CODING AGENTS WITH HARNESS FOR SAFE ROBOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents have emerged as a promising paradigm for robot manipulation: a language model writes the robot controller as a program, and agents built in this way now operate robots without robot-specific training. Whether this paradigm is also safe, however, has not been asked. We evaluate coding agent under a safety constraint, where each task pairs a manipulation goal with an obstacle the robot must not touch. The agent pursues the goal but collides with the obstacle in most cases, treating task completion as its sole objective while neglecting safety. The agent reasons about the obstacle in its traces, and the prompt already forbids touching it, so neither perception nor instruction is at fault; the fault lies in the planning, where the stated constraint never becomes a priority. By decomposing manipulation into a route phase and a contact-rich moment, we locate the source of the failure. Along the route, the model cannot prioritize the safety constraint, having no notion of a clearing route and none of replanning once a chosen route becomes infeasible. At the contact, it is unaware that contact execution is bounded by the same constraint. To close this gap, we present **SAFEHARNESS**, which equips the model with two obstacle-aware harnesses that enable it to prioritize the safety constraint. Obstacle-aware route planning grounds the objects as bounding boxes and draws candidate routes over them as sequences of waypoints. The agent then plans a route in advance, verifies it, replans when necessary, and only then executes it. Obstacle-aware contact execution instead selects the contact position so that the contact itself avoids the obstacle. **SAFEHARNESS** attains 81.2% task success and 91.9% collision avoidance with GPT-6-Astra, surpassing the previous SOTA by 13.7 and 23.0 points, respectively. Compared with the same agent without harnesses, these results represent gains of 31.2 and 57.5 points, respectively.

1 INTRODUCTION

Coding agents have emerged as a promising paradigm for robot manipulation. Large Language Models (LLMs) can write robot controllers as programs. Code as Policies [27] set the pattern: the model composes perception and control APIs into a program, and the program is the policy. Later agents rewrite their own controller code after a failure [23], keep a library of skills that worked [32], and spend more test-time compute for more reliability [14]. Frontier agents now drive real robots without robot-specific training [20, 47]. Since the release of GPT-6 Astra [36], many groups have reported promising robot results with it. With no robot-specific training, Astra completes pick-and-place on a real arm almost perfectly [33, 20], and paired with a frozen $\pi_{0.5}$ it extends to bimanual, contact-rich tasks [41]. Yet Astra contains no robot policy. Every robot result attributed to it belongs to the harness around it, the tools the agent is given to call, and coding agents make this division of labor explicit. Harness VLA [52] lets the agent explore a scene, keep what worked as a skill memory, and reuse it on later tasks, calling a frozen VLA only where the task turns contact-rich. Xu et al. [48] extends it to long horizons.

None of this work has asked whether such paradigm on robot manipulation is safe. The studies above score a rollout on whether the goal was reached, and the benchmarks they inherit do the same [28, 11]. What the robot touched on the way is never measured. Yet a collision in a cluttered workspace damages hardware, injures people, and destroys property [15]. Success does not imply safety [17]. Across a model generation the two can even diverge [10, 13]. Scaling up data does not resolve this

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

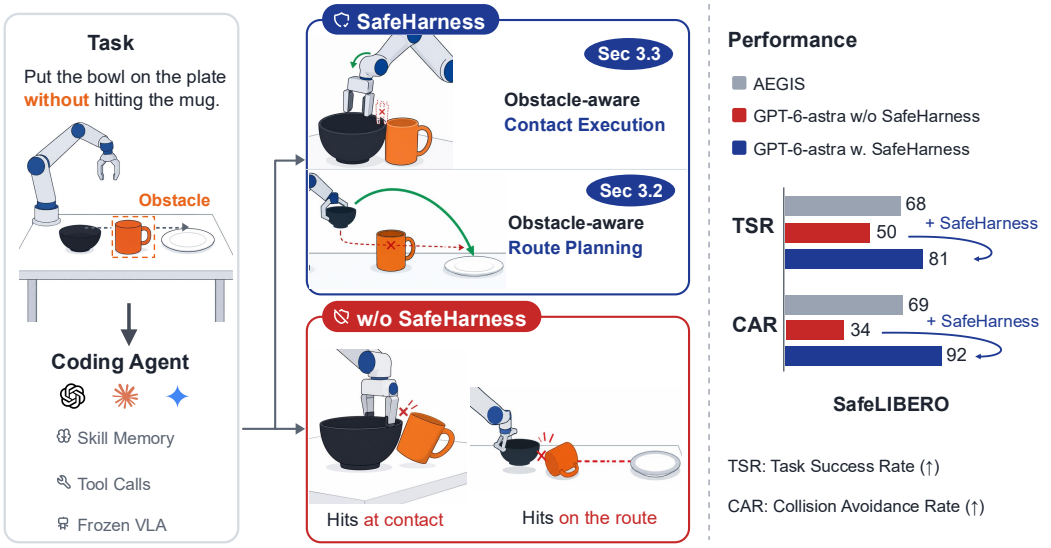


Figure 1: **SAFEHARNESS puts safety on par with task completion.** Coding agent receives a task instruction with a safety constraint, along with observation images. **SAFEHARNESS** then equips it with obstacle-aware route planning (Sec. 3.2) and obstacle-aware contact execution (Sec. 3.3) to enforce safety throughout the task.

either. Training on ten times as many collision-free demonstrations lowers the collision rate by under three points [11]. Completing the task and touching nothing else are two halves of one requirement.

To solve this problem, we first evaluate the coding agent on robot safety benchmark. Upon evaluation, *we observe that coding agent performs poorly under a safety constraint*. The coding agent completes the task but collides with the obstacle in most cases. It treats the task goal as the core objective and neglects that safety has to be satisfied as well. The cause is not recognition, since the agent identifies the obstacle correctly. Nor is it instruction. Adding the safety requirement to the prompt does not change the behavior [31, 49]: the agent restates the constraint and then violates it. The reasons follow the decomposition that coding agent itself gives, a route through free space and a contact-rich behavior at its end. First, for the route ❶, the model has no notion of a path that clears the obstacle, and it takes the shortest one. Second, it has no notion of replanning when the route it chose turns infeasible midway. Third, for the contact ❷, it does not recognize that the contact itself carries a safety constraint. A stronger planner does not close the gap (Sec. 4.3), as also reported at scale [51]. The safety constraint is simply never a core priority.

We propose **SAFEHARNESS** to address the gap that the coding agent fails to accord safety the same priority as task completion. **SAFEHARNESS** addresses this gap by equipping the model with two obstacle-aware harnesses that give it the ability to prioritize the safety constraint (Fig. 1). *Obstacle-aware route planning* (Sec. 3.2) handles ❶. The objects in the scene are grounded as bounding boxes [7], and a route over them is drawn as a sequence of waypoints. The agent plans the route ahead, verifies it, replans when a problem is met along the way, and finally executes the path that has been cleared. The safety constraint is thereby held at the same top priority as the task goal. *Obstacle-aware contact execution* (Sec. 3.3) handles ❷. The contact strategy, meaning the exact contact position and whether the gripper rotates, is decided by whether an obstacle stands next to the contact location. The same question is asked where the object is picked and where it is placed. Our core contributions are:

- **Coding agents for safe robot control.** We study coding agent for robot control under a requirement it has never been scored on: finish the task under safety constraint. To our knowledge, this is the first coding agent for robot manipulation whose harness itself enforces safety: obstacle avoidance is verified throughout the task, including its contact-rich phases. It is also the first such agent evaluated on a collision-scored manipulation benchmark.
- **SAFEHARNESS.** Our two obstacle-aware harnesses treat safety as a first-class objective, on equal footing with task completion. ❶ Route planning plans, verifies, guards, and replans every free-space route. ❷ Contact execution selects the contact position and wrist rotation from the spatial relation between the target and its neighboring obstacle.

- **Results.** SAFEHARNESS attains 81.2% task success and 91.9% collision avoidance on Safe-LIBERO, surpassing the previous SOTA by 13.7 and 23.0 points, respectively. Compared with the same agent without harnesses, these results represent gains of 31.2 and 57.5 points, respectively.

2 RELATED WORK

From VLAs and world-action models to coding agents. Vision-language-action (VLA) models map pixels and instructions directly to actions. RT-2 casts action prediction as next-token generation over a vision-language backbone [5], Octo trains a generalist policy across many embodiments [35], and OpenVLA and its fine-tuning recipe scale an open backbone to new manipulation tasks with modest data [21, 22]. The π_0 and $\pi_{0.5}$ family adds a flow-matching action head for dexterous, contact-rich control [4, 37]. A parallel line pretrains on video before it predicts action: UniPi extracts actions from a generated goal video [12], GR-1 and GR-2 predict future frames and actions jointly [45, 9], and WorldVLA and V-JEPA 2-AC couple world-model prediction and action generation inside one architecture [8, 3]. Both families are trained policies, so their competence reaches only as far as their demonstrations. A world-action model moreover inherits much of the brittleness of the VLA it is built from [53], and in both cases the decision resolved inside an action head resists inspection and correction. A coding agent instead has a language model write the controller as a program that calls perception and control skills; written rather than trained, such a policy transfers across tasks without robot-specific data, can be read and edited before execution, and may compose existing skills, learned policies included.

Coding agents in robot manipulation. A coding agent acts by writing and executing code rather than emitting actions or language directly [44, 43]. Code as Policies brought this idea to robot manipulation, with a program composed from perception and control APIs serving as the policy [27]. Follow-on work structured what the program could express, from the prompt and its perception calls [40, 18] to object-centric code with affordance and safety constraints [34] and 3D value maps for a motion optimizer [19]. A second line lets the agent improve its own code, rewriting it from observed failures [23], growing a skill library by evolutionary search [30] or with a human in the loop [32], self-improving on real hardware [47], and trading test-time compute for reliability [14]. The strongest recent results come from GPT-6 Astra [36], which without robot-specific training reaches near-perfect pick-and-place on a real arm [33, 20] and, with a frozen $\pi_{0.5}$ as a callable tool, handles bimanual, contact-rich tasks [41]. The model itself emits no actions, so each of these results is as much a statement about the harness as about the model. A third line exposes a frozen policy as one callable primitive: Harness VLA identifies the scene once, crosses free space with analytic inverse-kinematics primitives, and hands over to the frozen policy only where the task turns contact-rich [52], and BATON extends this design to long-horizon tasks [48]. Across all three lines, the code plans and sequences skills, but a skill executes as a straight-line motion or a learned primitive regardless of what stands in its path; we are the first to study coding agents for robot manipulation with safety placed on equal footing with task success.

Safe control in robotics. Sampling- and optimization-based motion planners construct an explicit route through free space and certify it against a collision model: RRT grows a tree of feasible configurations toward the goal [25], and CHOMP and TrajOpt optimize a trajectory against a swept-volume penalty [38, 39]. These planners fix a route before execution. Runtime safety filters address that limitation: control barrier functions wrap a controller in a filter that minimally edits each commanded action to keep the system inside a safe set [1], later extended to semantic, scene-grounded regions [6]. Such a filter edits whatever action it is handed; it cannot choose a different grasp side or route. SafeLIBERO adds one obstacle, near the target or on the transport path, to four tasks in each of the LIBERO-Spatial, Goal, Object, and Libero-10 suites [28], scores an episode on task success and on whether the obstacle ever moves, and pairs the benchmark with AEGIS, a barrier-function layer over a frozen policy [16]. LIBERO-Safety applies the same measurement to semantic constraints and finds that scaling demonstrations tenfold barely lowers the collision rate [11]. Across this line, safety is scored or filtered after a policy commits to a motion, rather than built into the decision that produces it. SAFEHARNESS instead puts the check inside the decision, and the model that proposes the route could be replaced by any planner without changing the rest of the harness. we are the first to bring the coding-agent paradigm to safe robot control.

3 METHOD

3.1 PRELIMINARIES

Coding agents have become a strong recipe for robot manipulation: a language model writes the controller as a program, and the program generalizes across tasks without robot-specific training. Safe manipulation has so far been pursued along other routes: planners that certify a trajectory in advance [25, 38, 39], filters that edit each commanded action [1, 6, 16], policies trained under a safety objective [50, 42]. Whether a coding agent can deliver safe manipulation has not been asked, and we are the first to explore it.

We ask it here, and the recipe itself suggests how to approach it. A coding agent composes a task from callable skills and motion primitives, and the natural place to cut such a program is a contact event. Each phase is then a *route* through free space followed by a *contact-rich behavior* at its end. An obstacle can be touched in either part. The two parts nonetheless put it at risk in different ways. A route sweeps a long path across the workspace, whereas a contact-rich behavior sweeps a small volume that cannot be moved away from the target it acts on. We therefore split the safety constraint in the same place. The route constraint asks for a clear path where no single point would collide with the obstacle. The contact constraint asks that the contact-rich behavior be executed strategically when an obstacle is nearby. Neither constraint implies the other, so an episode is safe only when both of them hold. The split gives **SAFEHARNESS** its two components (Fig. 2). *Obstacle-aware route planning* (Sec. 3.2) is responsible for the route constraint ❶. *Obstacle-aware contact execution* (Sec. 3.3) is responsible for the contact constraint ❷. Both components run once per phase, so a task with two sub-goals repeats them twice.

3.2 OBSTACLE-AWARE ROUTE PLANNING

A route is decided before the arm moves. Planning and execution are asymmetric in cost: a plan can be revised freely, whereas a motion already issued acts in the workspace and cannot be recalled, so errors are best caught while they are still words rather than movements. **SAFEHARNESS** accordingly plans the route of a phase ahead, verifies the plan against the obstacle, and executes only a plan that has passed verification.

Route planning begins anew with each phase rather than once for the whole task. Once the object is in the gripper, arm and object move as one body, and the clearance a route requires changes with it. A route cleared for the empty gripper is not cleared for the loaded one. The first step of a phase is visual grounding. The target and the obstacle are grounded as bounding boxes by a segmentation model. On that view the model is asked for a route as a polyline of waypoints, under two conditions: it must end at the target, and it must not cross the obstacle box.

The proposal obtained this way is not yet a route. Before any motion is issued, it is checked against the obstacle box outside the language model; the check covers the gripper and, when an object is held, the object as well. A candidate that intrudes into the box is rejected and the model replans, until a route passes and is released for execution.

A route released for execution carries, however, only a limited guarantee. The verifier evaluates a single condition, namely whether the route intersects the obstacle, and it does so on an approximately measured scene. Other physical constraints that arise during execution fall outside its scope, and a route that passes verification may still fail against them. Replanning is therefore triggered at two points. The first occurs at verification, before any motion is issued: if the proposed route would enter the obstacle box, it is rejected and a new route is planned. The second occurs during execution, when the robot encounters conditions the verifier did not evaluate. The arm may approach the obstacle more closely than the plan predicted, or it may become stuck partway along the route due to a physical constraint that the bounding boxes did not capture. In either case, execution halts and a new route is planned from the pose at which the motion stopped. Planning can thus be invoked repeatedly within a phase rather than performed once at its outset.

Beyond replanning, execution imposes one further discipline: economy of observation. The natural impulse is to consult the route-plan image at every step, yet each consultation appends the same view to the context window, and a context window inflated by repeated identical observations degrades the quality of the model's subsequent predictions. **SAFEHARNESS** therefore bounds the number

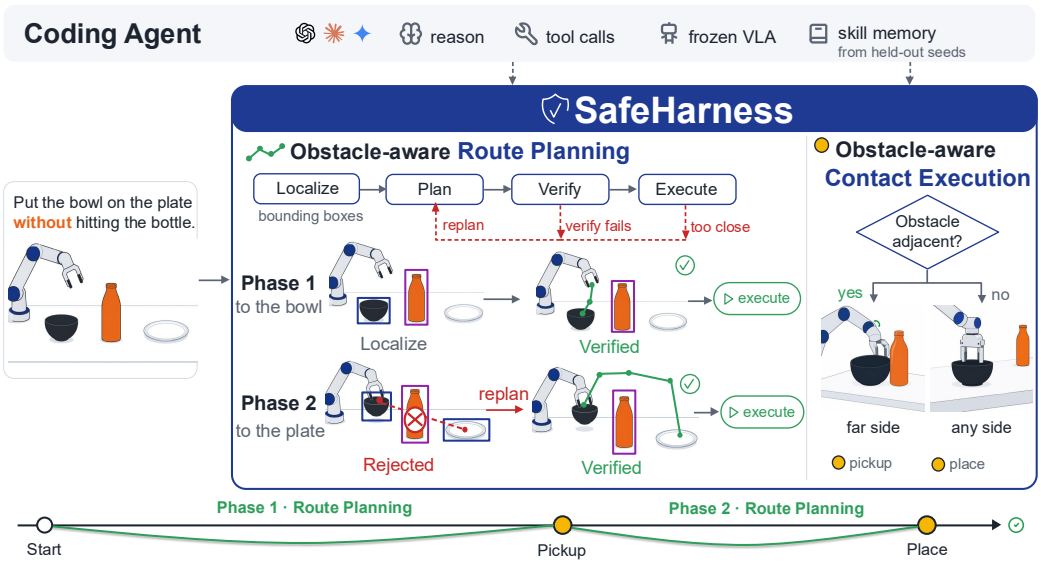


Figure 2: **SafeHarness overview.** The task is cut at its contact events into phases, and both components run once per phase. ❶ *Obstacle-aware route planning*: the scene is grounded as boxes, the route of the current phase is planned, verified against the obstacle, and only then executed, with a failed verification or a halt during execution sending the loop back to planning. ❷ *Obstacle-aware contact execution*: the contact target is tested for an adjacent obstacle, and the contact is made from the far side whenever one is found.

of times the route-plan image may be read during the execution of a route. The bound removes redundant views without withholding information, since a repeated image lengthens the context but adds nothing to what the model knows.

Taken together, the phase runs as a plan–verify–execute cycle: a route is proposed on the grounded scene, admitted only after it clears the obstacle box, and executed under the same check, with planning re-entered whenever verification or execution rejects the current route. What the cycle delivers is a path that has been confirmed clear before and during motion, ending just above the contact target at a pre-grasp or a release pose, where the phase is handed to the contact step (Sec. 3.3). A long-horizon task is covered by repeating this cycle phase by phase rather than by a longer plan.

3.3 Obstacle-Aware Contact Execution

The second harness addresses the contact motion, the segment of the phase that the route does not cover. The route terminates above the contact target, at a pre-grasp or a release pose; below this pose the gripper opens, descends, and closes, and this motion sweeps a volume that route verification does not examine. Moreover, whereas the route is free to bend around the obstacle, the contact motion is anchored to the target and cannot avoid its vicinity. When the obstacle is adjacent to the target, collisions therefore occur in this final segment even when the route itself is collision-free, and the contact motion must be planned with respect to the obstacle as well.

The contact step is resolved in the same manner as the route: the strategy is determined anew at each contact rather than fixed once for the entire task. Each contact begins with an adjacency test between the obstacle and the contact target. If no obstacle is adjacent, any contact strategy is admissible and the step proceeds as it would in an uncluttered scene. If an obstacle is adjacent, the contact is constrained to approach from a direction that maintains clearance between the gripper and the obstacle, and among the directions that satisfy this condition, the direction farthest from the obstacle is selected. If no direction satisfies the condition, the gripper is reoriented so that its opening axis runs tangent to the obstacle, and the payload is lowered vertically to avoid lateral motion toward the obstacle. The contact strategy is therefore governed by obstacle clearance at every contact.

3.4 Discussion: Constraints Under a Growing Context

The design of **SafeHarness** also addresses a property of coding agents that is independent of any particular constraint. An episode unfolds as a single conversation spanning hundreds of steps, in which every tool call appends its result and every observation appends an image. By the time the arm

approaches the obstacle, the context holds tens of thousands of tokens, while the constraint is stated near its beginning. Language models exploit such content less reliably as the context grows: recall degrades for material in the middle of long inputs [29], instruction following deteriorates with length [46], system-prompt adherence drifts within a few turns [26], performance declines over multi-turn conversations [24], and even trained safety behavior weakens under sufficiently long contexts [2]. A constraint conveyed solely through the prompt is thus exposed to the very context that task execution produces, and a stronger backbone does not remove this exposure (Sec. 4.3). **SAFEHARNESS** does not rely on the constraint surviving the context. The route is planned and verified before the arm moves, using tools that are stateless with respect to the conversation: they maintain the obstacle box outside the context and return results on which the planner acts in the next step. Safety is thereby enforced explicitly wherever a motion is decided, rather than requested once at the outset.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmark. [16] is constructed from four representative tasks drawn from each of the LIBERO-Spatial, Goal, Object, and Libero-10 suites [28] yielding 16 base tasks. Each augmented with one obstacle object selected from six household items: a moka pot, storage box, milk carton, wine bottle, mug, or book. Each task is instantiated at two difficulty levels: in Level I, the obstacle is placed in close proximity to the target object, whereas in Level II it is placed away from the target but directly on the natural transport path, resulting in 32 tasks in total. We evaluate **SAFEHARNESS** on all 32 SafeLIBERO tasks with 10 seeds per task, and on real-robot using PiPER-X 6-DoF research arm for two tasks with 20 trials each.

Metrics. Following the official SafeLIBERO evaluator, an obstacle is considered displaced if it moves by more than 1 mm at any point during an episode. We report three metrics: task success rate (TSR), the percentage of episodes that achieve the task goal; collision-avoidance rate (CAR), the percentage of episodes in which no obstacle is displaced; and safe success rate (SSR), the percentage of episodes that achieves the task goal with no displaced obstacle.

Models. Our system uses a coding agent to compose a frozen VLA policy with the obstacle-aware harnesses introduced in Sec. 3. We use the same frozen VLA policy as Harness VLA [52] without any fine-tuning, SAM-3 [7] for segmentation, and either GPT-5.5 or GPT-6-Astra [36] as the coding-agent backbone, reporting results for both backbones in Tab. 1 – 3. To keep comparisons controlled, the coding-agent loop and the LIBERO skill memory [28] follow the configuration of Harness VLA [52], so that performance differences can be attributed to the proposed harnesses rather than to the surrounding infrastructure. In the ablation study, the backbone alone, without tools, skills, or memory, serves as the *Model* variant, and this shared configuration without our harnesses serves as *Model + Skills* (Sec. 4.3). In the main experiments, we compare against two foundation VLA models, $\pi_{0.5}$ [37] and OpenVLA-OFT [22], as well as the previous state-of-the-art method AEGIS [16], which augments $\pi_{0.5}$ with a barrier-function-based safety layer.

Implementation Details. At every step the agent receives the third-person view, the wrist view, and the proprioceptive state of the arm. Objects are grounded on the third-person view with SAM-3 [7]. Routes are planned on the third-person view with at least 15 cm of clearance from the obstacle box, and the verifier rejects any route along which the gripper, or the held object when loaded, enters the inflated box. During execution, motion halts if the end effector comes within 3 cm of the obstacle box or makes no progress for 30 control steps, and the route is replanned from the current pose, at most three times. The adjacency test fires when the obstacle box lies within 10 cm of the contact target, and the VLA policy is invoked only after the wrist view confirms that the gripper is ready to make contact. The route-plan image is read at most 20 times per episode. On SafeLIBERO, an episode is allowed at most 900 environment steps; a run without replanning takes about 250 steps on average, and about twice that on Libero-10, whose tasks comprise two sub-goals.

4.2 MAIN RESULTS

The source of these gains lies in how each method accounts for obstacles during execution. A frozen VLA policy never encounters obstacle avoidance in its training objective, so nothing inside the policy distinguishes an approach that clears the obstacle from one that sweeps through it. A barrier layer such as AEGIS sits downstream of the policy and edits the commanded action: it can veto a motion that violates the safety constraint but has no means of proposing a better one, so each intervention halts an unsafe motion without steering the policy back toward the goal, and this cost

Table 1: SafeLIBERO [16]: per-suite and average TSR and CAR, in %. **SAFEHARNESS** instantiated with two coding-agent backbones (GPT-5.5 and GPT-6-Astra). Best per column in bold.

	Spatial		Goal		Object		Libero-10		Average	
Method	TSR	CAR	TSR	CAR	TSR	CAR	TSR	CAR	TSR	CAR
OpenVLA-OFT [22]	36.5	8.3	24.3	19.5	28.8	11.5	15.3	6.0	26.2	11.3
$\pi_{0.5}$ [37]	59.3	14.0	60.0	20.0	57.5	18.0	54.3	16.5	57.8	17.1
AEGIS [16]	68.5	68.0	**82.8**	76.5	72.5	71.3	46.3	59.8	67.5	68.9
SAFEHARNESS (GPT-5.5)	**81.2**	76.2	68.8	88.8	90.0	87.5	75.0	95.0	78.8	86.9
SAFEHARNESS (GPT-6-Astra)	76.2	**90.0**	78.8	**90.0**	**92.5**	**91.2**	**77.5**	**96.2**	**81.2**	**91.9**

Table 2: Real-robot results, TSR/ CAR in %, 20 trials per task. Each scene holds two obstacles, one beside the target object and one on the path to the plate. Small numbers give the relative improvement of **SAFEHARNESS** over the GPT-5.5 coding agent with the same backbone. Best per column in bold.

	White can → plate		Coke bottle → plate		Average	
Method	TSR	CAR	TSR	CAR	TSR	CAR
$\pi_{0.5}$	20	15	30	20	25.0	17.5
GPT-5.5	30	40	25	50	27.5	45.0
SAFEHARNESS (GPT-5.5)	**70** +133%	**85** +113%	**60** +140%	**80** +60%	**65.0** +136%	**82.5** +83%

compounds when interventions recur. On the Libero-10 suite, whose two sub-goals each call for such an intervention, AEGIS even falls below the frozen policy it wraps in task success. The vanilla GPT-5.5 agent can in principle reason about obstacles, but its awareness of safety derives entirely from the prompt, so its safety behavior depends on whether the constraint remains salient as the context grows over the episode, which we examine in Sec. 4.3. **SAFEHARNESS**, in contrast, grounds the obstacles in the scene and plans the route and execution steps around them before acting, so that collision avoidance is built into the plan rather than enforced after the fact or left to the prompt. This design retains the adaptability of a coding agent while avoiding the fixed behavior of VLA policies that replay patterns learned from training data.

The real-robot results in Tab. 2 follow the same pattern, with **SAFEHARNESS** leading on both tasks under both metrics. We compare against $\pi_{0.5}$ and a vanilla coding agent built on GPT-5.5, which shares the backbone of **SAFEHARNESS** but lacks our harness. $\pi_{0.5}$ performs worst and rarely avoids collisions. Its trajectory remains largely unchanged whether the obstacle sits beside the target or lies across the path, which reflects the insensitivity to scene layout discussed above. The vanilla agent adapts more readily, varying its approach routes and retrying after failures, yet it still collides frequently. Its traces show explicit reasoning about obstacles, but this reasoning is not translated into the plan it generates (Sec. 4.3). The result is a persistent gap between what the agent recognizes and what it executes. **SAFEHARNESS** stays robust under both obstacle placements, maintaining high task success while seldom colliding, which indicates that safety and task completion need not be traded against each other. On physical hardware, this robustness again follows from resolving obstacles at the planning stage. Once the route is committed with the obstacle accounted for, every subsequent step inherits the constraint rather than depending on the agent to keep it in view.

More broadly, these results point to a general principle concerning where safety knowledge resides in an embodied system. When safety is absent from what a system has learned, its behavior can only reflect the regularities of its training distribution, and nothing within it prefers a safe outcome over an unsafe one. When safety is imposed from outside as a constraint on individual actions, it can prevent violations but cannot contribute to accomplishing the task, so safety and progress come into conflict whenever the constraint becomes active. When safety is conveyed only as an instruction, it remains contingent on that instruction staying influential throughout execution, which grows increasingly uncertain as the interaction lengthens. In each case, safety operates apart from the reasoning that produces behavior, whether as a missing signal, a corrective filter, or a reminder. **SAFEHARNESS** instead treats safety as an input to planning: the constraints of the environment are made explicit before any action is taken, and the resulting plan is shaped by them from the outset. Safety thereby becomes a structural property of how the task is solved rather than a condition checked against the solution afterward. This perspective suggests that reliable safe manipulation depends less on stronger external safeguards than on bringing safety into the process by which a plan is formed.

Table 3: Ablation on model family and the source of safety gains. TSR/ CAR/ SSR in %. ***Model***: the language model drives motion and primitives directly. ***Model + Skills***: adds skill memory and the frozen VLA. **SAFEHARNESS**: further adds the obstacle-aware harnesses of Sec. 3. Small numbers give the relative improvement of **SAFEHARNESS** over Model + Skills with the same planner.

		Level I			Level II			All 32		
Planner	Agent	TSR	CAR	SSR	TSR	CAR	SSR	TSR	CAR	SSR
GPT-5.5	Model	43.1	43.1	25.6	19.4	56.9	18.8	31.3	50.0	22.2
	Model + Skills	44.4	18.1	13.1	68.1	69.4	49.4	56.2	43.8	31.3
	SAFEHARNESS	**86.2** +94%	**90.0** +397%	**80.0** +511%	**71.2** +5%	**83.8** +21%	**66.3** +34%	**78.8** +40%	**86.9** +98%	**73.1** +134%
GPT-6-Astra	Model	36.9	50.6	30.6	31.9	74.4	31.3	34.4	62.5	30.9
	Model + Skills	56.9	19.4	11.9	43.1	49.4	25.6	50.0	34.4	18.8
	SAFEHARNESS	**90.0** +58%	**92.5** +377%	**83.8** +604%	**72.5** +68%	**91.2** +85%	**72.5** +183%	**81.2** +62%	**91.9** +167%	**78.1** +315%

Table 4: Component ablation on SafeLIBERO, TSR/ CAR/ SSR in %. Rows add one harness at a time to *Model + Skills* (MS). Small numbers give the relative improvement over *MS + Route*.

		Level I			Level II			All 32		
Planner	Agent	TSR	CAR	SSR	TSR	CAR	SSR	TSR	CAR	SSR
GPT-5.5	Model + Skills (MS)	44.4	18.1	13.1	68.1	69.4	49.4	56.2	43.8	31.3
	MS + Contact	55.0	38.8	32.5	68.8	70.0	50.0	61.9	54.4	41.3
	MS + Route	75.0	71.3	62.5	71.2	82.5	58.8	73.1	76.9	60.6
	SAFEHARNESS	**86.2** +15%	**90.0** +26%	**80.0** +28%	**71.2** +0%	**83.8** +2%	**66.3** +13%	**78.8** +8%	**86.9** +13%	**73.1** +21%
GPT-6-Astra	Model + Skills (MS)	56.9	19.4	11.9	43.1	49.4	25.6	50.0	34.4	18.8
	MS + Contact Harness	67.5	45.0	40.0	46.3	53.8	38.8	56.9	49.4	39.4
	MS + Route Harness	78.8	72.5	65.0	70.0	87.5	63.8	74.4	80.0	64.4
	SAFEHARNESS	**90.0** +14%	**92.5** +28%	**83.8** +29%	**72.5** +4%	**91.2** +4%	**72.5** +14%	**81.2** +9%	**91.9** +15%	**78.1** +21%

4.3 ABLATION: WHERE THE CODING AGENT FAILS, AND WHAT FIXES IT

We fix the 32 scenes and the ten seeds per task, so that every row rests on 160 episodes per level, and cross two backbones with the three settings of Sec. 4.1. *Model* measures the intrinsic capability of the backbone, and *Model + Skills* tests whether an agent that already completes the tasks can respect a constraint given only as language.

As shown in Tab. 3, a coding agent does not reliably act on a safety constraint it has understood, and adding task skills can widen rather than close this gap. Without external tools, *Model* fails on both metrics, since the backbone alone lacks the execution abilities manipulation requires. Its TSR is lowest at Level II, where transport must be planned from scratch, and its higher CAR there comes from episodes that fail before reaching the obstacle, so safe successes stay rare. Skills raise TSR as expected, yet do not make execution safer: at Level I, CAR and SSR fall below those of *Model* for both backbones. When a stored skill matches the subtask, the agent replays it instead of reconciling it with the constraint, and since these skills were acquired in obstacle-free scenes, replaying them near the target leads to collisions. Transport is rarely covered by a stored skill, so the agent plans that motion from scratch, and mainly then does the constraint enter its reasoning. A constraint that exists only as language is thus honored mostly when no ready-made solution is at hand, so that competence and safety compete rather than reinforce each other; a stronger backbone does not change this, as *Model + Skills* with GPT-6-Astra is the least safe setting in the table.

SAFEHARNESS resolves this tension by making safety part of the planning procedure rather than a sentence in the prompt. Because the agent must ground the obstacles, plan the route, and settle the contact strategy before acting, the constraint shapes execution whether or not a matching skill exists. **SAFEHARNESS** thus raises TSR and CAR together, improving on *Model + Skills* by 22.6 and 43.1 points with GPT-5.5 and by 31.2 and 57.5 points with GPT-6-Astra, and SSR more than doubles. Since both share backbone, tools, and skill memory, the gains isolate the harnesses.

4.4 ABLATION: ROUTE PLANNING VERSUS CONTACT EXECUTION

Sec. 4.3 evaluates the harnesses as a whole; here we ablate the two components of the safety harness, route planning (Sec. 3.2) and contact execution (Sec. 3.3), to quantify the contribution of each. When a component is removed, the corresponding phase is left to the agent itself: *MS + Route* lets the agent perform grasp and release on its own, and *MS + Contact* lets it plan its own transport motions. Two

findings emerge. First, both components are necessary. Removing either one lowers SSR across all base models and both levels, and neither partial configuration reaches the full method, indicating that the two address distinct rather than redundant failure modes. Second, route planning contributes more than contact execution. At the moment of contact, the agent's own grasps typically follow a standard approach direction and pose that often keep the gripper clear of a nearby obstacle, so many contact-phase hazards are avoided without the harness. During transport, however, nothing in the agent's own motion inherently avoids an obstacle on the path, so the obstacle is struck unless the trajectory is explicitly rerouted. This effect extends to Level I as well: even when the obstacle stands beside the target, a tall object such as a wine bottle sweeps the surrounding space as soon as it is lifted, so the motion immediately after grasping already falls within the scope of route planning. The two failure modes of Sec. 3 are therefore distinct, and each harness component removes one of them.

4.5 Qualitative Analysis

Fig. 4 contrasts a failed and a successful rollout on the same SafeLIBERO [16] scene, where the orange juice must reach the basket without touching the wine bottle.The skills-only agent (top row) acknowledges the bottle in its reasoning, but the acknowledgment never shapes the plan that follows, which takes the shortest line at grasp height, straight through the bottle, as if success were defined without the constraint rather than under it. **SafeHarness** (bottom rows) instead clears the bottle and completes the task, because every route is checked before the arm moves rather than left to the model's reasoning. On Libero-10 this loop runs twice per episode, once per sub-goal, and the second leg re-uses the obstacle box measured for the first. Fig. 3 shows a successful rollout on the real robot.



Figure 3: **Real robot experiment demo**. **SafeHarness** leads to safe and successful task completion.

We now walk through the successful rollout phase by phase, each running the full loop of Sec. 3.2 and Sec. 3.3. In the first phase the agent grounds the juice and the bottle as boxes and plans the route to the pick; the bottle blocks the way, so the route detours around it, passes verification, and execution follows the verified plan. In the second phase, with the bottle no longer in the way, the route to the basket proceeds directly and is verified as proposed, so the harness plans a detour only when an obstacle demands one and safety is not bought by timidity. At both contact moments, the adjacency test finds no obstacle beside the target, so contact execution admits the default strategy. The rollout thus exercises every decision point of **SafeHarness**, each intervening exactly where the scene requires it.

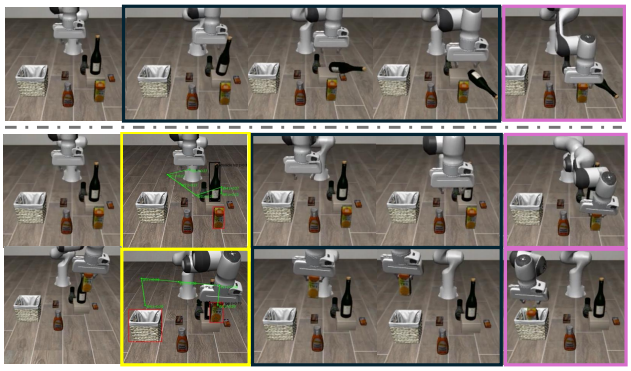


Figure 4: **Rollouts on SafeLIBERO with and without SafeHarness.** The task is to place the orange juice in the basket without hitting the wine bottle. Without **SafeHarness** (top row), hits the obstacle en route to the grasp; with **SafeHarness** (bottom rows), the agent detours around it and finishes collision-free. Panel frames: yellow = route planning (bounding boxes and waypoints), black = verified-route execution, purple = contact-rich moments.

5 Conclusions

Safety remains a critical yet largely overlooked dimension of coding agents for robot manipulation, whose progress has thus far been measured solely by task completion. To address this gap, we present **SafeHarness**, the first coding-agent framework designed for safe robot control. **SafeHarness** comprises two complementary components: obstacle-aware route planning, which plans routes in advance and replans them once they become infeasible, and obstacle-aware contact execution, which keeps contact-rich motion clear of surrounding obstacles. Together, they elevate safety to the same priority as task completion.

AI Disclosure. In this work, we used generative AI tools to polish paper writing and image illustrations. All authors reviewed the polished text against the original draft to ensure that no claim or result was altered, and independently verified that the edited figures accurately depict our method. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility Statement. We are committed to ensuring the reproducibility of our results. The source code will be publicly released upon acceptance. Detailed descriptions of our experimental setup and implementation details are provided in Sec. 4.1.

Ethics Statement. This work focuses on robot manipulation in simulated and controlled laboratory environments, using only publicly available benchmarks and datasets. It does not involve human subjects, personal data, or sensitive information. We do not foresee any direct negative societal impact from this research.

REFERENCES

[1] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *European Control Conference (ECC)*, 2019.

[2] Cem Anil, Esin Durmus, Nina Panickssery, Mrinank Sharma, Joe Benton, Sandipan Kundu, Joshua Batson, Meg Tong, Jesse Mu, Daniel Ford, et al. Many-shot jailbreaking. *Advances in Neural Information Processing Systems*, 2024.

[3] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[6] Lukas Brunke, Yanni Zhang, Ralf Römer, Jack Naimer, Nikola Staykov, Siqi Zhou, and Angela P Schoellig. Semantically safe robot manipulation: From semantic scene understanding to motion safeguards. *IEEE Robotics and Automation Letters*, 2025.

[7] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris Coll-Vinent, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. SAM 3: Segment anything with concepts. In *International conference on learning representations*, 2026.

[8] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[9] Chi-Lam Cheang, Guangzeng Chen, Ya Jing, Tao Kong, Hang Li, Yifeng Li, Yuxiao Liu, Hongtao Wu, Jiafeng Xu, Yichu Yang, et al. GR-2: A generative video-language-action model with web-scale knowledge for robot manipulation. *arXiv preprint arXiv:2410.06158*, 2024.

[10] Zixing Chen, Yifeng Gao, Li Wang, Yunhan Zhao, Yi Liu, Jiayu Li, Xiang Zheng, Zuxuan Wu, Cong Wang, Xingjun Ma, and Yu-Gang Jiang. HazardArena: Evaluating semantic safety in vision-language-action models. *arXiv preprint arXiv:2604.12447*, 2026.

[11] Rongxu Cui, Zongzheng Zhang, Jingrui Pang, Haohan Chi, Jinbang Guo, Saining Zhang, Shaoxuan Xie, Xin Jin, Yao Mu, Jiaolong Yang, et al. Libero-safety: A comprehensive benchmark for physical and semantic safety in vision-language-action models. In *European Conference on Computer Vision*. Springer, 2026.

[12] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B. Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[13] Jialiang Fan, Weizhe Xu, Oleg Sokolsky, Insup Lee, and Fanxin Kong. Safevla-bench: A benchmark for the success-safety gap in vision-language-action models. *arXiv preprint arXiv:2606.00773*, 2026.

[14] Letian Fu, Justin Yu, Karim El-Refai, Ethan Kou, Haoru Xue, Huang Huang, Wenli Xiao, Guanzhi Wang, Dantong Niu, Fei-Fei Li, et al. Cap-x: A framework for benchmarking and improving coding agents for robot manipulation. *arXiv preprint arXiv:2603.22435*, 2026.

[15] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: A survey on detection, isolation, and identification. *IEEE Transactions on Robotics*, 2017.

[16] Songqiao Hu, Zeyi Liu, Shuang Liu, Jun Cen, Zihan Meng, Shihefeng Wang, Xiang Li, and Xiao He. Vlsa: Vision-language-action models with plug-and-play safety constraint layer. *arXiv preprint arXiv:2512.11891*, 2025.

[17] Chengyue Huang, Khang Vo Huynh, Sebastian Elbaum, Zsolt Kira, and Lu Feng. Safemanip: A property-driven benchmark for temporal safety evaluation in robotic manipulation. *arXiv preprint arXiv:2605.12386*, 2026.

[18] Siyuan Huang, Zhengkai Jiang, Hao Dong, Yu Qiao, Peng Gao, and Hongsheng Li. Instruct2act: Mapping multi-modality instructions to robotic actions with large language model. *arXiv preprint arXiv:2305.11176*, 2023.

[19] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023.

[20] Mengzhao Jia, Yang Lin, Xixin Zhang, Zhihan Zhang, Xiaobai Liu, and Meng Jiang. Agent as policy for robotic manipulation. *arXiv preprint arXiv:2609.12541*, 2026.

[21] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[22] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[23] Vaishak Kumar. Act-observe-rewrite: Multimodal coding agents as in-context policy learners for robot manipulation. *arXiv preprint arXiv:2603.04466*, 2026.

[24] Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. Llms get lost in multi-turn conversation. *arXiv preprint arXiv:2505.06120*, 2025.

[25] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. *Research Report 9811*, 1998.

[26] Kenneth Li, Tianle Liu, Naomi Bashkansky, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Measuring and controlling instruction (in)stability in language model dialogs. *arXiv preprint arXiv:2402.10962*, 2024.

[27] Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. arXiv:2209.07753.

[28] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023.

[29] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 2024.

[30] Runyu Lu, Yubo Wu, Ethan Kou, Letian Fu, Wenli Xiao, Ajay Mandlekar, Yinzhen Xu, Guanya Shi, Ken Goldberg, Ang Chen, et al. Aspire: Agentic skills discovery for robotics. *arXiv preprint arXiv:2607.00272*, 2026.

[31] Xiaoya Lu, Zeren Chen, Xuhao Hu, Yijin Zhou, Weichen Zhang, Dongrui Liu, Lu Sheng, and Jing Shao. IS-Bench: Evaluating interactive safety of VLM-driven embodied agents in daily household tasks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2026.

[32] Yuan Meng, Zhenguo Sun, Max Fest, Xukun Li, Zhenshan Bing, and Alois Knoll. Growing with your embodied agent: A human-in-the-loop lifelong code generation framework for long-horizon manipulation skills. *arXiv preprint arXiv:2509.18597*, 2025.

[33] Achu Menon, Sravanthi Machcha, Sabrina Zou, Tzu Kit Chan, and Jay Chooi. GPT-6 Astra on robotic manipulation. Robocurve evaluation report, https://openai.robocurve.org/gpt-6-astra/, 2026. 4 September 2026.

[34] Yao Mu, Junting Chen, Qinglong Zhang, Shoufa Chen, Qiaojun Yu, Chongjian Ge, Runjian Chen, Zhixuan Liang, Mengkang Hu, Chaofan Tao, et al. Robocodex: Multimodal code generation for robotic behavior synthesis. *arXiv preprint arXiv:2402.16117*, 2024.

[35] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024. arXiv:2405.12213.

[36] OpenAI. GPT-6 Astra: A new generation of intelligence. https://openai.com/index/gpt-6-astra/, 2026. Accessed 2026-09-14.

[37] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[38] Nathan Ratliff, Matthew Zucker, J. Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[39] John Schulman, Yan Duan, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, Jia Pan, Sachin Patil, Ken Goldberg, and Pieter Abbeel. Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research*, 2014.

[40] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2023. arXiv:2209.11302.

[41] Jiayi Su, Yixin Zheng, Mi Yan, Li Yi, Zhizheng Zhang, and He Wang. GPT 6 Astra as an embodied policy. Technical report and code, 2026. URL https://github.com/anonymous-report-421/eval-of-gpt-6-astra-as-policy.

[42] Kai Tang, Peidong Jia, Zhong Chu, Jixian Wu, Rui Ma, Jiajun Cao, Fangyuan Zhao, Sixiang Chen, Yichen Guo, Xiaowei Chi, et al. Safedojo: Safe reinforcement learning for vla via interactive world model. *arXiv preprint arXiv:2606.20698*, 2026.

[43] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

[44] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better LLM agents. *arXiv preprint arXiv:2402.01030*, 2024.

[45] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

[46] Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Xiangju Lu, Junmin Zhu, and Wei Zhang. Lifbench: Evaluating the instruction following performance and stability of large language models in long-context scenarios. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

[47] Wenli Xiao, Jia Xie, Tonghe Zhang, Haotian Lin, Letian Fu, Haoru Xue, Jalen Lu, Yi Yang, Cunxi Dai, Zi Wang, et al. ENPIRE: Agentic robot policy self-improvement in the real world. *arXiv preprint arXiv:2606.19980*, 2026.

[48] Bingxin Xu, Yuzhang Shang, and Emilio Ferrara. Don't drop the baton: Long-horizon robot manipulation via agentic subtask exploration and transition-aware memory. *arXiv preprint arXiv:2608.16889*, 2026.

[49] Huaigang Yang, Ya Li, Min Ren, Bo Dai, Zhenliang Zhang, and Zhaofeng He. SafeRelBench: A spatial-relation-aware benchmark for process-level safety in VLM-driven embodied agents. *arXiv preprint arXiv:2607.14543*, 2026.

[50] Borong Zhang, Yuhao Zhang, Jiaming Ji, Yingshan Lei, Yishuai Cai, Josef Dai, Yuanpei Chen, and Yaodong Yang. Safevla: Towards safety alignment of vision-language-action model via constrained learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. Spotlight; arXiv:2503.03480.

[51] Tao Zhang, Kaixian Qu, Zhibin Li, Jiajun Wu, Marco Hutter, Manling Li, and Fan Shi. Using large language models for embodied planning introduces systematic safety risks. *arXiv preprint arXiv:2604.18463*, 2026.

[52] Yixian Zhang, Huanming Zhang, Feng Gao, Xiao Li, Zhihao Liu, Chunyang Zhu, Jiaxing Qiu, Yuchen Yan, Jiyuan Liu, Wenhao Tang, et al. Harness vla: Steering frozen vlas into reliable manipulation primitives via memory-guided agents. *arXiv preprint arXiv:2607.08448*, 2026.

[53] Zhanguang Zhang, Zhiyuan Li, Behnam Rahmati, Rui Heng Yang, Yintao Ma, Amir Rasouli, Sajjad Pakdamansavoji, Yangzheng Wu, Lingfeng Zhang, Tongtong Cao, et al. Do world action models generalize better than vlas? a robustness study. *arXiv preprint arXiv:2603.22078*, 2026.