

Kyle Vedder's essays explained

Notes from a chat with Claude, October 3–4, 2026.

Question: Explain Kyle Vedder's essays "The Case Against Human Hands", "State of Robot Learning (December 2025)" and "Cargo Cults, Data Flywheels, and Novelty Pumps", plus his X thread on selling post-training data, from first principles, with examples and every robotics concept defined.

Kyle's four pieces make one argument:

1. Robots today learn almost entirely by copying human demonstrations, so progress depends on demonstration data.
2. What makes that data valuable is **novelty**: situations and strategies the current policy hasn't seen. Producing novelty takes continuous operational work; deploying robots doesn't produce it automatically.
3. Hardware should play to a machine's strengths, such as big replaceable motors and cheap cameras, instead of copying the human hand.
4. The business value is in data that provably makes a policy reliable, not in bulk data.

I'll go through them in the order that builds the concepts: state of robot learning, then novelty, then hands, then the thread. Then I cover where he's strong, where to push back, and what it means for the options you're weighing.

1. "State of Robot Learning" (Dec 2025)

Behavior cloning

A **policy** (π) is the robot's brain: a function from what it senses to what it does.

- **State (s)** is what the robot senses:
 - camera images, usually a scene camera plus one on each wrist;
 - joint angles, called **proprioception** (the body's sense of where its limbs are);
 - often a text instruction.
- **Action (a)** is what it commands: target joint angles, or a target pose for the **end effector** (the gripper or hand at the end of the arm).
- **Behavior cloning (BC)**, also called imitation learning, is plain supervised learning. You record a human doing the task, pair each state with the action the human took, and train the network to predict that action. It's like training an image classifier where the label is "what the human did next."

Two details he mentions:

- **50 Hz control**: the motor controllers need a new target every 20 ms.
- **Action chunks**: the policy predicts the next ~50 actions (one second) at once, for three reasons:
 - Fewer decisions mean fewer chances to go wrong: a one-minute task becomes 60 decisions instead of 3,000.

- The robot commits to one coherent motion instead of dithering.
- Big models can't run every 20 ms.

BC's ceiling is the data. It copies the demonstrator's choices, speed and mistakes.

Where demonstrations come from

Leader–follower teleop (ALOHA, GELLO). The operator moves a "leader" device, and its joint angles are streamed to the "follower" robot as targets. ALOHA uses real arms as leaders (about \$20k per two-arm station). GELLO uses a 3D-printed scaled replica with cheap servos as joint sensors (under \$300).

- **Why it's good:** the data is recorded on the robot itself, so the cameras, joint readings and action format match deployment exactly. Every motion was physically executed, so it is **kinodynamically feasible**. It stays within the robot's reach and joint limits (kinematics) and its speed and torque limits (dynamics).
- **Why it's up to 10× slower:**
 - The operator feels no contact forces.
 - They watch through the robot's cameras, or from an awkward angle.
 - They fight lag and the robot's limits.
- **Why operators need weeks of practice:** they have to judge depth and contact without touch. Hesitant, jerky demos produce hesitant, jerky policies.
- **Why it's capital-heavy:** every station needs a whole robot.

Handheld devices (UMI). A person holds a copy of the robot's gripper with a camera where the robot's wrist camera would be, and does the task by hand.

- **SLAM** (simultaneous localization and mapping: recovering the camera's 3D path from video and an inertial sensor) reconstructs where the gripper went.
- **Inverse kinematics (IK)**, which solves for the joint angles that put the gripper at a given pose, converts that path into robot joint angles.
- **Why it's good:** no robot is needed, so collection is cheap, runs at natural speed and works anywhere. Generalist and Sunday built their data strategies on devices like this.
- **Why the data is noisy:**
 - SLAM drifts, so the recorded actions are noisy.
 - Joint angles are computed, not measured, and IK may have several solutions or none.
 - Training images show a human arm, but at deployment the robot sees a robot arm, so it is always slightly outside its training data.
 - Humans reach and twist in ways the robot can't, so some demos are impossible to execute.

Plain human video (YouTube, or head cameras on workers). It is the largest, cheapest and most diverse source, at full speed. But video contains no actions:

- They must be inferred by hand and body trackers, which gives approximate **pseudo-labels**.
- The viewpoint often differs from the robot's.
- Humans use their whole body (leaning, shifting weight) in ways the robot can't.

Human video has been the fastest-moving source since he wrote this. NVIDIA's EgoScale pretrained on 20,854 hours of it with steady gains, but still needed a small aligned set (about 50 h of human and 4 h of robot data, with matched cameras and gloves) to bridge to the robot. So the gap he describes is real, and it is also bridgeable.

The hard problem: drifting out of distribution

Out of distribution (OOD) means a state unlike anything in training. There are three causes:

1. **The world never matches training.** Lighting, backgrounds and object positions shift, and networks latch onto irrelevant cues. Starting from a **VLM** (vision-language model pretrained on internet images and text) helps the policy recognize unfamiliar scenes. It does not teach the policy what to do in states it never practiced.
2. **Ambiguity.**
 - **Partial observability:** the policy can't see what matters, such as the folds inside a crumpled shirt.
 - **Multimodality:** demonstrators disagree. If half pass an obstacle on the left and half on the right, a model minimizing average error predicts the average and drives straight into it. That's why modern policies such as π_0 use generative action heads (diffusion or flow matching) that pick one option instead of averaging.
3. **Compounding error.** Each action shapes the next state. A small error puts the robot slightly off the demonstrated path, where the policy is less accurate, so the next error is bigger. Ross and Bagnell showed BC's expected mistakes grow with the *square* of task length.

A worked example. A one-minute task is 60 chunks.

- If each chunk has a 1% chance of an unrecoverable mistake, success is $0.99^{60} \approx 55\%$.
- If recovery training cuts that to 0.1%, success is $0.999^{60} \approx 94\%$.

Reliability comes mostly from recovery, not initial accuracy. His point is that the training data's coverage matters more than the model architecture: you can't model your way into states you've never seen.

Dagger: teaching recovery

Dagger (Dataset Aggregation) works in rounds:

1. Run the current policy and let it reach its own mistakes.
2. Have an expert label the correct action in those states.
3. Add that data, retrain and repeat.

Because it trains on the states the policy actually visits, error growth becomes linear instead of quadratic. On real robots, an operator watches and takes over by teleop when failure looms. Those takeover segments are the recovery data.

His curation warning: BC imitates *everything* in the data. If a clip includes the slide into the bad state, the policy learns to go there. Keep the way out, not the way in.

Why it's an art:

- Each round means watching failures, judging which matter and designing data to fix them. His "data sommelier" is the person with the taste to do this.

- You develop a feel for one policy's quirks. Then a new **base policy** (the large pretrained model you fine-tune; the fine-tuning is **post-training**) fails in different places, and much of that feel resets.

Better policies are harder to evaluate

Each trial is pass or fail, so you need enough failures to tell two policies apart:

- 80% vs 90%: about 200 trials each.
- 99% vs 99.5%: about 4,700 trials each.
- A policy that fails every 15 s shows a difference in minutes. One that fails every 2 hours needs about 20 robot-hours per policy to collect 10 failures each.

Offline metrics, such as validation error on held-out demos (the kind of scaling curve in Generalist's blog), predict real success poorly, for three reasons:

- They test only on states from human demos, never the states the policy's own mistakes create.
- They average thousands of easy steps, while success hinges on a few critical moments, like the instant of a grasp.
- They score a valid alternative motion as an error.

Speed: BC can't beat the demonstrator

- **Keep only the fastest demos:** you discard most of the data and lose robustness.
- **Condition on speed** (Eric Jang's "Just Ask for Generalization"): tag demos by speed, then ask for "fast" at run time. This works only within the demonstrated range and can't exceed the fastest human.
- **Play actions faster** (a 50 Hz plan executed at 70 Hz): the motors must track sharper accelerations, and physics doesn't speed up. A flicked shirt still takes its time to settle.

Beating humans requires optimizing the outcome rather than copying it, which means reinforcement learning.

Why RL works for LLMs but not yet for robots

Reinforcement learning (RL): try actions, get a reward (often **sparse**: just success or failure at the end), and shift toward what worked.

LLMs have two luxuries:

- **The same state can be reproduced endlessly.** Sample 16 answers to one math problem, check which are right, and reinforce those. GRPO uses the group's average as the baseline. This is online, on-policy RL, meaning you learn from fresh attempts by the current model, and the comparison between choices is *observed*.
- **A strong base model.** It already succeeds sometimes, so there is a signal to amplify.

Robots have neither:

- You can't recreate the same messy kitchen 16 times.
- Long tasks may never succeed by chance.
- Rollouts are slow, and mistakes break things.

What's missing is the **counterfactual**, "what if I'd acted differently here?" LLMs observe it; robots must estimate it.

Two routes around that

RL in simulation. Resets are free, but the simulator isn't reality.

- **Where sims fall short:**
 - Rigid contact is nearly discontinuous (impacts, stick-slip friction), so simulators approximate it and errors pile up as contacts multiply.
 - Cloth, cables and food need slower, different physics models.
 - Rendered images don't look like real camera images.
- **Domain randomization** varies friction, mass, motor strength, delays, lighting and textures, so reality looks like one more variation.
- **Scan dots** are terrain heights sampled on a grid around a walking robot. They look identical in sim and real, unlike raw pixels.
- **RMA (Rapid Motor Adaptation)** trains with the true friction and payload known, then learns a module that infers them from the robot's recent motion, so the real robot adapts on the fly.

Locomotion works because it is mostly proprioception, with simple foot-ground contact and easy-to-write rewards. Manipulation needs precise contact, deformable objects and vision. Today whole-body controllers (like NVIDIA's SONIC) are routinely trained in simulation; dexterous manipulation mostly isn't.

World models are learned simulators: given the current state and an action, they predict what happens next. His key insight is that a policy needs *good* actions, while a world model can learn from *any* action. Random flailing, failures and other robots' data all teach physics, and physics is shared across tasks, so the usable data supply is far larger. As of late 2025, though, none modeled contact-rich manipulation well. Once one does, you can train policies inside it ("policy extraction") and evaluate them without a robot.

RL on the real robot: Q, V and advantage

- **V(s):** expected future reward from this state if the policy keeps acting as usual (for example, the chance of finishing).
- **Q(s,a):** the same, but after first taking action a .
- **Advantage, $A = Q - V$:** how much better a was than the policy's typical move.

Each real rollout starts from a different state, so a success might reflect good actions or an easy start. V separates the two. He calls Q and V "world models by another name" because they compress long-horizon consequences into a single number.

They're hard to learn:

- Rewards are sparse and the inputs are images.
- Estimates are built from other estimates, so errors compound.
- Q is overconfident about actions it has never seen tried.

Advantage-weighted regression (AWR) is BC in which each step counts in proportion to its advantage, so good actions are copied more than bad ones. Physical Intelligence's $\pi^{*0.6}$ ("Recap") is a variant:

- It trains a model that estimates progress toward success.
- It labels each action as better or worse than average.
- It trains the policy with that label, then asks for "better" at run time.

PI reported more than double the throughput and about half the failures on its hardest tasks. Kyle reads the gain over plain BC as minor, and it still needed human corrections.

Since then, real-robot RL has become a practical tool for single skills:

- **RL Token** (PI, Apr 2026) needs 15 minutes to 5 hours on the robot and took screw insertion from 20% to 65%.
- **HIL-SERL** reaches near-perfect success on specific tasks in 1–2.5 hours.

Neither is the general self-improvement he wants.

His predictions

1. **VLAs will be replaced by video-model backbones within 2 years.** A VLA (vision-language-action model) is an internet-pretrained vision-language model with an action output bolted on. A video model learns how scenes evolve, which is closer to physics. This is under way: NVIDIA's DreamZero (Feb 2026) built a 14B policy on a video model, and GR00T N2 is previewed as a "world-action" model. Both are slow so far, and neither is dominant yet.
2. **World models will handle open-world interaction within 10 years,** with policies trained inside them.
3. **Game engines will become data generators for world models,** not the simulator itself.
4. **Expert data will still matter** for fine-tuning.
5. **Real robots will still be needed** to reach superhuman performance on a given body, because only reality exposes rare contact errors.

His "picks and shovels" advice

(In a gold rush, tool sellers profit whoever strikes gold; here the gold rush is general-purpose robots, "Embodied AGI".)

- **Data labeling is a commodity.** It is labor arbitrage with no technical moat, and you'd have to out-operate Scale AI.
- **Pretraining data sales are a hustle.** You must prove the data helps the buyer's model, and not all robot data does.
- **Evals must stay in-house.** They are the lab's daily inner loop on its exact robots and unreleased models, so outsourcing adds delay and leaks secrets.
- **There will be no universal data platform.** Self-driving companies had similar sensors and problems, yet each built its own data engine. Robotics is more varied still.
- **The durable bet is better demonstration hardware and software** (GELLO or UMI class) that removes the pain points above, proven by training policies on it. Demonstrations stay the core input, so whoever makes them faster and cleaner sells to every lab.

2. "Cargo Cults, Data Flywheels, and Novelty Pumps"

Cargo cult: after WWII, some Melanesian islanders who had watched Allied planes deliver goods built imitation runways and radio huts from wood and straw, expecting the planes to return. Feynman used the term for copying a thing's form without its mechanism. Kyle's charge is that deployment "flywheels" copy the form of a data loop without the mechanism that makes data valuable.

The pitch: deploy robots, collect data, improve the policy, unlock more deployments, and compound.

His claim: scale only matters for capturing **novelty**, and novelty is relative to what the policy already knows. It comes in two forms:

- demos that visit new states within a task;
- new tasks that show new strategies.

The gridworld. A robot must go from start to goal around obstacles. Two demos label the right move only along their own paths, and everywhere else (his red cells) the policy is guessing. Training longer on those two demos sharpens the paths but provides zero evidence about the red cells.

From first principles: if the model already predicts a demo correctly, the loss is near zero, the gradient is near zero, and the demo changes nothing. Information is whatever the model couldn't already predict. The millionth identical pick teaches nothing.

Two ways to pump novelty:

- **Within a task:** Dagger corrections where the current policy fails. These are valuable but specific to that policy and task, so moving a general model takes many such fixes across many tasks.
- **Across tasks:** diverse tasks, strategies and environments that make the policy pinch, scoop, pull, regrasp and use both arms. This builds a reusable vocabulary of motions: scooping rice helps with scooping beans, and regripping helps with reorienting a tool.

Both demand that ops teams do something new every day. As with alpha in markets, a pocket of novelty is exploited until the model learns it and then it's gone. Novelty doesn't compound.

Why flywheels avoid novelty by construction:

- **Selection effect:** a robot gets deployed only where it already works, so deployment data mostly covers states it has mastered.
- **No cross-task transfer:** towel folding in 50 hotels yields millions of folds that never teach dishwasher loading.

The retorts and his answers:

- *"Teleop takes over beyond the model."* That is a novelty pump bolted onto the back, not the self-running flywheel that was pitched.
- *"Deployment finds long-tail failures."* Only if the robot reaches them and someone recognizes, logs and routes them into corrective data. That is an operations loop.

- *"On-policy data is still useful."* It saturates fast. His footnote concedes one use: $\pi^*0.6$ learns from its own rollouts through advantage conditioning.

One nuance he underplays: diverse deployments, such as different homes, do encounter novelty. The work is mining it, and Tesla's driving-data engine is exactly that: automated detection plus labeling. It is infrastructure, not magic.

The C4 analogy. Early LLMs trained on Common Crawl, which is full of duplicates and boilerplate. C4 (the cleaned crawl behind Google's T5) filtered and deduplicated it, so less data carried more distinct material. Today labs pay experts for financial models and for books in rare languages because those contain what the web lacks.

Why the ops team matters:

- Within-task correction needs a very tight loop between the operator and the researcher watching evals. Quality follows a power law: a great operator is worth many average ones.
- Across-task novelty needs someone who holds the global picture of what has already been collected. That runs against standard ops practice, which is to standardize everything and minimize the context anyone needs.

He predicts strong ops leads will be paid like top researchers.

3. "The Case Against Human Hands"

Thesis: copying the hand imports biology's constraints (small fingers, dense cabling, delicate joints) without its advantages (free self-repair, free softness, an energy budget that punishes extra limbs). Robots should get dexterity from what's cheap for machines: more arms, simple contact surfaces, big serviceable motors.

The two substrates:

- **Biology:**
 - It self-repairs, and soft tissue gives free **compliance** (yielding on contact, which absorbs errors).
 - Skin packs thousands of touch sensors.
 - It is severely energy-constrained, and parts can't be swapped.
- **Machines:**
 - No self-repair, so wear accumulates.
 - Power is cheap, cameras cost tens of dollars, and compute is centralized.
 - Parts are swappable.
 - The binding constraints are reliability and maintenance.

Why biology chose two hands with touch:

- An arm is about 5% of body mass. Daily energy use (**TDEE**, total daily energy expenditure) scales roughly with mass, both to maintain muscle and to carry it, so extra arms would cost roughly 5–10% more energy before counting the brain needed to control them.
- Eyes on the hands would need long, exposed optic nerves and more visual cortex. Brain is about 2% of body mass but uses about 20% of resting energy. Touch is cheap and local.

Why machines should choose differently: for a machine, wear is the problem.

- **Wear is real:** fingertips hit objects thousands of times a day.
 - ETH's open ORCA hand reported its first tactile skin wearing out after 2,000–4,000 cycles. At 60 grasps an hour, that is one to three days of shifts.
 - Tendons and cables flex constantly, and all of them cross the wrist.
- **The substrate-correct design:**
 - Put complexity in big, replaceable modules (arm motors).
 - Sense without contact (wrist cameras).
 - Make the parts that touch cheap to replace (3D-printed tips).

Parallel-jaw grippers are enough for most tasks. A **parallel-jaw gripper** is two flat fingers that slide together: one motor, few parts. Researchers distinguish:

- **Power grasps**, where the whole hand wraps a handle, like a hammer or knife;
- **Precision grips**, where fingertips hold small things, like a pen or screw.

Grippers can power-grasp most tools: to chop a cucumber, clamp the knife handle. For fine work, PI's RL Token drove small screws and closed zip-ties with parallel jaws. For truly harder tasks (his example is tying a bow), add a third or fourth arm rather than fingers. **DoF** (degrees of freedom) means independently controlled joints: a 6-DoF arm has six; a dexterous hand has 15–22.

Economies of scale won't rescue hands:

- **Learning curves:** cost falls a fixed percentage with each doubling of total units built (Wright's law), so it falls fast early and slowly later, until it hits a floor set by materials, part count, tolerances and assembly labor.
- **Watch movements** (finger-sized, about 100 parts, micron tolerances):
 - The Miyota 8215 fell about 87% in real terms since 1977, to about \$21, mostly by the late 1980s.
 - The Seiko NH35 is flat.
 - The Swiss ETA 2824-2 *rose* about 30% when Swatch restricted supply.
 - Prices now reflect market power, not manufacturing gains.
- **Hard drives:** a complete drive fell about 99.5%, to \$70–100, and has sat in a \$50–200 band for 20 years. The famous per-byte collapse came from packing more data onto the same mechanism; the mechanism itself didn't get much cheaper.
- **Both are optimistic stand-ins:** they are sealed and never touch the world.
- **Hands today:**
 - The cheapest five-finger hands cost about \$5,000–7,500 (Inspire RH56), roughly 250–350× a watch movement.
 - Premium hands (Schunk SVH about \$54k, Shadow over \$60k) are about 3,000×.
- **Big actuators are cheap:** their electronics ride chip scaling, and their mechanics are large and serviceable.
 - An integrated servo with motor, encoder and drive in one package (Teknic ClearPath) costs \$249.
 - A NEMA 23 brushless motor (a standard 2.3-inch frame) costs \$100–200.
 - A **planetary gearbox** (small gears orbiting a central one: compact and strong) costs about \$88.

- A **harmonic drive** (a flexible toothed ring bent by an oval cam, giving roughly 50–160:1 reduction with almost no play) starts around \$119.
- Whole 6-axis arms cost \$2–3k: I2RT YAM \$2,999, AgileX PiPER \$1,999.
- **The comparison:**
 - Two arms with two hands cost \$14–126k, dominated by the hands.
 - Three arms with grippers cost \$6–9k.

"Free" transfer from human video isn't free. Video gives intent, which objects change, and the rough order of contacts. It does not give joint targets, forces, friction or touch.

- A **friction cone** is the range of push directions a contact can apply without slipping, and its width depends on friction. Soft, grippy skin has a wide cone and deforms to enlarge the contact patch; hard robot fingertips don't. Humans also use fingernails, for example to pick up flat things or peel tape.
- Hence a whole literature on **retargeting**, which maps human hand motion onto robot joints:
 - Antotsiou et al. chain pose estimation, IK and particle-swarm optimization just to drive one hand.
 - DexPilot is a custom camera teleop system.
 - DexUMI uses a wearable exoskeleton that matches the robot hand, plus video inpainting of the robot hand over the human one.

Where he's strong: the wear argument, today's cost gap, and the real alignment work that human video needs.

Where to push back:

- **Hand prices are falling fast:** Linkerbot's O6 Lite sells for about \$560 after subsidy in China. The wear argument still stands.
- **Two arms may already suffice:** Google DeepMind's ALOHA Unleashed tied shoelaces (a bow) with two parallel jaws.
- **The third arm collides with his own data argument.** Humans have two arms, so no human video shows three-arm strategies, and teleoperating a third arm needs a second operator. If data is the bottleneck, human-like bodies get the most transferable data.
- **Some tools need fingers or adapted versions:** scissors, spray triggers, drill triggers, and in-hand reorientation.
- **The data side is moving toward hands.** EgoScale used a 22-joint five-finger hand as its shared action space, since human motion maps onto it most cleanly. Middle grounds are spreading too: three-finger hands (Unitree Dex3-1) and Atlas's four-digit hand.
- **Net:** for a defined commercial task today, grippers usually win. For general-purpose humanoids, it's an open bet.

4. The thread: "Why is no one selling post-training data?"

The idea:

1. Take an open base policy, such as $\pi_{0.5}$ or GR00T N1.7.
2. Pick a valuable task.

3. Run DAgger until the policy reaches 99.9% success ("three nines").
4. Sell the correction data as a premium bundle.

Why pretraining data is "slop" and post-training data is "value-add":

- Pretraining data is sold in bulk by the hour. Its quality is hard to verify, and each hour's marginal value is small. That is a "market for lemons": when buyers can't judge quality, prices sink to the level of bad data.
- DAgger data is novelty by construction: corrections exactly where a policy fails.

"If you can show videos of your policy working, we don't need to QA your data": the working policy is the quality certificate. You judge the data by what it produced.

What three nines takes to prove: with zero failures, you need about 3,000 straight successes to claim at most 0.1% failure with 95% confidence. At one attempt a minute, that is 50 hours of flawless running.

- A video shows possibility, not reliability.
- A buyer will want an evaluation, which he says must happen in-house. That is a tension inside his own pitch.

A catch his novelty essay implies: DAgger data fixes *that policy's* failure states, and a lab with a different base model fails elsewhere. The bundle is worth most when:

- the buyer uses the same model family, or
- the hard states belong to the task itself (cables snagging, cloth corners) rather than to one policy's quirks.

"Pivot to a neodeployment team, with ops paid for by your competitors": reaching three nines *is* the core skill of deploying robots. If labs buy your data, they fund your ops team while you learn, and later you can deploy yourself, possibly against them. "Neodeployment" reads like an analogy to "neoclouds": deployment firms that post-train others' base models.

The replies:

- **Nicolas Keller** offers remote DAgger: "connect our robots to your policy server." A **policy server** means the robot sends observations to the lab and gets actions back, so the lab's weights never leave its hands. That makes the service easy for labs to accept. His habit of training vanilla policies to check data quality is exactly Kyle's metric.
- **Abhinav** says most data companies have never trained a policy. Kyle agrees they are "dead by default" (Paul Graham's term for being on track to run out of money). Without training policies you can't tell which data is novel, so you end up selling volume.
- **Parav:** deployers already swap data for lab policies. Kyle: demand exceeds supply, and the swap is a temporary alliance, since each side eventually wants the other's half.
- **Dennis (Tixu)** "daggers" head-camera data from German tradespeople (*Facharbeiter*) and chefs. Strictly, DAgger needs a robot policy failing and being corrected. Human data can be *targeted*, but it can't be DAgger. He admits running robots is "a different beast," which is Kyle's point.
- **Alan:** task setup and transfer to the buyer's robot are hard, so "just deploy yourself." That is the portability problem.
- **seanpixel:** a worldwide remote-teleop DAgger factory. Round trips of 100+ ms across continents hurt fine manipulation, and the robots still need a site, resets and maintenance.

- **Wootzapp:** neutral third-party licensing and evals. That contradicts "evals in-house," but it is exactly what would resolve the buyer-verification problem.
 - **Deepanshu:** the flywheel works when deployment produces paid labor (hotel laundry). That answers *who pays* for the data, not *whether it's novel*. He conflates the two.
 - **KubaS:** data is a moat. If so, buyers will want exclusivity, which caps resale.
 - **Anoop:** "If you can get to three nines, why not deploy it yourself?" This is the strongest objection. The answers:
 - Deployment also needs sales, integration, uptime guarantees, field service, safety certification and liability cover.
 - Reliability in your own replica of the task may not survive a customer's site.
 - The same data can be sold to several labs.
 - Kyle himself pitches data as the on-ramp to deployment.
-

5. What this means for the options you're weighing

You're weighing four options: gloves data, egocentric video, simulation, or environments. Through Kyle's lens:

- **Generic egocentric video is pretraining data**, which he puts in the "hustle" category, and Build AI has already released 100,405 hours under Apache 2.0. It only differentiates if it carries novelty: skills models lack, plus proof it improves a policy.
- **Gloves data fits his favored category** (a demonstration stack proven by trained policies) if it is paired with a matching robot hand, as DexUMI and Sunday do. But his hands essay says the five-finger target may be wrong, while Figure, Tesla and NVIDIA's Sharpa-hand reference humanoid bet the other way. Your earlier "attack the hand" idea sits right on that fault line.
- **Simulation and environments are his least favored:** evals stay in-house, sim works poorly outside locomotion, and world models are years away. NVIDIA also gives away Isaac Lab-Arena. The most credible opening is neutral evaluation for data buyers.
- **His two endorsed wedges:**
 - post-training data with proven reliability;
 - demonstration tooling proven by trained policies.
- **Your edge:** a novelty pump is a data-infrastructure problem:
 - logging every intervention;
 - scoring each episode by how badly the current policy predicts it;
 - coverage maps and eval statistics.

Since he argues no horizontal data platform will win, that edge is strongest inside a service (post-training data or deployment) rather than as a standalone platform.

- **Your training-loop doc already agrees with him** on several points: evals before scale, offline proxies only after checking them against real trials, HG-Dagger correction rounds, and a gripper fallback.

Sources:

- [Open-Source Humanoid Robotics Landscape \(Oct 2026\)](#)
- [How Humanoids Learn](#)
- [Humanoid Training Loop: First Principles to Commercial Proof](#)