-one support across the google-genai SDKs. Google Antigravity: Available for Google AI Ultra subscribers. To start building, select Gemini 4 Argon (gemini-4-argon) as your active model in Google Antigravity for complex planning, /goal runs, and multi-agent orchestration.

posted in:

AI

Case Studies

Announcements

Learn

Explore

Solve

Previous

Next

Related Posts

AI

Cloud

Announcements

Solutions

Three layers, one repo: a tour of google/adk-recipes

OCT. 15, 2026

AI

Cloud

Best Practices

Industry Trends

Evaluating AI agents: moving past the vibe check

OCT. 15, 2026

AI

Cloud

Tutorials

Case Studies

Accelerating Spatio-Temporal Attention for Video Diffusion on TPUs

SEPT. 30, 2026

Mobile

Web

Case Studies

Announcements

ML Drift: Next-Gen GPU AI/ML Inference at the Edge

OCT. 8, 2026