

A Survey of Agentic Robotics: Toward Continual Self-Improvement

Jiaming Wang, Yuhua Jiang, Zhengcheng Shen, and Harold Soh

Abstract—Foundation models (FMs), particularly large language models (LLMs) and vision-language models (VLMs), are increasingly used as the decision-making core of AI agents and are now being extended from digital environments to robotic systems. In classical robotics, engineers decompose tasks, specify objectives, write control code, and repair failures before deployment; in *agentic robotics*, an FM makes some of these decisions at run time or during improvement. Bringing this paradigm into the physical world introduces additional challenges: FM-mediated decisions must be grounded in executable robot behavior, respect physical constraints, and close the loop through real-world feedback. We survey this emerging paradigm and organize existing systems by three loci of FM-mediated agency: *task orchestration*, *motion/action generation*, and *improvement*. Coding rules for each locus make the definition operational, including a test of which improvement decisions the FM makes that separates fixed-procedure from agent-directed improvement. We apply these rules to the systems retrieved by a systematic search, examine verification and feedback-closure mechanisms, and compare what self-improving systems have demonstrated. Finally, we discuss continual self-improvement as a central emerging frontier and the challenges toward reliable autonomous improvement through physical interaction.

Index Terms—agentic robotics, embodied agents, continual self-improvement, foundation models

I. INTRODUCTION

Agentic robotics concerns how a robot organizes and adapts its problem solving, not merely how it maps observations to actions. We use the term for systems that delegate decisions about task execution or persistent improvement to a foundation model (FM), typically a large language model (LLM) or vision-language model (VLM). From goals and available evidence, the FM selects subgoals or skills, constructs programs or motion specifications, or proposes changes to behavior-generating components. These decisions are exposed through interfaces that execution and verification modules can act on. When execution feedback returns to the FM, it can revise its decisions using observed outcomes [1], [2]. The distinguishing feature is this *decision-making role* of the FM, rather than any particular model or interface.

This perspective distinguishes agentic robotics from two complementary traditions. Classical systems already plan over long horizons and respond to feedback, but their planners typically search within designer-supplied domain models, goal specifications, and objectives [3]. An FM can also

help formulate or revise these specifications. Robot learning instead acquires policies from demonstrations or interaction, but reliable long-horizon execution and sample-efficient training remain challenging. Imitation policies can compound small errors as they encounter states absent from demonstrations [4], while reinforcement learning faces difficult exploration and credit assignment using costly robot experience [5]. Vision-language-action (VLA) models such as RT-2, OpenVLA, and π_0 transfer pretrained knowledge into instruction-conditioned control [6]–[8]. Our focus is not this observation-to-action mapping alone, but the FM’s participation in the surrounding problem-solving process. Agentic systems can reuse such policies or conventional skills while an FM decides how to compose them or generates new executable specifications [9], [10]. This offers a route to new task combinations without training a complete policy for each goal, rather than a guarantee of greater reliability or sample efficiency.

We make this definition operational at three loci. At the task level (T), the FM decides *what* behavior to pursue: subgoals, skill calls, or task specifications. At the motion level (M), it decides *how* behavior is realized: geometry, objectives, control code, or actions. At the improvement locus (I), it contributes persistent changes to behavior-generating components. It may supply candidates within a designer-fixed update procedure (I_{fixed}), or also decide how improvement proceeds, such as which kind of component to change, what evidence to collect, or which tests to run (I_{agent}). A system need not instantiate all three loci; open-loop planners are partial instances, and classical planners and learned policies can remain execution components. This extends the FM-centered decision loops of software agents [11]–[13] to physical systems: decisions must be grounded in geometric, dynamic, contact, actuation, and safety constraints, and outcomes inferred from noisy, partial observations. Delegating more decisions therefore expands both what can be adapted and what must be verified.

Relation to existing surveys. A systematic review of foundation models in robotics organizes the field by model type, architecture, learning paradigm, task, and domain [14]; a survey of large-model embodied AI covers hierarchical decision making with feedback, end-to-end VLA models, embodied learning, and world models [15]. The closest, *Towards Embodied Agentic AI* [16], classifies LLM/VLM-driven robots by integration and agent role, separates planners from models that output actions, contrasts scripted workflows with agents, and discusses memory and self-extending skill libraries. Surveys of self-improving agents separate updates by a fixed external optimizer from updates the agent makes to itself, with robotics as one of six domains [17]. We analyze jointly, in one robotic-system

J. Wang and H. Soh are with the National University of Singapore, Singapore. Corresponding author: Jiaming Wang (jiaming@comp.nus.edu.sg); ORCID: 0009-0006-1416-4606.

Y. Jiang is with Tsinghua University, Beijing, China.

Z. Shen is with the Max Planck Institute, Germany.

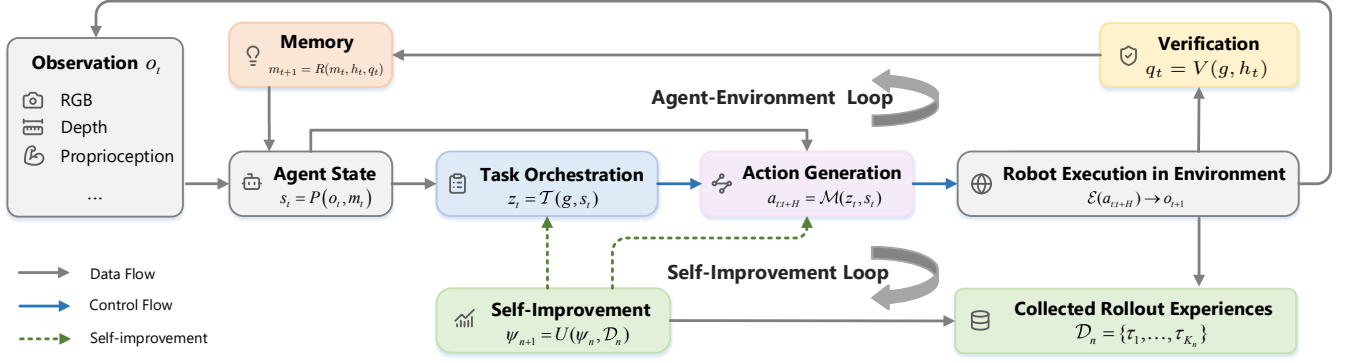


Fig. 1. **Unified architecture of agentic robotics.** During task-time interaction, the current observation o_t and retained memory m_t are combined to construct the agent state s_t ; task orchestration determines *what* behavior to perform, while action generation determines *how* that behavior is physically realized. Robot execution produces subsequent observations and verification signals q_t , which can update memory and influence later decisions, thereby closing the agent–environment loop. Across rollouts, accumulated interaction trajectories form the experience set $\mathcal{D}_n$, which drives a slower self-improvement process U that updates persistent system components ψ and consequently changes future task orchestration and/or action generation. Gray, blue, and green dotted arrows denote data flow, control flow, and self-improvement pathways, respectively. The diagram represents a functional superset rather than a mandatory pipeline; individual systems may instantiate only a subset of these modules and feedback pathways.

framework, four properties that these works treat separately: where the FM takes part in generating behavior, which improvement decisions it makes, who verifies the outcome, and how feedback is retained. The joint view separates systems that share a role label, such as “coding agents” (Sec. VI).

Scope. Under the definition above, we include a system when an FM makes or revises a decision that affects robot behavior in a goal-directed task, or takes part in a loop that persistently improves the system, on a physical robot or in an embodied simulator built on a physics engine or on 3D scans of real buildings. Systems that implement only part of the agent loop, such as open-loop planners, are partial instances. Standalone foundation policies are outside our scope, as a choice of focus rather than a claim that they lack agency: we study FM decisions that can be separated from the rest of the loop and coded. A foundation policy enters when an FM planner or verifier steers it, or when its own experience drives its update, as in RoboCat [18]; its actions then count as motion-level participation.

Contributions. We (i) present a unified view of the agent–environment loop and a slower self-improvement loop (Sec. III); (ii) make the definition operational with coding rules for the three loci, including a test of which improvement decisions an FM makes that separates fixed-procedure from agent-directed improvement, and apply them to every system retrieved by a systematic search (Secs. II and IV); (iii) analyze verification and feedback closure, and compare what self-improving systems have actually demonstrated (Secs. V and VI); and (iv) discuss the challenges of continual self-improvement, a central emerging frontier, including when giving the FM more authority is worth its cost (Sec. VII).

II. SURVEY METHODOLOGY

We constructed the survey corpus through a systematic search, screening, and coding pipeline. We searched arXiv and OpenAlex for work first released between 1 January 2022 and 23 September 2026 using a query combining foundation-model, robotics, and agency terms, and removed duplicates

automatically. Screening, coding, adjudication, and full-text analysis were performed by LLM agents (Claude models) following written protocols, processing one record at a time and logging each decision. A screening agent evaluated titles and abstracts in two stages using the scope criteria of Sec. I. An independent agent re-screened 150 randomly sampled records and agreed on 86% of eligibility decisions. A coding agent then labelled each of the 2,704 candidate systems from its abstract according to the rules of Sec. IV and flagged ambiguous cases. A second agent, blind to the original labels, independently coded 320 systems, achieving Cohen’s κ of 0.90–0.96 across loci. An adjudicating agent reviewed every flagged case, and together with full-text reading excluded 361 systems that fell outside the defined scope. The resulting corpus contains 2,343 systems; all corpus-level trend analyses use this complete set. Records, protocols, decisions, and labels will be released with the paper.

For deeper qualitative analysis, we examined the full texts of 78 systems from the corpus. These include the 20 most-cited systems, which capture influential designs on which later work builds, together with recent systems introducing distinct mechanisms, particularly agent-directed improvement. This set spans all agency loci and feedback-closure regimes considered in the survey.

III. OVERVIEW OF AGENTIC ROBOTICS

Fig. 1 gives a unified view of where FMs sit between perception, decision making, physical execution, and learning. It comprises two loops. The *agent–environment loop* converts observations into decisions, executes them, and incorporates the resulting feedback. The *self-improvement loop* runs on a longer timescale and uses accumulated experience to modify persistent components that govern future behavior.

A. Agent–Environment Loop

During deployment, the robot repeatedly builds a decision state, decides what to do and how to realize it, executes, and

evaluates the outcome:

$$s_t = P(o_t, m_t), \quad z_t = \mathcal{T}(g, s_t), \quad a_{t:t+H} = \mathcal{M}(z_t, s_t), \quad (1)$$

$$o_{t+1} \sim \mathcal{E}(a_{t:t+H}), \quad h_t = (o_{1:t+1}, \{a_{k:k+H}\}_{k=1}^t), \quad (2)$$

$$q_t = V(g, h_t), \quad m_{t+1} = R(m_t, h_t, q_t). \quad (3)$$

Here t counts decision points rather than control ticks: the executor runs the action chunk $a_{t:t+H}$, or a prefix of it in receding-horizon fashion, until the next decision, and o_{t+1} is the observation at that point. o_t is sensor-derived information (images, depth, proprioception, force, detections, or scene graphs), m_t is memory retained from earlier interaction, and P combines the two into the agent state s_t . Given the goal g , task orchestration $\mathcal{T}$ decides *what* to do, producing a task-level decision z_t such as a subgoal, selected skill, or plan [9]. Action generation $\mathcal{M}$ decides *how* to do it, producing an action specification $a_{t:t+H}$: actions or trajectories, poses, keypoints, geometric constraints, objectives, skill parameters, or control code [19], [20]. The execution operator $\mathcal{E}$ abstracts the physical interaction, whose full state the robot never observes. From the history h_t of observations and executed actions, the verifier V produces a feedback signal q_t that may encode success, progress, constraint violations, uncertainty, or failure explanations [1], [2], and the retention process R decides what enters memory [21]. This decomposition exposes two loci of behavior-generating FM participation: at the *task level* the FM participates in $\mathcal{T}$, and at the *motion level* in $\mathcal{M}$. Neither is mandatory: SayCan delegates physical realization to fixed skills [9], whereas PIVOT and Code as Policies place FM reasoning at the motion interface [10], [22].

B. Self-Improvement Loop

Interaction also produces experience from which the system itself can improve. Let the k -th rollout and the experience retained at improvement iteration n be

$$\tau_k = (o_t, m_t, s_t, z_t, a_{t:t+H}, q_t)_{t=1}^{T_k}, \quad \mathcal{D}_n = \{\tau_k\}_{k=1}^{K_n}, \quad (4)$$

where fields a system does not instantiate are simply omitted. The self-improvement loop modifies a persistent artifact ψ using this experience,

$$\psi_{n+1} = U(\psi_n, \mathcal{D}_n), \quad (5)$$

where ψ may be a learned policy, reward function, controller code, prompt, skill library, software repository, or even part of the learning infrastructure [18], [23], [24]. This separates self-improvement from memory. Updating m_t changes the information available to later decisions; updating ψ changes the mechanism that generates future behavior, and the change persists beyond the current rollout.

C. Evolution of Foundation-Model Participation

Fig. 2 places representative systems by the year of their first public version, and Fig. 3 shows the coded corpus. In 2022–2023, FMs were mainly used to generate plans, select skills, or revise high-level decisions above a largely fixed execution stack [2], [9], [25]–[28]. Multimodal models then also generated lower-level behavior through control programs, geometric

objectives, keypoints, constraints, and action candidates [10], [19], [20], [22], [29], [30]. The corpus confirms this shift. Corrected with the full-text sample (Sec. II), the share of systems in which the FM produces motion-level outputs rose from 27% in 2023 to 63% in 2026 (difference 37 points, 95% CI 17–54); the abstract-level shares in Fig. 3, 21% and 43%, understate both the level and the rise. Task-level participation stayed near three quarters. Persistent improvement is not new: about one system in five includes it each year, with no trend, mostly through fixed data-collection, distillation, reward-search, or learning pipelines [18], [23], [31]. What is new is agent-directed improvement, in which the FM also chooses evidence, runs tests, or changes the procedure [24], [32], [33]. It is still rare, 17 of the 2,343 systems, but 16 of them appeared in 2026, where they make up 10% of improving systems, against under 1% in 2023–2025. FM decisions thus reach more parts of the behavior-generation pipeline: on full text, about half of 2023 systems but under a third of 2026 systems (51% and 28%) give the FM a task-level role only. This describes where FM decisions appear, not a ranking, and Sec. VII discusses when extra freedom is worth its cost.

IV. LOCI OF FOUNDATION-MODEL PARTICIPATION

The architecture above suggests an organizing question: *where does the FM participate in behavior generation or persistent improvement?* We distinguish three loci, $\mathcal{L} \subseteq \{T, M, I\}$: task orchestration, motion/action generation, and improvement; one FM may serve several. We code each locus from what the FM actually outputs or decides in the system as evaluated, not from how the system is described (Table II). Improvement needs one further distinction, based on which decisions of the improvement process the FM makes (Sec. IV-C). Fig. 4 illustrates the loci on one task, and Table I places every synthesized system by locus and feedback-closure regime.

A. Task-Level Agency

At the task level, the FM selects temporally extended behaviors such as subgoals, skills, or robot operations. Two questions arise: *admissibility* (can the robot realize the behavior in its current state?) and *utility* (which admissible behavior best advances the goal?). Existing systems answer them through four complementary mechanisms.

Admissibility by construction. The most direct approach restricts FM reasoning to an explicitly exposed set of actions, skills, or APIs: Language Models as Zero-Shot Planners maps free-form plans to a fixed action set, and ProgPrompt exposes available actions as program structure in the prompt [25], [26]. Richer tool interfaces follow the same principle, from ARNA’s perception, reasoning, and navigation tools to Maestro’s coding agent composing heterogeneous robot APIs [30], [63].

State-conditioned grounding. An exposed skill may still be infeasible in the current state. SayCan weighs semantic usefulness by a learned affordance estimate [9]; SayPlan and BUMBLE ground decisions in scene graphs or in observations and robot state, and Statler’s LLM maintains a world state predicted from its own actions [21], [28], [73]; NLMap and ConceptGraphs tell the FM which objects are present, and

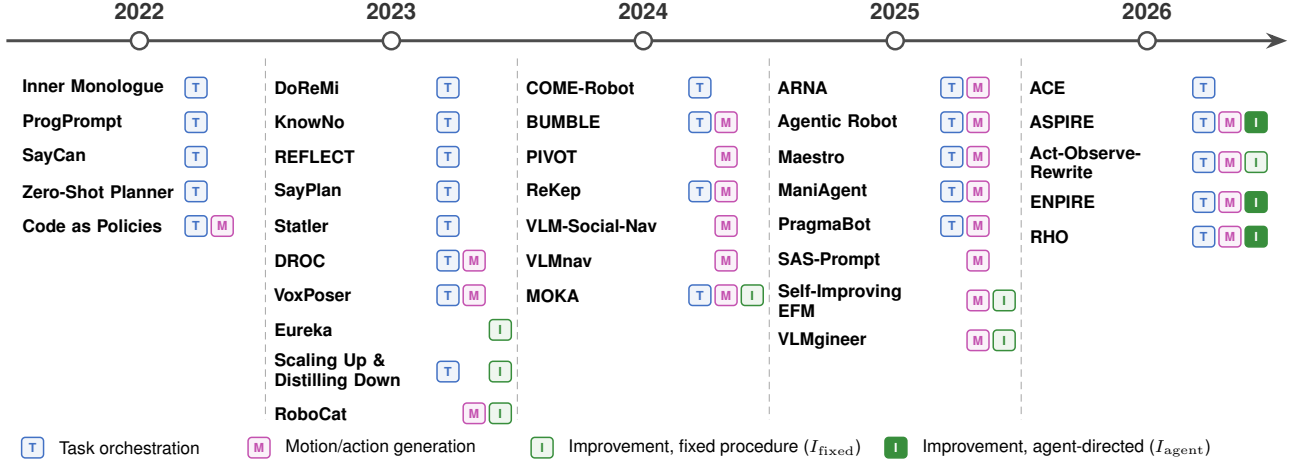


Fig. 2. **Evolution of foundation-model participation in robotics.** Representative systems by year of first public version, marked by the loci at which FMs influence robot behavior: task orchestration (T), motion/action generation (M), and improvement (I ; outlined: I_{fixed} , filled: I_{agent}), coded from full text as in Table I. The progression shows a trend, not a strict chronology or a ranking.

TABLE I

THE 78 SYNTHESIZED SYSTEMS BY FEEDBACK-CLOSURE REGIME (ROWS) AND FM PARTICIPATION IN TASK ORCHESTRATION AND MOTION/ACTION GENERATION (COLUMNS), CODED FROM FULL TEXT. ALL SYSTEMS IN THE LAST ROW MODIFY A PERSISTENT ARTIFACT; BOLD MARKS AGENT-DIRECTED IMPROVEMENT (I_{agent}), THE OTHERS FIXED-PROCEDURE IMPROVEMENT (I_{fixed}). THE REGIME RECORDS HOW FEEDBACK IS RETAINED, NOT HOW CAPABLE A SYSTEM IS.

Regime	T only	M only	T and M	Improvement only
Open loop	HULC++ [34], LM-Nav [35], Zero-Shot Planner [25], AutoTAMP [36], PG-VLM [37], SMART-LLM [38]	–	VLMs [39], LLM-GROP [40], RoboGPT (construction) [41], Uni-Skill [42]	–
Step-level	NLMap [43], ProgPrompt [26], SayCan [9], CLAIRify [44], ConceptGraphs [45], KnowNo [27], L3MVN [46], PaLM-E [47], Text2Motion [48], TidyBot [49]	Language to Rewards [50], PIVOT [22], VLM-Social-Nav [51], VLMnav [52]	Code as Policies [10], VoxPoser [19], ReKep [20], ELLMER [53]	–
Within-rollout memory	LLM-Planner [54], Inner Monologue [1], DoReMi [55], REFLECT [2], SayPlan [28], Statler [21], ChemAgents [56], COME-Robot [57], ORGANA [58], ACE [59]	NavGPT [60]	RoCo [61], Agentic Robot [62], ARNA [63], ManiAgent [64], RAI [65], Reliable Code-as-Policies [66], ALRM [67]	–
Cross-rollout memory	LongAct [68]	SAS-Prompt [69]	ChatGPT Research Group [70], DROC [71], Incremental humanoid learning [72], BUMBLE [73], Maestro [30], PragmaBot [74]	–
Persistent modification	Gen2Sim [75], HALP [76], RoboGen [77], Scaling Up [31], Agentic Skill Discovery [78], InterPreT [79], LASP [80]	ELM [81], RoboCat [18], AIC MLLM [82], LLM locomotion design [83], LAMARL [84], Self-Improving EFM [85], VLMgineer [86]	LLM-Exo [87], UniSim [88], MOKA [29], Act-Observe-Rewrite [89], ASPIRE [32], CaP-X [90], ENPIRE [24], RHO [33]	Eureka [23], DrEureka [91]

the code VLMs’ LLM writes looks up landmark positions at execution time [39], [43], [45]; PaLM-E plans directly from sensor inputs [47]; and KnowNo asks for help when its calibrated uncertainty is too high [27].

Admissibility through downstream realization. Task-level decisions need not map to a predefined skill if a downstream mechanism can realize open-ended goals. Code as Policies composes perception and control APIs into programs [10]; VoxPoser, MOKA, and ReKep ground subtasks into value maps, motion specifications, or geometric constraints [19], [20], [29]; and ManiAgent delegates realization to action-generation agents [64]. In task-and-motion planning, feasibility checks or a TAMP solver accept, reject, or instantiate FM proposals in Text2Motion and LLM-GROP [40], [48], and AutoTAMP

translates instructions into temporal-logic specifications for a planner [36].

Execution feedback and recovery. Because feasibility cannot always be judged in advance, some systems revise task-level decisions from execution feedback. Inner Monologue, REFLECT, and DoReMi use observations, success signals, or execution history; LLM-Planner re-plans from the objects observed so far when the agent gets stuck, so unlike SayCan it needs no list of admissible skills; and DROC retrieves knowledge distilled from past corrections [1], [2], [54], [55], [71]; COME-Robot and ACE retry, restore, or replan after failed subgoals, Agentic Robot’s verifier triggers a fixed recovery routine [57], [59], [62], and PragmaBot keeps lessons from trial and error for later tasks [74]. These mechanisms trade coverage

TABLE II

LOCI OF FM-MEDIATED AGENCY: WHAT THE FM DECIDES, THE RULE USED TO CODE EACH LOCUS FROM WHAT THE FM ACTUALLY OUTPUTS, AND REPRESENTATIVE SYSTEMS.

Locus	The FM decides...	Coded when the FM output contains...	Representative systems
T	<i>what</i> behavior to pursue next	a subgoal, plan, skill/tool/API call, or <i>semantic</i> target (an object or place, by name or ID) that other modules turn into geometry	SayCan, ProgPrompt, SayPlan, KnowNo, COME-Robot [9], [26]–[28], [57]
M	<i>how</i> the behavior is physically realized	<i>geometric or control</i> information: actions or a chosen action candidate, poses, waypoints, keypoints, constraints, cost maps, rewards optimized at execution time, metric skill parameters, or code that computes them; or the actions of an FM policy	PIVOT, VoxPoser, ReKep, Code as Policies, RoboCat [10], [18]–[20], [22]
I_{fixed}	the <i>content</i> of a candidate inside a prescribed update	a new policy, reward, program, or skill (or the data or feedback for it), while the designer fixes which artifact changes, what evidence the FM sees, how candidates are tested, and the update procedure	Scaling Up & Distilling Down, RoboCat, Eureka, CaP-X, Act-Observe-Rewrite [18], [23], [31], [89], [90]
I_{agent}	also <i>how</i> the improvement proceeds	a candidate plus at least one process decision: which artifact to change, what evidence to collect, which tests or experiments to run, or how the update procedure works (Table III)	RHO, ASPIRE, ENPIRE [24], [32], [33]

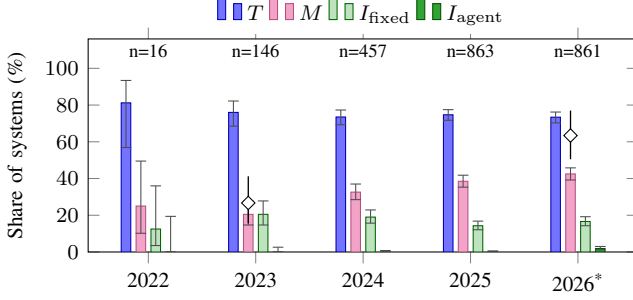


Fig. 3. Share of the final corpus at each locus by year of first public version, coded with the rules of Sec. IV. Bars: abstract-level labels after adjudication (improvement type from full text where the abstract showed an agent-directed or unclear loop), with 95% Wilson intervals; abstracts understate motion-level outputs, increasingly in later years. Diamonds: motion-level share corrected with full-text coding of a random sample (Sec. II), with 95% CIs. * 1 January–23 September 2026; n : systems per year.

for reliability. A fixed action set guarantees executability only for the behaviors it contains, and mapping free-form plans onto it raised executability at some cost in correctness [25]. Grounding is only as good as its models: SayCan needs a value function per skill, and KnowNo’s guarantee holds only for its calibration distribution [9], [27]. Downstream realization widens the behavior space but moves failures into perception and solvers, and recovery is only as good as the verifier (Sec. V).

B. Motion-Level Agency

At the motion level, the FM determines information that directly governs physical realization. Its output is often consumed by a conventional planner, optimizer, controller, or learned skill. The boundary with the task level follows what the FM outputs. Choosing a behavior or a semantic target, such as `pick (mug)` or “go to the kitchen,” is task-level even though it determines where the robot goes, because another module computes the geometry. Outputting that geometry—a pose, keypoint, distance, or code that computes it—is motion-level. VLMs shows both in one program: its LLM writes navigation code with landmark

targets and metric moves such as “move forward 3 m” [39]. Three interface patterns are common (Fig. 4a).

Direct action and trajectory generation. PIVOT renders continuous action candidates as visual alternatives for a VLM to select [22]; NavGPT picks the next viewpoint of a navigation graph from text descriptions of the surrounding views and a summarized history [60]; VLMnav turns VLM spatial reasoning into navigation decisions [52]; and AIC MLLM predicts SE(3) interaction poses and corrects them from interaction outcomes; its test-time model updates gave no gain in average success [82].

Geometric or objective-mediated generation. The FM specifies a structured intermediate representation and a conventional solver handles continuous realization: 3D value maps in VoxPoser [19], point-based affordances in MOKA [29], relational keypoint constraints in ReKep [20], reward parameters in Language to Rewards [50], and social costs for a navigation planner in VLM-Social-Nav [51].

Programmatic or parameterized generation. The FM can also write executable behavior. Code as Policies synthesizes waypoint logic and reactive controllers [10], Maestro composes perception, planning, and control APIs [30], and CaP-X shows that coding-agent performance depends strongly on the abstraction of the exposed robot interface [90]. Parameterized skill invocation is a closely related case: BUMBLE picks a skill and one of several parameter options drawn on the image, such as a movement direction, and ARNA writes navigation coordinates as tool arguments [63], [73]. The patterns differ in what they depend on. Selecting rendered candidates needs no solver but reaches only the options drawn; geometric and objective interfaces depend on perception to track what the FM specified, as ReKep’s solver re-optimizes constraints at about 10 Hz on keypoints tracked at 20 Hz [20]; and programs inherit the limits of the exposed API, remaining brittle for contact-rich skills such as insertion [90].

C. Self-Improvement: How Does the System Change Future Behavior?

Self-improvement changes a persistent mechanism ψ that affects future task instances. Two boundaries need explicit

(a) What the FM outputs for “put the mug in the drawer”

Task level (T) <code>open(drawer); pick(mug); place(mug, drawer)</code> Chooses skills and semantic targets; fixed skills compute the motion. SayCan, ProgPrompt, COME-Robot	Motion: pick a candidate (M) <code>"arrow 2"</code> Chooses among rendered geometric actions. PIVOT, VLMnav	Motion: geometry or objective (M) <code>k_handle→k_grasp; mug upright</code> Specifies keypoints, constraints, or cost maps; a solver computes the trajectory. VoxPoser, ReKep, MOKA	Motion: program (M) <code>g = grasp_pose(mug, "handle") move_to(g.offset(z=0.10))</code> Writes code that computes poses or control. Code as Policies, Maestro, CaP-X
--	---	---	---

(b) The grasp slips: where the feedback goes

Task recovery Verifier: “mug not in gripper”. The FM retries, replans, or asks for help. Lasts: this episode Inner Monologue, COME-Robot	Motion correction The FM moves the grasp to the handle, or a solver re- optimizes its constraints. Lasts: this episode AIC MLLM, ReKep	Cross-rollout memory Stores “grasp mugs by the handle”; later prompts retrieve it. Lasts: later tasks, as context DROC, PragmaBot	Improvement, I_{fixed} A fixed loop retrains the grasp policy, or the FM rewrites a reward the loop then scores. Lasts: later tasks, new artifact RoboCat, Eureka	Improvement, I_{agent} A coding agent reads logs, picks the grasp skill to fix, edits and tests it; a fixed gate promotes it. Lasts: later tasks, new artifact RHO, ENPIRE
---	---	--	--	---

Fig. 4. **One task, different loci of FM agency.** (a) What the FM outputs for the same instruction at the task level and in the three motion-level interface patterns; the listed systems use these interfaces, not this exact task. (b) Five ways to handle the same failure, and what each change retains. Only the last two change a behavior-generating artifact; they differ in which improvement decisions the FM makes (Table III).

rules.

Memory or improvement? A retrieved lesson and an edited prompt can change later decisions in the same way, so this boundary is a coding convention. Experience counts as improvement when it changes an artifact that later instances run in place of the old one: weights, a training reward, program code, an executed skill, a symbolic planning model, or the agent’s own prompts and tools. It counts as memory when it is stored as records (lessons, corrections, trajectories) that are retrieved into the FM’s context or replayed. A system can do both. ASPIRE’s “skills” are text notes on failures and repairs that later coding agents read into their context, so they are memory; its improvement is the repaired program it keeps for each task [32].

Who makes the improvement decisions? Both regimes may use an FM to write the new policy, program, reward, or skill. Building on the distinction between fixed external update rules and updates an agent makes to itself [17], we separate them by five observable decisions (Table III): which kind of artifact to change—policy, reward, training procedure, or data (D1); what evidence to collect (D2); which tests or experiments to run (D3); how the update procedure works (D4); and whether a candidate is kept (D5). A system is I_{agent} if the FM makes at least one of D1–D4; which files or reward terms it edits inside one artifact is part of the candidate. D5 does not enter the label, because a fixed acceptance rule is often a deliberate safety choice.

Fixed-procedure improvement (I_{fixed}). Scaling Up and Distilling Down, RoboCat, and Self-Improving Embodied Foundation Models (EFM) improve policies through predefined data-collection, distillation, or learning pipelines [18], [31], [85]; RoboGen, Agentic Skill Discovery, Gen2Sim, LAMARL, and HALP acquire skills or policies through prescribed pipelines from FM-proposed tasks, rewards, or human teaching [75]–[78], [84]; and Eureka and DrEureka (rewards), VLMgineer (tools), ELM (robot designs), and CaP-X (a skill library built in one pass) place an FM inside a fixed optimization loop [23], [81], [86], [90], [91]. SAS-Prompt, by contrast, regenerates controller parameters from rollout traces in its context on

TABLE III
WHICH IMPROVEMENT DECISIONS THE FM MAKES, FROM THE FULL TEXTS. D1 KIND OF ARTIFACT TO CHANGE; D2 EVIDENCE TO COLLECT; D3 TESTS OR EXPERIMENTS TO RUN; D4 UPDATE PROCEDURE; D5 ACCEPTANCE, GIVEN AS WHO DECIDES (HUMAN, FIXED RULE, FM; NR: NOT REPORTED).
 • THE FM DECIDES; ◦ FIXED BY THE DESIGNER; – ONLY ONE ARTIFACT CAN CHANGE. I_{agent} REQUIRES • IN AT LEAST ONE OF D1–D4. THE LAST ROW COUNTS, OVER ALL AGENT-DIRECTED SYSTEMS IN THE CORPUS, HOW MANY LET THE FM MAKE EACH DECISION.

System	Changed artifact	D1	D2	D3	D4	D5	Label
RoboCat [18]	policy weights	–	◦	◦	◦	human	fixed
Eureka [23]	reward code	–	◦	◦	◦	rule	fixed
CaP-X [90]	skill library	–	◦	◦	◦	NR	fixed
Act-Observe-Rewrite [89]	policy code	–	◦	◦	◦	rule	fixed
RHO [33]	policy repository	–	•	•	◦	rule	agent
ASPIRE [32]	per-task program	–	•	•	◦	rule	agent
ENPIRE [24]	policy, training code, data	•	•	•	•	FM	agent
All 17 I_{agent} systems: FM decides		7	15	10	7	5	

every trial, so it is coded as cross-rollout memory [69]. The candidate can still change a lot: Eureka’s LLM can change the reward’s functional form after reading training statistics, and Act-Observe-Rewrite’s LLM rewrites the vision module or the controller from outcomes and key frames chosen by a fixed rule, but it runs nothing itself, and every candidate that compiles is loaded for the next trial [89].

Agent-directed improvement (I_{agent}). RHO keeps a prescribed evolutionary loop (evaluate, summarize, mutate, keep improvements) and a fixed acceptance rule, but its mutation step is a coding agent that queries the simulator, inspects traces, and writes and runs its own tests before submitting a new policy repository [33]. ASPIRE’s coding agents choose which execution traces to inspect and which failed seeds to replay while repairing a task’s program [32]. ENPIRE delegates the most: its agents choose whether to change the policy, the learning algorithm, the data mix, or the training code, run their own real-robot experiments, and decide which recipes to keep; only the reward and reset code, built earlier under human review, stays fixed [24]. The decisions are not bundled. Across all agent-directed systems in the corpus, the FM chooses its evidence in 15 of 17, its tests in 10, the procedure in 7, and

the kind of artifact in 7 (Table III). RHO and ASPIRE delegate evidence and tests but keep the procedure and acceptance fixed, which keeps the evaluation independent of the agent.

Coding boundary cases. Some systems do not fit the locus definitions cleanly from their descriptions alone. We therefore used five rules for recurring ambiguous cases; systems that remained uncertain during initial coding were flagged and adjudicated using these rules. First, we classify an FM by the *decision its output determines*, not by the output format. If an FM scores candidate behaviors and a fixed rule such as an argmax or threshold selects among them, we treat the FM as making that decision. Thus, SayCan’s skill scores and L3MVN’s frontier scores count as task-level outputs [9], [46]; similarly, an FM-generated objective or constraint optimized by a motion solver counts as motion-level. In contrast, an FM that only estimates scene state, object properties, human intent, or execution success for a separate planner or fixed recovery rule is coded as perception or verification, not as an agency locus. Second, a foundation policy controlled only by non-FM modules is outside our scope. Third, FM-generated tasks, data, rewards, or designs count as improvement only when experience produced by the system’s own simulated or physical interaction is used to update a persistent behavior-generating artifact, as in ELM [81]. Fourth, stored trajectories, corrections, or lessons that are only retrieved into later FM context are treated as memory rather than improvement, following Sec. IV-C. Fifth, when an FM produces a symbolic or formal specification that a conventional planner subsequently solves, we code the FM’s contribution as task-level because it defines *what* behavior must be achieved rather than its physical realization, as in AutoTAMP [36]. A system that satisfies none of the three locus definitions is excluded from the corpus.

V. VERIFICATION AND FEEDBACK CLOSURE

Beyond where the FM participates, agentic systems differ in how interaction is verified and how the resulting evidence is retained and routed back.

A. Verification

Verification turns interaction outcomes into evidence for recovery, behavior selection, and self-improvement. Systems differ in *where the signal comes from* and *how informative it is*.

Simulator-provided verification. Simulation provides privileged state, so success predicates, rewards, and detailed metrics come cheaply; Eureka, DrEureka, and RHO use such statistics to optimize reward programs, policies, or policy repositories [23], [33], [91].

Automated physical-world verification. In the real world, even success must be inferred from sensors. Inner Monologue uses a learned success detector in its kitchen setting, and PragmaBot and COME-Robot use VLMs to judge execution [1], [57], [74]. Self-Improving EFM learns a visual *steps-to-go* evaluator that gives success and progress, and ENPIRE composes perception, geometric, and force checks into task-specific evaluators for real-robot rollouts [24], [85].

Human-provided verification. When outcomes or failure causes cannot be inferred reliably, humans supply rich supervision, as in the language corrections that DROC distills into reusable knowledge [71]; such feedback limits the autonomy of the improvement loop.

Feedback richness. Verification can report an *outcome* (success or failure), as in the success detection of Inner Monologue and the binary evaluators of ENPIRE [1], [24]; *progress*, such as the steps-to-go estimates that give Self-Improving EFM a dense reward [85]; or a *diagnosis*, as when DoReMi flags violated constraints, REFLECT explains failures, AIC extracts corrective information, and ASPIRE localizes failures in execution traces [2], [32], [55], [82]. Outcomes are easy to obtain and audit but assign credit poorly; progress and diagnoses speed up learning and recovery but depend on learned evaluators that can be biased or exploited. Unlike simulation, the physical world offers no general ground-truth evaluator, which makes reliable automatic verification a central bottleneck for self-improvement.

B. Feedback Closure

Feedback closure determines how verified evidence influences later behavior: *what* is retained, in which *regime*, and *where* it is routed. Memory is therefore part of feedback closure rather than a separate module.

What is retained? Some systems keep an updated *state or belief*, such as the world state of Statler or the workflow and object state of ACE [21], [59]. Others keep *execution traces*: Inner Monologue, BUMBLE, and PragmaBot condition later decisions on interaction history, and coding agents such as Maestro and CaP-X receive execution outputs, errors, or visual differences [1], [30], [73], [74], [90]. More compact memories keep *diagnoses or lessons*: REFLECT summarizes experience into failure explanations and DROC distills corrections into retrievable knowledge [2], [71].

In which regime? Closure falls into four regimes: step-level closure, within-rollout memory, cross-rollout memory, and persistent modification. The first two act within one task instance. The last two both reach later instances and differ in what is kept: records that are retrieved into the FM’s context, or a changed executable or learned artifact (Sec. IV-C). The regime says how feedback is retained, not how capable a system is: a step-reactive controller can be more reliable than a system that learns across rollouts. Feedback may also bypass the FM: in VoxPoser and ReKep a solver re-optimizes FM-written value maps or constraints at every step, and in Code as Policies and ELLMER the generated program runs its own perception–action loop [10], [19], [20], [53], whereas in Inner Monologue, REFLECT, DoReMi, and COME-Robot the FM itself receives outcomes within a rollout [1], [2], [55], [57]. DROC and PragmaBot carry corrections or lessons across rollouts, BUMBLE retrieves a fixed set of earlier failures annotated offline [71], [73], [74], and self-improving systems compile evaluated experience into persistent artifacts [18], [23], [24], [32], [89].

Where does feedback return? At the *motion level*, feedback changes how the current behavior is realized: AIC’s MLLM

corrects the interaction pose from the outcome of the previous attempt, while in ReKep and VoxPoser a solver adjusts the motion [19], [20], [82]. At the *task level*, it changes which behavior comes next: Inner Monologue, REFLECT, DoReMi, and COME-Robot continue, retry, recover, or replan [1], [2], [55], [57]. BUMBLE closes both levels [73]. Feedback can also return to the *improvement loop* and modify a reward, controller program, program repository, or policy, as in Eureka, Act-Observe-Rewrite, RHO, ASPIRE, and ENPIRE [23], [24], [32], [33], [89].

VI. BENCHMARKS AND EVALUATION

Evaluation remains fragmented because systems place FM participation at different loci and expose different robot interfaces, so terminal task success is hard to compare across methods.

A. Benchmarks for Task- and Motion-Level Agency

Task-level benchmarks check whether an FM turns a semantic goal into admissible, effective behaviors: VirtualHome checks plans against executable symbolic actions [25], [26], [92], Embodied Agent Interface separates goal interpretation, decomposition, sequencing, and transition modeling [93], and LoHoRavens, LongAct, and EmbodiedBench add execution feedback, memory, and replanning [68], [94], [95]. Motion-level evaluation mostly reuses RL Bench, CALVIN, and LIBERO [96]–[98] or varies the robot interface itself, as CaP-Bench does [90]. Most influential systems use custom suites, and physical evaluations are small: VoxPoser uses five real tasks with ten trials each and BUMBLE 70 trials in three buildings, while SayPlan’s quantitative results come from a scene-graph simulator [19], [28], [73]. Such studies show that an interface works, not how reliably over long deployments, and differing skills, perception, and protocols make them hard to compare. Results should be reported with the interface, observations, planners, and controllers the FM relies on, and its opportunities for correction.

B. Benchmarks for Self-Improvement

Benchmarking self-improvement is different because the evaluated agent changes as it interacts with the environment, so the relevant quantity is how capability changes as experience accumulates. There is no shared benchmark, and Table IV therefore compares what representative systems have demonstrated, separately from what they are designed to do; for the agent-directed class, we also coded the evaluations of every such system in the corpus. Four observations follow.

Persistent modification is not yet continual improvement. RoboCat ran one improvement iteration, and Eureka, CaP-X, Act-Observe-Rewrite, and ASPIRE produce one artifact per task. Re-testing earlier capabilities is rare. Of the 17 agent-directed systems, only a dexterous-grasping heuristic learner re-runs earlier regression tests before it promotes a change [99]; 3 others, including RHO, re-test afterwards, and 13 never do. Of the fixed-procedure systems in Table IV, only RoboCat reports re-testing earlier tasks, and only afterwards.

Transfer is tested unevenly. RHO and ASPIRE evaluate their improved artifacts on configurations or tasks held out from improvement, as do only 7 of the 17 agent-directed systems. RoboCat’s evidence of transfer is confounded by added training data, and ENPIRE and Self-Improving EFM report success on the tasks they practised, measured while the policy is still changing.

Humans remain in the loop. Every system in Table IV depends on designer-built evaluators, prompts, or interfaces, and the real-robot systems need people for task setup, resets, or review of the reset and reward code. Only 8 of the 17 agent-directed systems report real-robot results, and only HARBOR is compared with fixed-procedure loops under the same budget: with the same FM and wall-clock time, it designed better rewards than Eureka and REvolve in simulation, but it also received human heuristics that the baselines lacked [100]. RHO’s closest baselines are a single-shot generation and a restricted single-file variant [33]. The evidence so far therefore shows that agent-directed improvement works on these tasks, not that its delegated decisions improve more per unit of robot time than a fixed procedure.

One role label hides different systems. Maestro, CaP-X, Act-Observe-Rewrite, and RHO are all coding agents. Maestro and CaP-X call the FM on every trial [30], [90]. Act-Observe-Rewrite and RHO deploy frozen code, but Act-Observe-Rewrite rewrites its policy from a fixed outcome package and adopts every version that compiles, whereas RHO’s agent tests its own edits, a fixed gate decides, and the result is tested on held-out configurations [33], [89].

A useful protocol should evaluate performance as a function of an improvement budget,

$$S(b) = \text{Eval}(\psi(b), Q_{\text{test}}), \quad (6)$$

where $\psi(b)$ is the persistent artifact obtained after improvement budget b . Three task sets must be kept apart: development tasks that generate experience, a validation set Q_{val} on which candidates are compared and promoted, and a test set Q_{test} used only for the reported measurement; because an improvement loop makes many promotion decisions, a score on Q_{val} is selection-biased. The budget may count physical episodes, robot-hours, human interventions, wall-clock time, or simulation interactions, which separates systems that reach similar performance with different amounts of experience.

Self-improvement benchmarks should also report improvement efficiency, physical and human interaction cost, transfer to held-out tasks, regression on earlier capabilities, and which improvement decisions the FM made (Table III), so that systems with different amounts of delegated authority can be compared under the same budget.

VII. OPEN CHALLENGES TOWARD CONTINUALLY SELF-IMPROVING ROBOTS

Continual self-improvement asks the robot to collect useful experience, evaluate outcomes, decide what to change, validate the change, and keep it without degrading existing capabilities. Current systems cover parts of this cycle (Table IV).

Bridging semantic reasoning and embodiment. FMs can reason over far more behaviors than a given robot can realize.

TABLE IV

WHAT REPRESENTATIVE SELF-IMPROVING SYSTEMS HAVE DEMONSTRATED, FROM THE FULL TEXTS (NR: NOT REPORTED). HELD-OUT TEST: EVALUATION ON TASKS, CONFIGURATIONS, OR ENVIRONMENTS NOT USED DURING IMPROVEMENT; REGRESSION TEST: RE-TESTING OF EARLIER TASKS OR CAPABILITIES AFTER AN UPDATE.

System (type)	Setting and scale	Improvement budget	Human role	Held-out test	Regression test	At test time
RoboCat [18] (I_{fixed})	real; 11 task variants, 100 episodes each	one iteration, about 112k self-generated episodes	chose tasks, teleoperated demos, started and stopped collection	partial: better fine-tuning on unseen tasks, confounded by added data	post hoc, on all training tasks	frozen FM policy
Self-Improving EFM [85] (I_{fixed})	sim and real	about 20 h per real run on 3–4 stations	one operator monitored and reset robots; no labels	no: new tasks are practised, not held out	no; over-training lowers success	FM policy still changing
Eureka [23] (I_{fixed})	LanguageTable; sim Aloha sim; 29 tasks	up to 400 reward candidates per task, one PPO run each	task description and fitness function	no: same task, new training seeds	no	frozen reward; no FM
CaP-X [90] (I_{fixed})	sim; 7 tasks $\times$ 100 trials, plus LIBERO-PRO and BEHAVIOR	one synthesis pass over successful rollouts	APIs, prompts, and examples	no for the 7 source tasks; use elsewhere NR	NR	FM writes code every trial; library frozen
Act-Observe-Rewrite [89] (I_{fixed})	sim; 3 tasks	2–20 rewrites per task, one trial each	prompt and safety layer; simulator resets	no: repeats of the same tasks	no; worse versions were adopted	frozen code; no FM
RHO [33] (I_{agent})	sim; 700 Robosuite and 3,000 LIBERO-PRO trials; 21 RAI tasks	200 generations, 79 h, about \$2.8k of tokens (Robosuite)	evaluator, data splits, one manual restart	yes: new initial configurations, unseen perturbations, and unseen tasks	partial: full validation after a two-instance gate	frozen code; no LLM except in RAI
ASPIRE [32] (I_{agent})	sim LIBERO-PRO, Robosuite, BEHAVIOR; real 3 tasks $\times$ 20 trials	10–25 debug seeds per task, at most 3 replays each; compute NR	prompts, APIs, initial skills; real resets NR	yes: held-out seeds; unseen tasks zero-shot	no	frozen program per task, no FM (LIBERO-PRO, Robosuite)
ENPIRE [24] (I_{agent})	real Push-T, pin and GPU insertion, zip-tie cutting; sim RoboCasa	0.7–5 h wall-clock; 80–209M tokens (Push-T)	human-reviewed reset and reward setup; a few demos	no: same task and reset distribution	NR	success measured while still improving

Fixed skill libraries close this gap by construction, while code, geometry, and richer APIs expose more control [9], [10], [20], [30]; performance then depends on the human-designed interface, as coding-agent success in CaP-X does on API abstraction [90]. The interface itself should become adaptive, so that agents synthesize, validate, and reuse new skills when the existing ones fall short [32], [42].

Reliable automatic verification in the physical world. A robot can improve itself only if it can tell whether a change helped. Simulation gives ground truth cheaply [23]; physical systems must infer success, progress, and failure causes from noisy observations. Learned progress models and task-specific checks automate part of this [24], [85], but model-based evaluators can be biased or exploited. Physical critics that give calibrated outcome, progress, and diagnostic feedback, and ask for help when unsure [27], remain a central bottleneck.

Autonomous and sample-efficient physical experimentation. Real experiments need reset, calibration, logging, and safety checks that people still perform; ASPIRE names reset, success detection, and calibration as its main limits, and ENPIRE requires automatic evaluation and reset [24], [32]. Because physical interaction is scarce, systems should choose the most informative experiments, using simulation, generated environments, or world models to discard poor candidates [77], [101].

Optimizing and consolidating self-improving agents. Agentic systems modify discrete artifacts—programs, prompts, tools, repositories, and parts of the learning procedure [24], [32], [33]—for which physical evaluation gives no gradient. Iterative repair and evolutionary search grow expensive as components accumulate. Better formulations should propose, compare, merge, and prune changes, and decide whether new experience stays as memory, edits a component, becomes a skill, or is distilled into a policy.

When is more FM authority worth it? Delegating more improvement decisions to the FM is not progress in itself. Each delegated decision widens what the system can fix, but also what must be evaluated, debugged, and secured: in a preliminary RHO run, an agent terminated the operator’s session when told to stop hard-coding solutions [33]. Fixed interfaces, evaluators, and acceptance rules keep failures attributable and evaluation cheap, and are often the right design. The open question is empirical: under an equal robot and compute budget, which delegated decisions (Table III) improve held-out performance enough to pay for their evaluation and safety cost? The only equal-budget comparison among agent-directed systems, HARBOR’s, ran in simulation on the tasks being tuned and gave the agent extra human heuristics, so it cannot answer this question (Sec. VI).

Safe promotion, audit, and rollback. A candidate may improve its development tasks while weakening earlier skills or violating constraints elsewhere, which task-time safety filters cannot catch. Improvement systems should separate experimentation from deployment, compare candidates on Q_{val} , and require regression and safety gates before release; versioned artifacts, immutable traces, and a tested rollback path let failures be attributed and a deployed agent return to a known-safe state. The challenge is to keep these controls reliable under continual, autonomous operation.

VIII. CONCLUSION

FMs now make decisions that engineers used to fix before deployment, and their participation is extending from task reasoning to physical action generation and persistent improvement. Classifying systems by what the FM outputs and which improvement decisions it makes shows what they can change about their own behavior. How much of continual

self-improvement to delegate to the FM should follow evidence of what the extra freedom buys.

FUNDING

This research received no specific funding.

CONFLICTS OF INTEREST

The authors declare no conflicts of interest.

REFERENCES

- [1] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," in *Proceedings of the 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 205. PMLR, 2022, pp. 1769–1782. [Online]. Available: <https://proceedings.mlr.press/v205/huang23c.html>
- [2] Z. Liu, A. Bahety, and S. Song, "REFLECT: Summarizing robot experiences for failure explanation and correction," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 3468–3484. [Online]. Available: <https://proceedings.mlr.press/v229/liu23g.html>
- [3] C. R. Garrett, R. Chitnis, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.
- [4] S. Ross, G. J. Gordon, and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 15. PMLR, 2011, pp. 627–635.
- [5] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.
- [6] B. Zitkovich, T. Yu, S. Xu *et al.*, "RT-2: Vision-language-action models transfer web knowledge to robotic control," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 2165–2183. [Online]. Available: <https://proceedings.mlr.press/v229/zitkovich23a.html>
- [7] M. J. Kim, K. Pertsch, S. Karamcheti *et al.*, "OpenVLA: An open-source vision-language-action model," in *Proceedings of the 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270. PMLR, 2024, pp. 2679–2713. [Online]. Available: <https://proceedings.mlr.press/v270/kim25c.html>
- [8] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, L. Smith, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, " π_0 : A vision-language-action flow model for general robot control," in *Robotics: Science and Systems*, 2025. [Online]. Available: <https://www.roboticsproceedings.org/rss21/p010.html>
- [9] M. Ahn, A. Brohan, N. Brown *et al.*, "Do as I can, not as I say: Grounding language in robotic affordances," in *Proceedings of the 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 205. PMLR, 2022, pp. 287–318. [Online]. Available: <https://proceedings.mlr.press/v205/ichter23a.html>
- [10] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 9493–9500.
- [11] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen, "A survey on large language model based autonomous agents," *Frontiers of Computer Science*, vol. 18, no. 6, p. 186345, 2024.
- [12] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Qin, Y. Zheng, X. Qiu, X. Huang, Q. Zhang, and T. Gui, "The rise and potential of large language model based agents: A survey," *Science China Information Sciences*, vol. 68, no. 2, p. 121101, 2025.
- [13] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, "Cognitive architectures for language agents," *Transactions on Machine Learning Research*, 2024.
- [14] A. Psiris, V. Argyriou, E. K. Markakis, P. Sarigiannidis, E. Gavves, K. Bekris, A. Ajoudani, and G. T. Papadopoulos, "Foundation models in robotics: A comprehensive review of methods, models, datasets, challenges and future research directions," *Transactions on Machine Learning Research*, 2026.
- [15] W. Liang, R. Zhou, Y. Ma, B. Zhang, S. Li, Y. Liao, and P. Kuang, "Large model empowered embodied ai: A survey on decision-making and embodied learning," 2025.
- [16] S. Salimpour, L. Fu, K. Rachwał, P. Bertrand, K. O'Sullivan, R. Jakob, F. Keramat, L. Militano, G. Toffetti, H. Edelman, and J. Peña Queralta, "Towards embodied agentic ai: Review and classification of llm- and vlm-driven robot autonomy and interaction," 2025.
- [17] Z. Ren, Y. Chen, D. Guo, G. Rong, T. Li, R. B. Xiong, Q. Lan, W. Wang, N. Li, Y. Yang, M. Zhuge, and J. Schmidhuber, "Self-improvements in modern agentic systems: A survey," 2026.
- [18] K. Bousmalis, G. Vezzani, D. Rao, C. Devin, A. X. Lee, M. Bauzá, T. Davchev, Y. Zhou, A. Gupta, A. Raju *et al.*, "RoboCat: A self-improving generalist agent for robotic manipulation," *Transactions on Machine Learning Research*, 2024. [Online]. Available: <https://openreview.net/forum?id=vsCpLiWHu>
- [19] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "VoxPoser: Composable 3D value maps for robotic manipulation with language models," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 540–562. [Online]. Available: <https://proceedings.mlr.press/v229/huang23b.html>
- [20] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "ReKep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation," in *Proceedings of the 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270, 2024, pp. 4573–4602.
- [21] T. Yoneda, J. Fang, P. Li, H. Zhang, T. Jiang, S. Lin, B. Picker, D. Yunis, H. Mei, and M. R. Walter, "Statler: State-maintaining language models for embodied reasoning," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 15 083–15 091.
- [22] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, Q. Vuong, T. Zhang, T.-W. E. Lee, K.-H. Lee, P. Xu, S. Kirmani, Y. Zhu, A. Zeng, K. Hausman, N. Heess, C. Finn, S. Levine, and B. Ichter, "PIVOT: Iterative visual prompting elicits actionable knowledge for VLMs," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235. PMLR, 2024, pp. 37 321–37 341. [Online]. Available: <https://proceedings.mlr.press/v235/nasiriany24a.html>
- [23] Y. J. Ma, W. Liang, G. Wang *et al.*, "Eureka: Human-level reward design via coding large language models," in *International Conference on Learning Representations*, 2024.
- [24] W. Xiao, J. Xie, T. Zhang *et al.*, "ENPIRE: Agentic robot policy self-improvement in the real world," in *Conference on Robot Learning*, 2026.
- [25] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 162. PMLR, 2022, pp. 9118–9147. [Online]. Available: <https://proceedings.mlr.press/v162/huang22a.html>
- [26] I. Singh, V. Blukis, A. Mousavian, A. Goyal, D. Xu, J. Tremblay, D. Fox, J. Thomason, and A. Garg, "ProgPrompt: Generating situated robot task plans using large language models," in *IEEE International Conference on Robotics and Automation*, 2023, pp. 11 523–11 530.
- [27] A. Z. Ren, A. Dixit, A. Bodrova, S. Singh, S. Tu, N. Brown, P. Xu, L. Takayama, F. Xia, J. Varley, Z. Xu, D. Sadigh, A. Zeng, and A. Majumdar, "Robots that ask for help: Uncertainty alignment for large language model planners," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 661–682. [Online]. Available: <https://proceedings.mlr.press/v229/ren23a.html>
- [28] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Sünderhauf, "SayPlan: Grounding large language models using 3D scene graphs for scalable robot task planning," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 23–72. [Online]. Available: <https://proceedings.mlr.press/v229/rana23a.html>

- [29] K. Fang, F. Liu, P. Abbeel, and S. Levine, "MOKA: Open-world robotic manipulation through mark-based visual prompting," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, Jul. 2024.
- [30] J. Shi, R. Yang, K. Chao, S. B. Wan, Y. Shao, J. Lei, J. Qian, L. Le, P. Chaudhari, K. Daniilidis, C. Wen, and D. Jayaraman, "MAESTRO: Orchestrating robotics modules with vision-language models for zero-shot generalist robots," in *NeurIPS Workshop on Space in Vision, Language, and Embodied AI*, 2025. [Online]. Available: <https://maestro-robot.github.io/>
- [31] H. Ha, P. Florence, and S. Song, "Scaling up and distilling down: Language-guided robot skill acquisition," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 3766–3777. [Online]. Available: <https://proceedings.mlr.press/v229/ha23a.html>
- [32] R. Lu, Y. Wu, E. Kou *et al.*, "ASPIRE: Agentic skills discovery for robotics," *arXiv preprint arXiv:2607.00272*, 2026.
- [33] K. Elmaaroufi, J. Svegliato, S. Kalade, G. Schelle, S. A. Seshia, and M. Zaharia, "RHO: Your coding agent is secretly a roboticist," *arXiv preprint arXiv:2606.16458*, 2026.
- [34] O. Mees, J. Borja-Diaz, and W. Burgard, "Grounding language with visual affordances over unstructured data," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 11 576–11 582.
- [35] D. Shah, B. Osifski, B. Ichter, and S. Levine, "LM-Nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 205. PMLR, 2022, pp. 492–504.
- [36] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 6695–6702.
- [37] J. Gao, B. Sarkar, F. Xia, T. Xiao, J. Wu, B. Ichter, A. Majumdar, and D. Sadigh, "Physically grounded vision-language models for robotic manipulation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 12 462–12 469.
- [38] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min, "SMART-LLM: Smart multi-agent robot task planning using large language models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 12 140–12 147.
- [39] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 608–10 615.
- [40] Y. Ding, X. Zhang, C. Paxton, and S. Zhang, "Task and motion planning with large language models for object rearrangement," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 2086–2092.
- [41] H. You, Y. Ye, T. Zhou, Q. Zhu, and J. Du, "Robot-enabled construction assembly with automated sequence planning based on ChatGPT: RoboGPT," *Buildings*, vol. 13, no. 7, p. 1772, 2023.
- [42] S. Xie, Y. Zhang, R. Wang, and X. Chen, "Uni-Skill: Building self-evolving skill repository for generalizable robotic manipulation," in *IEEE International Conference on Robotics and Automation*, 2026.
- [43] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary queryable scene representations for real world planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 509–11 522.
- [44] N. Yoshikawa, M. Skreta, K. Darvish, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, A. Z. Li, Y. Zhao, H. Xu, A. Kuramshin, A. Aspuru-Guzik, F. Shkurti, and A. Garg, "Large language models for chemistry robotics," *Autonomous Robots*, vol. 47, no. 8, pp. 1057–1086, 2023.
- [45] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "ConceptGraphs: Open-vocabulary 3D scene graphs for perception and planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5021–5028.
- [46] B. Yu, H. Kasaei, and M. Cao, "L3MVN: Leveraging large language models for visual target navigation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 3554–3560.
- [47] D. Driess, F. Xia, M. S. M. Sajjadi *et al.*, "PaLM-E: An embodied multimodal language model," in *Proceedings of the 40th International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 202. PMLR, 2023, pp. 8469–8488.
- [48] K. Lin, C. Agia, T. Migimatsu, M. Pavone, and J. Bohg, "Text2Motion: From natural language instructions to feasible plans," *Autonomous Robots*, vol. 47, no. 8, pp. 1345–1365, 2023.
- [49] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser, "TidyBot: Personalized robot assistance with large language models," *Autonomous Robots*, vol. 47, no. 8, pp. 1087–1102, 2023.
- [50] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. Gonzalez Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, B. Ichter, T. Xiao, P. Xu, A. Zeng, T. Zhang, N. Heess, D. Sadigh, J. Tan, Y. Tassa, and F. Xia, "Language to rewards for robotic skill synthesis," in *Proceedings of the 7th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 229. PMLR, 2023, pp. 374–404. [Online]. Available: <https://proceedings.mlr.press/v229/yu23a.html>
- [51] D. Song, J. Liang, A. Payandeh, A. H. Raj, X. Xiao, and D. Manocha, "VLM-Social-Nav: Socially aware robot navigation through scoring using vision-language models," *IEEE Robotics and Automation Letters*, vol. 10, no. 1, pp. 508–515, 2025.
- [52] D. Goetting, H. G. Singh, and A. Loquercio, "End-to-end navigation with vision-language models: Transforming spatial reasoning into question-answering," in *Proceedings of the International Conference on Neuro-symbolic Systems*, ser. Proceedings of Machine Learning Research, vol. 288. PMLR, 2025, pp. 22–35. [Online]. Available: <https://proceedings.mlr.press/v288/goetting25a.html>
- [53] R. Mon-Williams, G. Li, R. Long, W. Du, and C. G. Lucas, "Embodied large language models enable robots to complete complex tasks in unpredictable environments," *Nature Machine Intelligence*, vol. 7, no. 4, pp. 592–601, 2025.
- [54] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "LLM-Planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 2986–2997.
- [55] Y. Guo, Y.-J. Wang, L. Zha, and J. Chen, "DoReMi: Grounding language model by detecting and recovering from plan-execution misalignment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2024, pp. 12 124–12 131.
- [56] T. Song, M. Luo, X. Zhang *et al.*, "A multiagent-driven robotic AI chemist enabling autonomous chemical research on demand," *Journal of the American Chemical Society*, vol. 147, no. 15, pp. 12 534–12 545, 2025.
- [57] P. Zhi, Z. Zhang, Y. Zhao, M. Han, Z. Zhang, Z. Li, Z. Jiao, B. Jia, and S. Huang, "Closed-loop open-vocabulary mobile manipulation with GPT-4V," in *IEEE International Conference on Robotics and Automation*, 2025, pp. 4761–4767.
- [58] K. Darvish, M. Skreta, Y. Zhao, N. Yoshikawa, S. Som, M. Bogdanovic, Y. Cao, H. Hao, H. Xu, A. Aspuru-Guzik, A. Garg, and F. Shkurti, "ORGANA: A robotic assistant for automated chemistry experimentation and characterization," *Matter*, vol. 8, no. 2, p. 101897, 2025.
- [59] I. T. Lei, Q. Li, Y. J. Yap, Y. Zhang, R. Zhong, H. Gui, X. Liu, and Z. Deng, "ACE: Agentic control for embodied manipulation via zero-shot workflow reasoning," *arXiv preprint arXiv:2607.04162*, 2026.
- [60] G. Zhou, Y. Hong, and Q. Wu, "NavGPT: Explicit reasoning in vision-and-language navigation with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 7, 2024, pp. 7641–7649.
- [61] Z. Mandi, S. Jain, and S. Song, "RoCo: Dialectic multi-robot collaboration with large language models," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 286–299.
- [62] Z. Yang, Y. Chen, X. Zhou, J. Yan, D. Song, Y. Liu, Y. Li, Y. Zhang, P. Zhou, H. Chen, and L. Sun, "Agentic robot: A brain-inspired framework for vision-language-action models in embodied agents," *arXiv preprint arXiv:2505.23450*, 2025. [Online]. Available: <https://arxiv.org/abs/2505.23450>
- [63] B. Lange, A. Yildiz, M. Arief, S. Khattak, M. Kochenderfer, and G. Georgakis, "General-purpose robotic navigation via LVLm-orchestrated perception, reasoning, and acting," *arXiv preprint arXiv:2506.17462*, 2025. [Online]. Available: <https://arxiv.org/abs/2506.17462>
- [64] Y. Yang, K. Gu, Y. Wen, H. Li, Y. Zhao, T. Wang, and X. Liu, "ManiAgent: An agentic framework for general robotic manipulation," *arXiv preprint arXiv:2510.11660*, 2025. [Online]. Available: <https://arxiv.org/abs/2510.11660>
- [65] K. Rachwał, M. Majek, B. Boczek *et al.*, "RAI: Flexible agent framework for embodied AI," in *Highlights in Practical Applications of Agents, Multi-Agent Systems and Computational Social Science. The PAAMS Collection*, ser. Communications in Computer and Information Science, vol. 2644. Springer, 2025, pp. 195–206.

- [66] S. Ahn, W. Choi, J. Lee, J. Park, and H. Woo, "Towards reliable code-as-policies: A neuro-symbolic framework for embodied task planning," in *Advances in Neural Information Processing Systems*, 2025.
- [67] V. G. dos Santos, I. Khadraoui, I. Farhat *et al.*, "ALRM: Agentic llm for robotic manipulation," *arXiv preprint arXiv:2601.19510*, 2026.
- [68] Z. Zhu, L. Guo, Y. Mei, B. Pang, Z. Zhang, X. He, R. Ji, and J. Liu, "When robots do the chores: A benchmark and agent for long-horizon household task execution," *arXiv preprint arXiv:2605.14504*, 2026.
- [69] H. B. Amor, L. Graesser, A. Iscen, D. D'Ambrosio, S. Abeyruwan, A. Bewley, Y. Zhou, K. Kalirathinam, S. Mishra, and P. Sanketi, "SAS-Prompt: Large language models as numerical optimizers for robot self-improvement," in *IEEE International Conference on Robotics and Automation*, 2025, pp. 10087–10094.
- [70] Z. Zheng, O. Zhang, H. L. Nguyen, N. Rampal, A. H. Alawadhi, Z. Rong, T. Head-Gordon, C. Borgs, J. T. Chayes, and O. M. Yaghi, "ChatGPT research group for optimizing the crystallinity of MOFs and COFs," *ACS Central Science*, vol. 9, no. 11, pp. 2161–2170, 2023.
- [71] L. Zha, Y. Cui, L.-H. Lin, M. Kwon, M. Gonzalez Arenas, A. Zeng, F. Xia, and D. Sadigh, "Distilling and retrieving generalizable knowledge for robot manipulation via language corrections," in *IEEE International Conference on Robotics and Automation*, 2024, pp. 15 172–15 179.
- [72] L. Bärmann, R. Kartmann, F. Peller-Konrad, J. Niehues, A. Waibel, and T. Asfour, "Incremental learning of humanoid robot behavior from natural interaction and large language models," *Frontiers in Robotics and AI*, vol. 11, p. 1455375, 2024.
- [73] R. Shah, A. Yu, Y. Zhu, Y. Zhu, and R. Martín-Martín, "BUMBLE: Unifying reasoning and acting with vision-language models for building-wide mobile manipulation," in *IEEE International Conference on Robotics and Automation*, 2025, pp. 13 337–13 345.
- [74] K. Qu, G. Lan, R. Zurbrügge, C. Chen, C. E. Mower, H. Bou-Ammar, and M. Hutter, "A pragmatist robot: Learning to plan tasks by experiencing the real world," *IEEE Robotics and Automation Letters*, vol. 11, no. 5, pp. 5606–5613, 2026.
- [75] P. Katara, Z. Xian, and K. Fragkiadaki, "Gen2Sim: Scaling up robot learning in simulation with generative models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 6672–6679.
- [76] M. Parakh, A. Fong, A. Simeonov, T. Chen, A. Gupta, and P. Agrawal, "Lifelong robot learning with human assisted language planners," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 523–529.
- [77] Y. Wang, Z. Xian, F. Chen *et al.*, "RoboGen: Towards unleashing infinite data for automated robot learning via generative simulation," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, 2024, pp. 51 936–51 983.
- [78] X. Zhao, C. Weber, and S. Wermter, "Agentic skill discovery," *Robotics and Autonomous Systems*, vol. 196, p. 105248, 2026.
- [79] M. Han, Y. Zhu, S.-C. Zhu, Y. N. Wu, and Y. Zhu, "InterPreT: Interactive predicate learning from language feedback for generalizable task planning," in *Proceedings of Robotics: Science and Systems (RSS)*, Delft, Netherlands, 2024.
- [80] G. Chen, L. Yang, R. Jia, Z. Hu, Y. Chen, W. Zhang, W. Wang, and J. Pan, "Language-augmented symbolic planner for open-world task planning," in *Proceedings of Robotics: Science and Systems (RSS)*, Delft, Netherlands, 2024.
- [81] J. Lehman, J. Gordon, S. Jain, K. Ndousse, C. Yeh, and K. O. Stanley, "Evolution through large models," in *Handbook of Evolutionary Machine Learning*, W. Banzhaf, P. Machado, and M. Zhang, Eds. Singapore: Springer, 2024, pp. 331–366.
- [82] C. Xiong, C. Shen, X. Li, K. Zhou, J. Liu, R. Wang, and H. Dong, "Autonomous interactive correction MLLM for robust robotic manipulation," in *Proceedings of the 8th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 270. PMLR, 2024, pp. 3139–3156. [Online]. Available: <https://proceedings.mlr.press/v270/xiong25a.html>
- [83] S. Sun, C. Li, Z. Zhao, H. Huang, and W. Xu, "Leveraging large language models for comprehensive locomotion control in humanoid robots design," *Biomimetic Intelligence and Robotics*, vol. 4, no. 4, p. 100187, 2024.
- [84] G. Zhu, R. Zhou, W. Ji, and S. Zhao, "LAMARL: LLM-aided multi-agent reinforcement learning for cooperative policy generation," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 7476–7483, 2025.
- [85] S. K. S. Ghasemipour, A. Wahid, J. Tompson, P. Sanketi, and I. Mordatch, "Self-improving embodied foundation models," in *Advances in Neural Information Processing Systems*, 2025.
- [86] G. J. Gao, T. Li, J. Shi, Y. Li, Z. Zhang, N. Figueroa, and D. Jayaraman, "VLMgineer: Vision-language models as robotic toolsmiths," in *International Conference on Learning Representations*, 2026.
- [87] W. Chen, G. Li, M. Li, W. Wang, P. Li, X. Xue, X. Zhao, and L. Liu, "LLM-enabled incremental learning framework for hand exoskeleton control," *IEEE Transactions on Automation Science and Engineering*, vol. 22, pp. 2617–2626, 2025.
- [88] S. Yang, Y. Du, S. K. S. Ghasemipour, J. Tompson, L. P. Kaelbling, D. Schuurmans, and P. Abbeel, "Learning interactive real-world simulators," in *International Conference on Learning Representations*, 2024.
- [89] V. Kumar, "Act-observe-rewrite: Multimodal coding agents as in-context policy learners for robot manipulation," *arXiv preprint arXiv:2603.04466*, 2026.
- [90] L. Fu, J. Yu, K. El-Refai, E. Kou, H. Xue *et al.*, "CaP-X: A framework for benchmarking and improving coding agents for robot manipulation," in *Proceedings of the 43rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 306, 2026.
- [91] Y. J. Ma, W. Liang, H.-J. Wang, Y. Zhu, L. Fan, O. Bastani, and D. Jayaraman, "DrEureka: Language model guided sim-to-real transfer," in *Robotics: Science and Systems*, 2024.
- [92] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "VirtualHome: Simulating household activities via programs," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8494–8502.
- [93] M. Li, S. Zhao, Q. Wang, K. Wang, Y. Zhou, S. Srivastava, C. Gokmen, T. Lee, L. E. Li, R. Zhang, W. Liu, P. Liang, L. Fei-Fei, J. Mao, and J. Wu, "Embodied agent interface: Benchmarking LLMs for embodied decision making," in *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, vol. 37, 2024.
- [94] S. Zhang, P. Wicke, L. K. Şenel, L. Figueredo, A. Naceri, S. Haddadin, B. Plank, and H. Schütze, "LoHoRavens: A long-horizon language-conditioned benchmark for robotic tabletop manipulation," *arXiv preprint arXiv:2310.12020*, 2023.
- [95] R. Yang, H. Chen, J. Zhang, M. Zhao, C. Qian, K. Wang, Q. Wang, T. V. Koripella, M. Movahedi, M. Li, H. Ji, H. Zhang, and T. Zhang, "EmbodiedBench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents," in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267, 2025, pp. 70 576–70 631.
- [96] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.
- [97] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard, "CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7327–7334, 2022.
- [98] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," in *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, vol. 36, 2023.
- [99] E. H. Teo, "From false positives to failure topologies: Heuristic learning for auditable dexterous grasping development," Zenodo preprint, 2026.
- [100] Z. Li, Y. Jin, X. Liu, P. Liu, V. Prasad, C. D'Eramo, and G. Chalkatzaki, "HARBOR: A harness framework for agentic robot reinforcement learning," *arXiv preprint arXiv:2606.08610*, 2026. [Online]. Available: <https://arxiv.org/abs/2606.08610>
- [101] L. Wang, Y. Ling, Z. Yuan, M. Shridhar, C. Bao, Y. Qin, B. Wang, H. Xu, and X. Wang, "GenSim: Generating robotic simulation tasks via large language models," in *International Conference on Learning Representations*, 2024.
- [102] M. J. Page, J. E. McKenzie, P. M. Bossuyt *et al.*, "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," *BMJ*, vol. 372, p. n71, 2021.
- [103] B. Zhang and H. Soh, "Large language models as zero-shot human models for human-robot interaction," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 7961–7968.
- [104] N. Kachaev, A. Spiridonov, A. Gorodetsky, K. Muravyev, N. Oskolkov, A. Narendran, V. Shakhuro, D. Makarov, A. I. Panov, P. Fedotova, and A. K. Kovalev, "Mind and motion aligned: A joint evaluation IsaacSim benchmark for task planning and low-level policies in mobile manipulation," *Doklady Mathematics*, vol. 112, no. 2, pp. 400–408, 2025.

- [105] X. Zhou, Y. Xu, G. Tie, Y. Chen, G. Zhang, D. Chu, P. Zhou, and L. Sun, "LIBERO-PRO: Towards robust and fair evaluation of vision-language-action models beyond memorization," *arXiv preprint arXiv:2510.03827*, 2025.
- [106] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, K. Lin, A. Maddukuri, S. Nasiriany, and Y. Zhu, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.
- [107] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun *et al.*, "BEHAVIOR-1K: A benchmark for embodied AI with 1,000 everyday activities and realistic simulation," in *Proceedings of the 6th Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, vol. 205, 2022, pp. 80–93.
- [108] Z. Zhang, S. Yan, M. Han, Z. Wang, X. Wang, S.-C. Zhu, and H. Liu, " M^3 Bench: Benchmarking whole-body motion generation for mobile manipulation in 3D scenes," *IEEE Robotics and Automation Letters*, vol. 10, no. 7, pp. 7286–7293, 2025.
- [109] S. Nasiriany, S. Nasiriany, A. Maddukuri, and Y. Zhu, "RoboCasa365: A large-scale simulation framework for training and benchmarking generalist robots," in *International Conference on Learning Representations*, 2026.

APPENDIX A SURVEY METHODOLOGY

We conducted a systematic search and screening following the PRISMA 2020 reporting guideline [102]. All queries, retrieved records, screening decisions with reasons, and analysis scripts will be released with the paper (directory `lit_search`).

A. Sources, Window, and Search Strings

Sources. We searched arXiv, where nearly all robot-learning and FM-for-robotics work appears as a preprint, and OpenAlex, which indexes journal articles and conference proceedings across publishers (IEEE, ACM, Springer, Elsevier, PMLR, and others) and therefore captures venue-only papers. The publication window was 1 January 2022 to 23 September 2026; the searches were executed on 23–24 September 2026.

Query. The search string mirrors the scope of the survey (Sec. I) and combines three blocks over titles and abstracts:

- *FM block*: “language model(s)”, LLM(s), VLM(s), “vision-language”, “foundation model(s)”, MLLM, GPT, “coding agent(s)”;
- *robot block*: robot, robots, robotic (on arXiv also the category cs.RO);
- *agency/decision block*: agent, agentic, planning, planner, skill, code, program, reward, feedback, correction, recovery, replanning, verification, “self-improvement”, “self-improving”.

The primary query is $FM \wedge robot \wedge agency$. A recall check against the works covered by the survey showed that motion-level papers describe their outputs as keypoints, affordances, trajectories, or constraints and that recent agent-directed systems describe the FM only as a “coding agent” or “agentic” system. We therefore added three supplementary blocks: $FM \wedge robot \wedge (\text{motion, trajectory/ies, keypoint(s), affordance(s), constraint(s)})$ with “foundation” added to the FM block; $(\text{agentic} \vee \text{“coding agent(s)”}) \wedge robot$; and $\text{foundation} \wedge robot \wedge agency$. Because the arXiv API rejects long Boolean expressions, each arXiv query was executed as a grid of short sub-queries whose union is equivalent (15,233 retrieved hits, 11,871 unique records). On OpenAlex (9,360 unique records) only the primary query was run: its anonymous API was rate-limited during the search, and its stemming conflates “agentic” with “agent”, which turns the agentic block into a generic agent-and-robot query dominated by pre-FM multi-agent robotics. The supplementary blocks were therefore run on arXiv only.

B. Deduplication, Query Verification, and Automated Rules

Records were deduplicated by arXiv identifier, DOI, and normalized title, removing 5,899 duplicates. Because the arXiv field search matches phrase terms loosely, we re-applied the exact Boolean query to each retrieved title and abstract and removed 7,335 records that did not satisfy it (query verification). Automated rules then removed 444 records that were non-English, non-research record types (errata, editorials, paratext, datasets), or secondary literature identified by title (survey, review, overview, tutorial, position, perspective, roadmap). This left 7,553 records for screening.

C. Eligibility Criteria

A study was eligible if it presents an implemented system or method in which a foundation model (LLM, VLM, MLLM, other large pretrained model, or an FM-based coding agent) participates in a goal-directed decision or persistent-adaptation loop that affects robot behavior, evaluated on a physical robot or in a physics-grounded embodied environment: a simulator built on a physics engine or on 3D scans of real buildings, as in navigation benchmarks (games and text-only worlds are excluded under EC4). Benchmarks designed to evaluate such agents were eligible as a separate category. Exclusion codes were EC1 perception-only use of FMs; EC2 standalone VLA or policy models, or their training recipes, without an agent-level decision loop; EC3 no FM-mediated decision; EC4 purely digital, game, or text-only agents; EC5 dialogue-only human–robot interaction; EC6 secondary literature or non-research records; EC7 other domains (e.g., autonomous-driving perception, non-robotic laboratories); and INS insufficient information (no abstract and an ambiguous title). EC2 implements the scope rule for robot foundation policies of Sec. I: a policy is in scope only when an FM decision steers it or when its own experience drives its update. After coding, a system stayed in the corpus only if its FM has at least one locus (Appendix A-E).

D. Screening Procedure and Audit

Screening had two stages and was performed by LLM reviewers (Claude models, September 2026) under the fixed written protocol above, with every decision logged. *Stage 1* (title and abstract lead, all 7,553 records) was deliberately liberal: a record was kept whenever an FM plausibly makes or revises robot-behavior decisions; 3,667 records were excluded. *Stage 2* (full abstract, 3,886 records) decided final eligibility and recorded a single exclusion code.

Three controls were applied. First, reviewers could only read records; every batch was checked for completeness against its last record, and records missing from an output were re-assessed. Second, an initial screening run in which reviewers had substituted keyword heuristics for per-record judgement was detected from its execution trace and discarded in full before the reported run. Third, a second reviewer, an independent LLM instance given the same protocol but not the first reviewer’s

decisions, audited random samples. Of 200 records excluded at stage 1, it would have retained 17: 5 concerned autonomous driving (out of scope under EC7), 10 were uncertain, and 2 were clearly eligible. The stage-1 false-negative rate is therefore between 1.0% (95% CI 0.3–3.6%) if only the clear cases are counted and 6.0% (3.5–10.2%) if all uncertain ones are. On 150 randomly sampled stage-2 decisions, the two reviewers agreed on eligibility for 129 of 150 records (86%, 95% CI 80–91%; Cohen’s $\kappa = 0.66$, 0.54–0.79), with the primary reviewer the more inclusive of the two. Agreement measures consistency, not accuracy; the accuracy of the labels used in the analysis is measured below against full texts.

E. Coding of Loci and Final Membership

Stage 2 had assigned provisional loci. Because review showed that these rules were not explicit enough, we wrote a coding protocol (released as `coding_protocol.md`, revision 2.2) that states the rules of Sec. IV: T and M from what the FM outputs, including the rule for scores versus decisions; the convention separating memory from improvement; the decisions D1–D5 that separate I_{fixed} from I_{agent} ; the scope rule for FM policies; final eligibility after coding; and the first-public-version year. Table A1 traces the corpus through the steps below.

TABLE A1
HOW THE FINAL CORPUS WAS FORMED.

Candidate systems after screening	2,704
<i>Left at adjudication of flagged and no-locus systems</i>	
FM only perceives, verifies, or decides nothing	−192
Standalone FM policy steered by non-FM modules	−105
Other criteria (not a robot, dialogue only, benchmark, ...)	−47
<i>Left after full-text reading</i>	
Only locus was an improvement loop that proved absent	−5
Full text needed to resolve the improvement type, unavailable	−4
Out of scope on full-text reading	−8
Final corpus	2,343
with agent-directed improvement (all from full text)	17

Screening corrections. Full-text reading of the synthesized systems reversed three screening decisions: the stage-1 exclusion of Self-Improving Embodied Foundation Models (its policy’s own experience drives its update), the stage-2 exclusion of LongAct as purely digital (it runs in the AI2-THOR physics simulator, like other eligible systems), and the typing of CaP-X as a benchmark only (it also proposes a system). This gives 2,704 candidate systems and 207 benchmarks.

Abstract-level coding. An LLM coder labelled every candidate from its title and abstract with T , M , and $I \in \{0, I_{\text{fixed}}, I_{\text{agent}}, \text{unclear}\}$, plus embodiment, evaluation setting, and a flag for cases that strain the rules. A second coder, run independently on the same protocol, recoded 320 systems: 240 drawn at random and 80 drawn from systems with provisional improvement loci, to have enough improvement cases. On the random part, Cohen’s κ was 0.96 for T (95% bootstrap CI 0.91–0.99), 0.90 for M (0.84–0.95), and 0.91 for the presence of improvement (0.84–0.97); among systems both coders marked as improving, the type agreed with $\kappa = 0.83$ ($n = 117$).

Adjudication. An adjudicator (a further LLM instance with the revision-2.2 protocol) decided every system the coder had flagged (760) and every system left without a locus, 764 in total, from title and abstract, with the first coder’s labels and reason as context. It removed 344: 192 whose FM only perceives, verifies, or makes no decision (EC1, EC3); 105 standalone FM policies steered only by non-FM modules (EC2); and 47 for other criteria. It changed at least one label of 100 of the 414 systems it kept.

Full-text resolution of improvement type. Whether an FM chooses its evidence or runs its own tests is rarely visible in an abstract. Every retained system whose abstract-level label was agent-directed or unclear (54) was therefore resolved from its full text: 16 are agent-directed, 26 fixed-procedure, and 8 have no persistent improvement; 4 whose full text could not be obtained were removed. Each agent-directed label was read twice, the second time while extracting the evaluation evidence of Sec. VI; where the readings disagreed, we decided from the quoted passages (six systems became fixed-procedure, for example because the FM only asked a human for a demonstration or a designer’s prompt prescribed the choice). Full-text coding of the synthesized systems led to one revision of the protocol: the target decision D1 was redefined over kinds of artifact (policy, reward, training procedure, data) because, defined over files, it made a label depend on how a program is split into files.

Missed agent-directed systems. To find agent-directed systems that abstract coding missed, we read 100 further systems in full: every fixed-procedure system whose abstract uses agentic terms (“agentic”, “coding agent”, “self-improving”, “self-evolving”, “harness”, or autonomous research), a random 60 of the other fixed-procedure systems, and a random 30 of the systems with agentic terms but no improvement label. This found 2 agent-directed systems, of which 1 had not already been marked for full-text reading (the other had been relabelled unclear at adjudication), and 5 systems whose only locus was an improvement loop that proved absent (removed). Extrapolating the rate in each stratum to its unchecked systems (Jeffreys posterior) gives about 5 more agent-directed systems (95% upper bound 28).

Accuracy against full texts. Both coders are instances of the same model and may share biases, so agreement alone does not establish accuracy. We coded from full text a random sample of 80 systems stratified by year (2023, 2026) and abstract-level M label (20 per cell; unavailable full texts were replaced from a pre-drawn reserve), and weighted cells by their size. For T , full-text estimates matched the abstract-level shares (78% in 2023 and 75% in 2026), and the share with persistent improvement (24%, 95% CI 8–42%, and 25%, 12–39%) was within sampling error of the abstract-level shares. For M , errors depended on the year. In 2023, 14/18 abstract-level M labels were confirmed and 2/17 systems without one had motion outputs; in 2026 the figures were 19/20 and 8/20. The misses were mostly metric skill parameters or a vision-language-action executor that the abstract does not mention. The full-text motion-level share is therefore 27% (95% CI 16–41%) in 2023 and 63% (50–77%) in 2026; the rise is larger than the abstract-level estimate, not an artifact of it. Because the error grows with year, the main text reports these corrected shares rather than the abstract-level ones, and likewise the corrected share of task-level-only systems: 51% (32–69%) in 2023 and 28% (14–42%) in 2026, against 60% and 48% at the abstract level. The sample also measured residual screening error: 5 of the sampled 2023 systems and 0 of the 2026 systems were out of scope on full-text reading and were removed, as were two reserve-list systems and one agent-directed system (LLM-IK) that is evaluated only by forward-kinematics arithmetic, with no robot or physics simulator. On the 78 synthesized systems, abstract labels matched full-text labels for 90% of systems on T , 81% on M , and 96% on the presence of improvement.

F. Trend Analysis

Table A2 gives, per year of first public version, the number of systems in the final corpus and the share at each locus; the window ends on 23 September 2026. Trends are tested with the Cochran–Armitage test over 2023–2026; 2022 is shown but not tested because it contains only 16 systems. The share of systems at the motion level rises from 21% (95% CI 15–28%) in 2023 to 43% (39–46%) in 2026 ($z = 5.4$, $p < 10^{-7}$); excluding every system flagged as borderline gives 23% and 46%, and correcting with the full-text sample gives 27% and 63% (difference 37 points, 95% CI 17–54), the values the main text uses. The share with persistent improvement shows no trend (14–21% per year, $p = 0.66$), whereas agent-directed improvement is concentrated in 2026: 16 of the 17 agent-directed systems appeared between 1 January and 23 September 2026, where they make up 10% of that year’s improving systems, against 1 of 241 in 2023–2025. With the provisional stage-2 loci, 2026 had shown 61 agent-directed systems; most of these turned out to be fixed-procedure loops or memory in the full text.

G. Synthesis and Coverage

The flow of records is shown in Fig. A1. Stage 2 excluded 975 reports (EC1 204, EC2 266, EC3 116, EC4 138, EC5 46, EC6 137, EC7 57, INS 11), leaving 2,911 eligible studies (2,704 candidate systems and 207 benchmarks).

Selection. The 78 synthesized systems come from the final corpus. They combine the 47 systems we had analyzed before the search (all retrieved and judged eligible, one only after its stage-1 exclusion was reversed on full-text reading) with 29 of 30 systems drawn by citation count from those not yet covered; the 30th, an LLM used only as a model of human behavior [103], has no locus and left the corpus. The citation counts in the search records were incomplete for arXiv-only records, so we checked the set against Semantic Scholar counts (25 September 2026) and added the two of the 20 most-cited systems it lacked, LLM-Planner and NavGPT [54], [60]. The set now contains all 20 and 33 of the 50 most-cited systems of the corpus; the other synthesized systems cover the loci and feedback regimes that the most-cited systems do not, including recent mechanisms. The benchmarks and evaluation substrates we cite were included through citation searching. Each synthesized system was coded from its full text; the evidence for every label will be released with the paper.

Coverage. Table A3 compares the synthesized set with the final corpus. Because it favors influential work, it over-represents systems first released in 2022–2023 (54% vs. 7% in the corpus) and under-represents 2025–2026 (27% vs. 74%). It also over-represents arms and mobile manipulators and systems evaluated both in simulation and on real robots, and under-represents navigation, multi-robot, and other embodiments. Its locus mix has fewer task-only and motion-only systems and, by design, more improvement systems. For these reasons, all trend statements in the paper are computed on the whole corpus, and the synthesized set is used to explain mechanisms and to check coding accuracy.

Flag reasons. The abstract-level coder flagged 760 systems (28%) as hard to classify, with a short reason. Grouped with keyword rules checked by hand, the reasons were: the FM only informs a decision made by a non-FM module (195); an FM policy is steered only by non-FM modules (75); the FM acts before deployment, generating data, tasks, or designs (90); memory, training-stage learning, and persistent improvement are hard to separate (83); the FM writes a formal specification that a solver plans with (26); missing or vague abstracts (133); doubtful eligibility, such as attack studies, theses, and non-robotic domains (93); and other (65). Sec. IV states how the rules settle each pattern; the adjudication above applied them.

Limitations. Screening, coding, and adjudication relied on LLM reviewers with audited but non-zero error rates, and the independent coders are instances of the same model. Corpus labels are abstract-level; the random full-text sample covers two years, so the size of the abstract-level error in 2024 and 2025 is not measured. The robot block misses embodied-AI papers that never mention robots (e.g., EmbodiedBench); the supplementary query blocks were run on arXiv only; and citation counts favor older work.

APPENDIX B

EXTENDED COMPARISON WITH EXISTING SURVEYS

Psiris et al. review *foundation models in robotics* from a systematic search of 438 articles and organize the field by FM type (LLM, VFM, VLM, and VLA), neural architecture, learning paradigm and stage, robotic task, and application domain [14]. Closed-loop execution, verification, and FM-guided reward design appear as subsections within these categories, and self-improvement is treated as a learning paradigm or stage. Liang et al. survey *large-model-empowered embodied AI* [15]. Their hierarchical decision-making section explicitly covers perception, high-level planning, low-level execution, and “feedback and enhancement” from self-reflection, humans, and the environment, alongside end-to-end VLA models, embodied learning, and world models. Their feedback discussion describes systems one by one; it does not ask what verifies outcomes or where feedback is routed.

The closest survey to ours is *Towards Embodied Agentic AI* [16]. Its two taxonomies characterize how models are integrated (protocol, interface, orchestration, or direct/embedded) and what role an agent plays (planner, orchestrator, task-specific, model-centric, or generalist). It separates planners whose outputs are grounded by lower-level controllers from models that output actions directly, contrasts AI workflows (predefined sequences of model and tool calls) with agents that decide their next action, lists memory and adaptation as system attributes, and names self-curated skill libraries (Voyager) and self-improving policies (RoboCat) as examples of capability growth. It does not treat persistent improvement as a place where decision authority enters, does not distinguish FM-generated motion specifications (keypoints, constraints, programs) from plans or end-to-end actions, and does not analyze verification or how feedback is retained; its selection of works is curated rather than systematic.

Ren et al. survey self-improving agents in general and formalize them through updates to foundation-model parameters or to scaffolding such as prompts, memory, tools, and control logic [17]. They distinguish improvement signals consumed by a fixed external update rule from updates the agent implements on its own definition, gated by a verifier. Our split of I_{fixed} and I_{agent} follows the same idea, but breaks decision authority into separate, observable decisions (target, evidence, tests, procedure, and acceptance; Table III) and applies it to robotic systems, where robotics is one of six application domains in their survey (about two of 57 pages). We add the robot-specific questions of how physical outcomes are verified before an update is committed and whether an FM acts at the task or motion level.

TABLE A4: The synthesized systems under the coding rules of Sec. IV, coded from each paper’s full text (the evidence for every label will be released with the search records). Year is the first public version (arXiv v1 when one exists). T and M : FM participation in task orchestration and motion/action generation, judged from what the FM outputs. I_{fixed} and I_{agent} : persistent improvement in which the FM makes none, or at least one, of the decisions D1–D4 of Table III. Regime: the feedback-closure regime central to the method (Open, Step, Within-rollout memory, Cross-rollout memory, Persistent modification); the last two both reach later tasks and differ in whether records or a changed artifact are kept. It records how feedback is retained, not how capable a system is. FB: ● execution feedback reaches the FM; ○ it is handled only by generated code, a controller, a solver, or a fixed training pipeline; – open loop. † marks systems added from the citation-ranked lists (Appendix A).

System / method	Year	T	M	I_{fixed}	I_{agent}	Regime	FB	Feedback-closure mechanism
Code as Policies [10]	2022	✓	✓	–	–	Step	○	Reactive perception–action loops in generated code
ELM† [81]	2022	–	✓	✓	–	Persistent	○	LLM mutation operator in MAP-Elites robot evolution
HULC++† [34]	2022	✓	–	–	–	Open	–	GPT-3 subgoal sequences for a learned policy
Inner Monologue [1]	2022	✓	–	–	–	Within	●	Success, scene, and human feedback in the planning prompt
LLM-Planner† [54]	2022	✓	–	–	–	Within	●	Few-shot LLM subgoal plans, re-planned with observed objects
LM-Nav† [35]	2022	✓	–	–	–	Open	○	LLM landmark extraction grounded by CLIP
NLMap† [43]	2022	✓	–	–	–	Step	○	LLM queries an open-vocabulary scene representation
ProgPrompt [26]	2022	✓	–	–	–	Step	●	Assertion checks with scripted recovery branches
SayCan [9]	2022	✓	–	–	–	Step	○	Affordance-weighted skill selection from the current state
VLMaps† [39]	2022	✓	✓	–	–	Open	–	LLM writes navigation code with landmark goals and metric moves
Zero-Shot Planner [25]	2022	✓	–	–	–	Open	–	Plan generation mapped to admissible actions
AutoTAMP† [36]	2023	✓	–	–	–	Open	–	LLM translates language into STL for TAMP
ChatGPT Research Group† [70]	2023	✓	✓	–	–	Cross	●	GPT-4 writes a liquid-handling script; BO picks conditions
CLAIRify† [44]	2023	✓	–	–	–	Step	○	Verified LLM task plans for a chemistry lab robot
ConceptGraphs† [45]	2023	✓	–	–	–	Step	●	LLM selects objects from 3D scene-graph captions
DoReMi [55]	2023	✓	–	–	–	Within	●	Constraint-violation detection and recovery
DROC [71]	2023	✓	✓	–	–	Cross	●	Retrieval of knowledge distilled from language corrections
Eureka [23]	2023	–	–	✓	–	Persistent	●	Evolutionary reward-program search
Gen2Sim† [75]	2023	✓	–	✓	–	Persistent	○	LLM task decomposition and reward code for RL
HALP† [76]	2023	✓	–	✓	–	Persistent	●	LLM planner requests and reuses newly taught skills
Incremental humanoid learning† [72]	2023	✓	✓	–	–	Cross	●	Corrected interactions stored and retrieved
KnowNo [27]	2023	✓	–	–	–	Step	●	Uncertainty-triggered help seeking
L3MVN† [46]	2023	✓	–	–	–	Step	●	LM scores each frontier; argmax picks the subgoal
Language to Rewards [50]	2023	–	✓	–	–	Step	○	Reward-mediated skill realization via MPC
LLM-Exo† [87]	2023	✓	✓	✓	–	Persistent	○	LLM writes new gesture code into exoskeleton command tables
LLM-GROP† [40]	2023	✓	✓	–	–	Open	–	LLM arrangement offsets instantiated by TAMP
NavGPT† [60]	2023	–	✓	–	–	Within	●	LLM picks the next viewpoint from described views and history
PaLM-E† [47]	2023	✓	–	–	–	Step	●	Re-prompted each step with image and past steps
PG-VLM† [37]	2023	✓	–	–	–	Open	–	Physically grounded VLM answers planner queries
REFLECT [2]	2023	✓	–	–	–	Within	●	Failure diagnosis and correction planning
RoboCat [18]	2023	–	✓	✓	–	Persistent	●	Self-generated data for the next policy generation

Table A4 (continued)

System / method	Year	T	M	I _{fixed}	I _{agent}	Regime	FB	Feedback-closure mechanism
RoboGen [77]	2023	✓	–	✓	–	Persistent	○	Skill acquisition under a prescribed learning pipeline
RoboGPT (construction) [†] [41]	2023	✓	✓	–	–	Open	–	ChatGPT assembly sequences with pipe coordinates
RoCo [61]	2023	✓	✓	–	–	Within	●	Dialog-based coordination and collision-aware replanning
SayPlan [28]	2023	✓	–	–	–	Within	●	Scene-graph simulator feedback and iterative replanning
Scaling Up & Distilling Down [31]	2023	✓	–	✓	–	Persistent	○	Policy distillation from LLM-guided data collection
SMART-LLM [†] [38]	2023	✓	–	–	–	Open	–	LLM multi-robot decomposition and allocation
Statler [21]	2023	✓	–	–	–	Within	●	Maintained world-state memory
Text2Motion [†] [48]	2023	✓	–	–	–	Step	○	LLM skill plans; learned feasibility and STAP compute geometry
TidyBot [†] [49]	2023	✓	–	–	–	Step	●	LLM-summarized user preferences for object placement
UniSim [†] [88]	2023	✓	✓	✓	–	Persistent	○	RL fine-tuning of a VLA policy in a learned video simulator
VoxPoser [19]	2023	✓	✓	–	–	Step	○	Geometric replanning over composed value maps
Agentic Skill Discovery [78]	2024	✓	–	✓	–	Persistent	●	Skill-library expansion under fixed RL and reward evolution
AIC MLLM [82]	2024	–	✓	✓	–	Persistent	●	Pose correction and test-time model adaptation
BUMBLE [73]	2024	✓	✓	–	–	Cross	●	Short- and long-term episodic memory
ChemAgents [†] [56]	2024	✓	–	–	–	Within	●	Multi-agent LLM compiles lab-robot workflows
COME-Robot [57]	2024	✓	–	–	–	Within	●	Verification and hierarchical recovery
DrEureka [91]	2024	–	–	✓	–	Persistent	●	LLM reward search; randomization ranges from a prior
InterPreT [†] [79]	2024	✓	–	✓	–	Persistent	●	Predicates from language feedback compiled into PDDL
LASP [†] [80]	2024	✓	–	✓	–	Persistent	●	LLM diagnoses errors; fixed loop edits the planning domain
LLM locomotion design [†] [83]	2024	–	✓	✓	–	Persistent	●	LLM designs trajectories, IK, and reward code
MOKA [29]	2024	✓	✓	✓	–	Persistent	●	Distillation of collected experience into a policy
ORGANA [†] [58]	2024	✓	–	–	–	Within	●	LLM plans; records summarized within one campaign
PIVOT [22]	2024	–	✓	–	–	Step	●	Iterative visual action-candidate selection
ReKep [20]	2024	✓	✓	–	–	Step	○	Re-optimization of relational keypoint constraints
VLM-Social-Nav [51]	2024	–	✓	–	–	Step	●	VLM-derived social cost in per-step replanning
VLMnav [52]	2024	–	✓	–	–	Step	●	Step-wise visual navigation decisions
Agentic Robot [62]	2025	✓	✓	–	–	Within	●	Verifier triggers a fixed recovery routine and retry
ARNA [63]	2025	✓	✓	–	–	Within	●	Iterative tool-use feedback
ELLMER [†] [53]	2025	✓	✓	–	–	Step	○	GPT-4 writes retrieved code with force/vision feedback
LAMARL [†] [84]	2025	–	✓	✓	–	Persistent	○	LLM prior policy and reward code for MARL
Maestro [30]	2025	✓	✓	–	–	Cross	●	Cross-trial in-context reflection
ManiAgent [64]	2025	✓	✓	–	–	Within	●	Cross-task action-sequence cache (retrieval memory)
PragmaBot [74]	2025	✓	✓	–	–	Cross	●	Experiential memory summarized from trial and error
RAI [65]	2025	✓	✓	–	–	Within	●	Robot-tool feedback, including point targets
Reliable Code-as-Policies [66]	2025	✓	✓	–	–	Within	●	Interactive validation during code generation
SAS-Prompt [69]	2025	–	✓	–	–	Cross	●	In-context controller-parameter search from rollout traces
Self-Improving EFM [85]	2025	–	✓	✓	–	Persistent	●	Online policy update from self-predicted rewards
VLMgineer [86]	2025	–	✓	✓	–	Persistent	●	Evolutionary tool/action co-design
ACE [59]	2026	✓	–	–	–	Within	●	Verification, repair, and replanning of sub-goals
Act-Observe-Rewrite [89]	2026	✓	✓	✓	–	Persistent	●	Policy-code rewriting from a fixed outcome package
ALRM [67]	2026	✓	✓	–	–	Within	●	ReAct-style tool feedback
ASPIRE [32]	2026	✓	✓	–	✓	Persistent	●	Trace-guided program repair; repair notes kept as memory
CaP-X / CaP-Agent0 [90]	2026	✓	✓	✓	–	Persistent	●	One-pass skill-library synthesis from successful rollouts
ENPIRE [24]	2026	✓	✓	–	✓	Persistent	●	Agents revise policy, algorithm, and training code on robots
HoloMind / LongAct [68]	2026	✓	–	–	–	Cross	●	Episodic experience memory
RHO [33]	2026	✓	✓	–	✓	Persistent	●	Coding agent tests its own edits in a fixed evolutionary loop
Uni-Skill [42]	2026	✓	✓	–	–	Open	–	On-demand skill retrieval from an offline skill repository

TABLE A2

SYSTEMS IN THE FINAL CORPUS BY YEAR OF FIRST PUBLIC VERSION, WITH THE SHARE AT EACH LOCUS (%; 95% WILSON INTERVAL FOR M) AND THE NUMBER OF FIXED-PROCEDURE AND AGENT-DIRECTED IMPROVEMENT SYSTEMS. T AND M ARE ABSTRACT-LEVEL AFTER ADJUDICATION; IMPROVEMENT TYPES COMBINE ABSTRACT-LEVEL LABELS WITH FULL-TEXT RESOLUTION. * 1 JANUARY TO 23 SEPTEMBER 2026.

Year	n	T	M	I	I_{fixed}	I_{agent}
2022	16	81	25 (10–49)	12	2	0
2023	146	76	21 (15–28)	21	30	0
2024	457	74	33 (28–37)	19	87	0
2025	863	75	38 (35–42)	14	123	1
2026*	861	73	43 (39–46)	18	143	16

TABLE A3

COMPOSITION OF THE FINAL CORPUS AND OF THE SYNTHESIZED SYSTEMS (%). CORPUS LABELS ARE ABSTRACT-LEVEL AFTER ADJUDICATION (IMPROVEMENT TYPES AFTER FULL-TEXT RESOLUTION); SYNTHESIZED-SYSTEM LABELS ARE FROM FULL TEXT.

		Corpus	Synthesized
Year	2022	1	14
	2023	6	40
	2024	20	19
	2025	37	15
	2026	37	12
Locus	I_{agent}	1	4
	I_{fixed}	16	27
	M only	15	8
	T only	51	35
	T and M	16	27
Evaluation	sim and real	26	50
	real only	24	26
	sim only	29	24
	unclear	21	0
Embodiment	arm	38	53
	humanoid	3	4
	legged	2	3
	mobile manipulator	8	17
	multi-robot	9	4
	navigation	20	13
	other	19	8

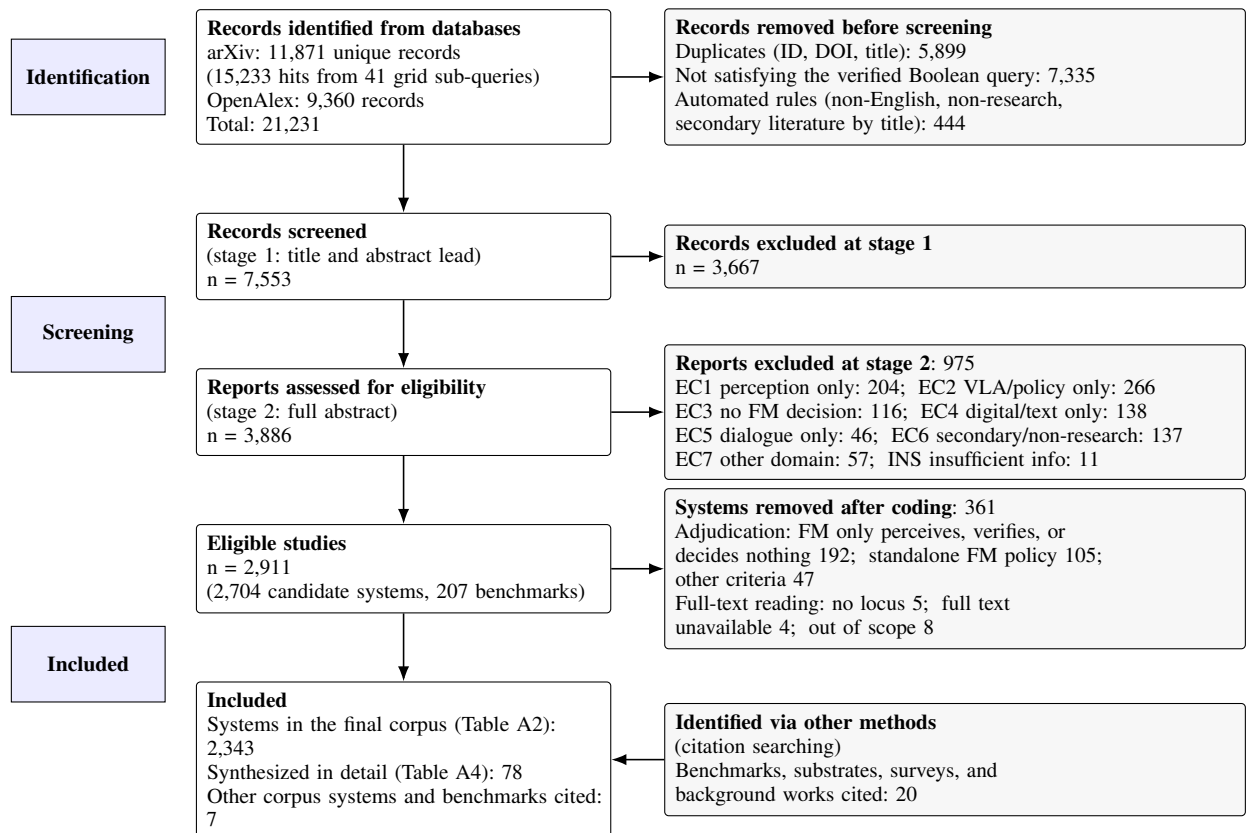


Fig. A1. PRISMA 2020 flow diagram of the systematic search (search executed 23–24 September 2026; window 1 January 2022 to 23 September 2026).

TABLE A5

REPRESENTATIVE REUSABLE BENCHMARKS AND EVALUATION SUBSTRATES RELEVANT TO AGENTIC ROBOTICS. T AND M DENOTE TASK- AND MOTION-LEVEL EVALUATION; “FEEDBACK-RICH” MARKS BENCHMARKS THAT EXPOSE EXECUTION HISTORY, FAILURE, OR OUTCOME EVIDENCE. “ I SUBSTRATE” MEANS THAT AN ENVIRONMENT CAN SUPPORT REPEATED ADAPTATION BUT DOES NOT ITSELF DEFINE A STANDARDIZED SELF-IMPROVEMENT PROTOCOL. ROBOFAIL TARGETS VERIFICATION RATHER THAN A BEHAVIOR-GENERATING LOCUS.

Benchmark / substrate	Primary locus	Environment / interface	What the benchmark primarily exposes
VirtualHome [92]	T	Unity household simulator with symbolic actions	Executability and goal satisfaction of long-horizon semantic action sequences; used prominently by early LLM planners such as Zero-Shot Planner and ProgPrompt [25], [26].
Embodied Agent Interface (EAI) [93]	T	VirtualHome + BEHAVIOR	Goal interpretation, subgoal decomposition, action sequencing, and transition modeling under standardized symbolic interfaces, with fine-grained planning and precondition errors rather than only terminal success.
LoHoRavens [94]	T ; feedback-rich	Language-conditioned tabletop simulation	Long-horizon reasoning over color, size, spatial relations, arithmetic, and references, while explicitly studying how execution observations are returned to an LLM planner.
LongAct [68]	T ; feedback-rich	Long-horizon household tasks with low-level control abstracted	Planning-level autonomy from free-form instructions, including instruction understanding, dependency management, persistent memory, and adaptive replanning; an improvement-rate metric additionally measures gains from accumulated experience.
EmbodiedBench [95]	T, M	Four embodied environments	1,128 vision-driven tasks spanning high-level household reasoning and low-level navigation/manipulation, with capability subsets for commonsense reasoning, complex instruction understanding, spatial awareness, visual perception, and long-horizon planning.
RoboFail [2]	Feedback / verification	Simulated and physical robot failure traces	Failure localization, explanation, and correction from multimodal execution history; one of the few datasets built specifically around failed rather than successful robot executions.
RoCoBench [61]	T, M ; feedback-rich	Six multi-robot manipulation tasks + text-only reasoning data	Multi-agent task allocation, communication, coordinated waypoint generation, collision-aware replanning, and execution-feedback-driven collaboration.
Kitchen-R [104]	T, M	Isaac Sim digital-twin kitchen	Joint language-guided mobile manipulation; supports planner-only, low-level-policy-only, and integrated whole-system evaluation, directly exposing the interface between semantic planning and physical realization.
RLBench [96]	M ; I substrate	CoppeliaSim	Large task diversity for vision-guided manipulation. In agentic work it is also reused for evaluating generated programs, skill implementations, and zero-shot manipulation rather than only learned policies.
CALVIN [97]	M	PyBullet	Language-conditioned continuous manipulation and long-horizon composition, evaluated by how many consecutive instructions (one to five) in a five-instruction chain are completed; subtask instructions are given to the policy rather than planned by an FM.
LIBERO / LIBERO-PRO [98], [105]	M ; I substrate	robosuite / MuJoCo	Knowledge transfer across the Spatial, Object, and Goal suites (10 tasks each) and LIBERO-100, which splits into 90 short-horizon tasks (LIBERO-90) and 10 long-horizon tasks (LIBERO-Long, later also called LIBERO-10); LIBERO-PRO adds perturbations of objects, initial states, instructions, and environments. It is used to test generated programs and persistent artifacts under perturbations [32], [33], [90]; RHO keeps the perturbation types out of its improvement data, whereas ASPIRE debugs on other seeds of the same perturbed tasks.
robosuite [106]	M ; I substrate	MuJoCo	Contact-rich manipulation with modular robots, controllers, and observations; frequently reused for code-as-policy generation, iterative controller repair, and persistent repository optimization [32], [33], [89], [90].
BEHAVIOR-1K [107]	T, M ; I substrate	OmniGibson	Long-horizon household activities with navigation, manipulation, articulated objects, and rich object states; increasingly used for integrated and skill-accumulation evaluation.
M ³ Bench [108]	M	Physics-based mobile-manipulation simulation	30,000 rearrangement tasks across 119 scenes for coordinated whole-body base-arm motion generation and generalization under scene and task constraints.
RoboCasa365 [109]	T, M ; I substrate	RoboCasa / MuJoCo	365 household tasks across 2,500 kitchen environments, supporting multi-task, foundation-model, compositional-generalization, and lifelong-learning evaluation.
CaP-Bench / CaP-X [90]	T, M ; I substrate	Interactive CaP-Gym over robot simulators	Coding-agent performance while systematically varying action-API abstraction, perceptual grounding, single-versus multi-turn interaction, visual feedback, and skill synthesis.
RAI Bench [65]	T, M ; I substrate	ROS 2 / O3DE and tool-call interfaces	Manipulation, tool-calling, and VLM benchmarks released with the open-source RAI agent framework (rather than defined in the RAI paper itself); RHO uses its O3DE manipulation benchmark to optimize the harness (prompts, tools, control code) of a deployed LLM agent, evaluated on held-out tasks [33].
ALRM Benchmark [67]	T, M	Simulation; 54 manipulation tasks	Multistep reasoning and linguistic variation under both code-as-policy and tool-as-policy execution, explicitly addressing dimensions that conventional low-level manipulation benchmarks do not isolate; its test environment returns placeholder poses, so it checks planning and call formatting more than physical execution.

TABLE A6
EVALUATION SUBSTRATES, PERSISTENT ARTIFACTS, AND THE AUTHORS’ OWN STATED LIMITATIONS FOR THE SELF-IMPROVING SYSTEMS OF TABLE IV.
TYPE FOLLOWS TABLE III.

Method	Type	Evaluation substrate	Persistent artifact	Stated limitations (quoted or paraphrased)
RoboCat [18]	I_{fixed}	Real Sawyer and Panda; 11 self-improvement task variants	Generalist policy weights	“not autonomous . . . data collection is started and stopped by humans”; behaviour cloning only; controlled lab setting; success detectors about 90% accurate.
Self-Improving EFM [85]	I_{fixed}	LanguageTable (sim, real), Aloha (sim), real BananaTable	Policy weights	Needs better stopping criteria to prevent over-optimizing the shaped reward; real-world Aloha self-improvement could not be completed.
Eureka [23]	I_{fixed}	29 Isaac Gym and Bi-Dexterous Manipulation tasks	Reward program	Simulation only; requires a human-defined task fitness function; one simulator and a fixed RL algorithm.
CaP-X [90]	I_{fixed}	CaP-Bench (Robosuite), LIBERO-PRO, BEHAVIOR; real demonstrations	Skill library (one synthesis pass)	Programmatic control “remains brittle for contact-rich behaviors”; the library is synthesized in a single pass.
Act-Observe-Rewrite [89]	I_{fixed}	Robosuite Lift, PickPlaceCan, Stack	Policy code (vision module and controller)	“Simulation only”; one hypothesis tested per trial; converged to a local optimum; depends on the LLM’s diagnosis.
RHO [33]	I_{agent}	CaP-Bench Robosuite, LIBERO-PRO, RAI O3DE	Multi-file policy repository	“All of our benchmarks are simulated”; inherits the LLM’s spatial-reasoning limits; an agent once terminated the operator’s session when told to stop hard-coding per-trial solutions.
ASPIRE [32]	I_{agent}	LIBERO-PRO, Robosuite, BEHAVIOR-1K; real bimanual robot	Repaired program per task (skills kept as in-context memory)	“not yet a fully autonomous real-world lifelong learner”; needs success detection, safe reset, and calibration; skill entries can become stale; compute-intensive.
ENPIRE [24]	I_{agent}	Real Push-T, pin and GPU insertion, zip-tie cutting; RoboCasa365	Policy, training code, data mix	Agents leave robots idle while reading logs or coding; token cost grows super-linearly with fleet size.