

What Do Robot Safety Benchmarks Observe? A Paper-Level Audit of Evaluation Oracles in Robot Foundation Models

Anonymous Author(s)
Affiliation
Address
email

Abstract: A robot can receive an unsafe label from evidence it never observes. We examine 36 published evaluator units from six manipulation-safety benchmarks. For each unit, we record the signal it uses and when it can reach a verdict. The units differ. They include temporal templates, scored constraints, damage models, and monitored channels. The published evaluators use simulator-internal state for 35 units. Under a declared RGB, joint-position, gripper, and robot-model interface, 34 cannot reproduce the exact evaluator signal. Two robot-geometry cases remain conditional. Of the 33 units with a pass or fail verdict, 29 admit an early failure witness. We distinguish offline scores from runtime claims.

Keywords: robot foundation models, safety evaluation, observability

1 Introduction

Robot foundation models (RFMs) and vision-language-action policies now receive safety scores as well as task-success scores, while recent benchmarks test collisions, force, object handling, damage, event order, and hazardous instructions [1, 2, 3, 4, 5, 6]. A safety score comes from an oracle: a rule, its evidence, and a way to decide the result. In simulation, the oracle may read exact contact records or object state. The policy may see only images, language, and selected robot signals, so an offline safety label does not by itself establish what a physical runtime monitor could observe.

Prior work questions whether VLA metrics support deployment claims [7, 8]. Runtime verification distinguishes a conclusive failure from a verdict still pending on a finite trace [9]. Robot monitors also depend on the messages a system exposes [10]. We ask what each published safety rule needs to observe and when it can decide. The selected rules are not equal samples of hazards: they include templates, damage models, and monitored channels. For each, we record the evaluator signal, compare it with two declared robot interfaces, and mark its verdict time.

2 Source-defined corpus and method

We selected six manipulation-safety papers that describe their evaluator rules and required signals. We fixed their versions and retained the unit each author defined: a template, scored family, family rule, damage model, constraint class, metric, or monitored channel [1, 2, 3, 4, 5, 6]. We excluded task instances. We also excluded alternate thresholds.

Table 1 lists the sources; Table 2 reports all 36 rows. We selected these papers to compare different oracle designs, not to count the prevalence of hazards across tasks.

Benchmark	IDs	Unit type	Source locator
SafeManip	SM01–10	LTL _f templates	v2, Table 2
SafeVLA-Bench	SV01–08	Scored families	v1, Table 2; App. C
ManiGuard	MG01–07	Family specifications	v1, App. A.3, Table 6
OopsieVerse	OP01–03	Damage models	v1, Sec. III-B
LIBERO-Safety	LB01–03	Physical classes + refusal	v2, App. A.2; Sec. 4.1
ForesightSafety-VLA	FS01–05	Safe-Core channels	v2, Table I; Sec. IV-A

Table 1: Source versions and selected unit types. These groups differ in kind; Table 2 names and codes all 36 units.

We coded the evidence named in each paper, not an evaluator code path. The comparison interface P^* contains RGB images, every controlled arm’s joint position, gripper state, and a calibrated robot model, but excludes force, tactile, object-state, semantic-label, and simulator channels. It does not describe every cited policy’s actual input. The arm-sensing reference R contains joint encoders, effort or torque telemetry, and gripper state, but no tactile or external scene sensor; a wrist force/torque cell is conditional. An R label is partial when these sensors recover only part of the oracle signal. For SV08 and FS04, joint positions and robot geometry could support the stated rule. Their P^* labels remain conditional because the papers do not fix every contact or distance convention needed to match the simulator’s exact verdict for every task.

Timing differs. H marks a need to retain episode state beyond the current sample, apart from a generic latch for an earlier breach. B is Y if some finite prefix proves failure for every continuation, even when other failures remain pending. T asks whether a final pass or complete metric needs the episode endpoint. OopsieVerse’s three damage models produce per-step increments from the current state; a separate health state accumulates those increments. We therefore mark their H as N and leave B and T undefined for the models themselves. The other 33 units have a pass or fail verdict. A rule can fail early and still need the endpoint to be certified as passed. These timing labels are inferences from the published rules, not observations of a deployed monitor.

Findings. In the published descriptions, 35 units use simulator-internal state. LB03 is the exception. Under P^* , 34 cannot match the exact evaluator signal and 2 robot-geometry cases remain conditional. Under R , 1 unit is directly observable, 4 are conditional, and 3 are partial. The remaining physical units need evidence beyond that arm-sensing reference. 13 units need memory to produce their current output. Among the 33 verdict-bearing units, 29 admit an early bad-prefix witness and 4 have none under the stated rule; all 33 need episode completion for a final pass or complete metric. The counts do not measure benchmark coverage.

3 Evidence and transfer

Contact is more specific than load. SafeVLA-Bench checks role-filtered contact force and arm self-contact [2], while LIBERO-Safety checks active contact pairs and the normal force on each gripper element [5]. Those rules read solver identities or forces. A wrist transducer instead reports a resultant wrench. In one sign convention,

$$\mathbf{w}_{\text{wrist}} = \sum_{i=1}^m \mathbf{G}_i(\mathbf{r}_i)\mathbf{f}_i + \mathbf{w}_{\text{gravity}} + \mathbf{w}_{\text{inertia}}. \quad (1)$$

Here, $\mathbf{r}_i$ and $\mathbf{f}_i$ are the location and force of contact i , and $\mathbf{G}_i$ maps that force to the wrist frame. One resultant does not generally identify each force when several contacts are present [11]. Opposing fingertip forces can cancel at the wrist. Joint-torque estimates also depend on dynamics and contact assumptions. A tactile gripper or a calibrated single-contact setup could narrow this gap, but neither is part of the reference arm sensor set.

71 Passive scene and material state. SafeVLA-Bench uses a 5 mm non-target displacement
72 bound, 15-degree held-object tilt, and 2 cm downward slip bound, all of which need object
73 pose and an earlier reference [2]. ManiGuard obtains dropped, upright, support, touching,
74 and spilled propositions from simulator state. Spill detection uses particle loss [3]. OopsieV-
75 erse computes mechanical, thermal, and fluid damage increments from current simulator
76 state, then accumulates them in object health [4]. Joint encoders do not report the ex-
77 act pose or temperature of an uninstrumented object. Images can supply estimates, but
78 matching a threshold requires calibration and an error test near that threshold.

79 Symbolic and temporal state. SafeManip tracks propositions such as contamination, san-
80 itization, release, and settling [1]. ForesightSafety-VLA adds a finite-state precondition
81 channel to its force, thermal, clearance, and dual-arm channels [6]. Keeping the history
82 may be simple: an automaton can remember that a release occurred. The harder step is
83 grounding Settled or a task precondition in measurements. Memory cannot create missing
84 observations.

85 4 Timing is a separate dimension

86 SafeVLA-Bench scores stored trajectories after rollout, although a force limit could be
87 breached at one sample if its force signal were available then [2]. The evaluator’s sched-
88 ule and the rule’s verdict time differ. SafeManip’s release-stability rule requires eventual
89 settling, leaving an unsettled release pending until termination if no deadline is given [1].
90 Its SM08 and SM10 strong-until rules can fail earlier when insertion or release occurs before
91 the required event. ManiGuard’s lid and jar rules instead use weak until, so the right-hand
92 event need not occur [3]. A monitor cannot assign these rules the same early verdict.

93 5 Implications

94 An offline oracle need not be limited to the policy’s input. Its privileged signals can expose
95 failures the policy misses. A physical monitor, however, must have its own evidence. Bench-
96 mark authors can make the difference checkable by listing each rule’s signal source, contact
97 filter, threshold, sample period, and task applicability alongside the policy input contract.
98 A physical monitor also needs a sensor list and an error test near each threshold. Temporal
99 results should distinguish alarms raised during an episode from the final trace verdict.

100 6 Limitations

101 This audit reads six paper versions rather than frozen evaluator code or a physical robot, and
102 the declared reference interfaces do not describe verified inputs for every cited policy. Ge-
103 ometry and calibration affect some observability labels. No second coder participated. The
104 unlike units and purposive selection make these counts unsuitable as prevalence estimates
105 for tasks, hazards, or deployed robots.

106 7 Conclusion

107 We record what each selected evaluator reads and when a verdict becomes possible. Most
108 use simulator state, while 29 verdict-bearing units can witness failure before an episode ends.
109 A physical monitor must acquire the corresponding evidence before it can make the same
110 claim. Benchmark authors can state that requirement beside each reported safety score by
111 listing the signal and verdict time.

ID	Published unit	Oracle signal	P^*	R	S	H	B	T
SM01	Collision and bad contact	contact, task event	N	N	Y	N	Y	Y
SM02	Grasp stability	object state	N	P	Y	Y	Y	Y
SM03	Release stability	object state	N	N	Y	Y	N	Y
SM04	Cross-contamination	contact, task event	N	N	Y	Y	Y	Y
SM05	Action onset	task event, object state	N	N	Y	N	Y	Y
SM06	Mechanism recovery	contact	N	P	Y	Y	N	Y
SM07	Containment	object state	N	N	Y	Y	N	Y
SM08	Enclosure clearing	object state	N	N	Y	Y	Y	Y
SM09	Enclosure access	object state, scene geometry	N	P	Y	N	Y	Y
SM10	Release inside enclosure	object state	N	N	Y	Y	Y	Y
SV01	Arm-furniture force	solver force	N	C	Y	N	Y	Y
SV02	Maximum contact force	solver force	N	N	Y	N	Y	Y
SV03	Non-target displacement	object pose	N	N	Y	Y	Y	Y
SV04	Target-furniture force	solver force	N	N	Y	N	Y	Y
SV05	Held-object tilt	object pose	N	N	Y	Y	Y	Y
SV06	Stable grasp	object pose	N	N	Y	Y	Y	Y
SV07	Joint torque	joint effort	N	Y	Y	N	Y	Y
SV08	Self-collision free	contact pair	C	C	Y	N	Y	Y
MG01	Clutter, dry	object state	N	N	Y	N	Y	Y
MG02	Clutter, liquid	fluid state, object state	N	N	Y	N	Y	Y
MG03	Cabinet	object state	N	N	Y	N	Y	Y
MG04	Lid	object state	N	N	Y	Y	Y	Y
MG05	Stack	object state	N	N	Y	N	Y	Y
MG06	Jar	object state	N	N	Y	Y	Y	Y
MG07	Dusty	contact, object state	N	N	Y	N	Y	Y
OP01	Mechanical damage	solver force, object state	N	N	Y	N	NA	NA
OP02	Thermal damage	temperature	N	N	Y	N	NA	NA
OP03	Fluid damage	fluid state	N	N	Y	N	NA	NA
LB01	Spatial collision constraint	contact pair	N	N	Y	N	Y	Y
LB02	Dynamic wrench constraint	solver force	N	N	Y	N	Y	Y
LB03	Hazardous-instruction refusal	response, prompt label	N	NA	N	N	N	Y
FS01	Force/Torque	solver force, joint effort	N	C	Y	N	Y	Y
FS02	Thermal/Energy	thermal region, scene geometry	N	N	Y	N	Y	Y
FS03	Spatial Boundary	scene geometry, object pose	N	N	Y	N	Y	Y
FS04	Collaborative	robot geometry	C	C	Y	N	Y	Y
FS05	Temporal Sequence	task event	N	N	Y	Y	Y	Y

Table 2: Coding of 36 selected units. P^* : exact evaluator match under the declared RGB, joint, gripper, and robot-model interface; R : reference arm observability; S : simulator state; H : stored state for the current output; B : early bad-prefix witness; T : endpoint needed for a final pass or metric. Y/N/C/P mean yes/no/conditional/partial. For per-step damage models OP01–03, H is N and B, T are NA. The refusal metric LB03 has no physical-sensing rating ($R = \text{NA}$). Source locators appear in Table 1.

References

- [1] C. Huang, K. V. Huynh, S. Elbaum, Z. Kira, and L. Feng. SafeManip: A property-driven benchmark for temporal safety evaluation in robotic manipulation. arXiv preprint arXiv:2605.12386, 2026.
- [2] J. Fan, W. Xu, O. Sokolsky, I. Lee, and F. Kong. SafeVLA-Bench: A benchmark for the success-safety gap in vision-language-action models. arXiv preprint arXiv:2606.00773, 2026.
- [3] Y. Peng, P. Wang, S. S. Zhan, Y. Lyu, Z. Ni, J. Yan, F. Wong, R. Jiao, H. Yin, X. Cao, H. Shao, M. Li, R. Zhang, and Q. Zhu. MANIGUARD: A benchmark and data suite for specification-grounded safety evaluation and improvement of robotic manipulation. arXiv preprint arXiv:2608.17386, 2026.
- [4] A. Balaji, A. Bahety, S. Ambatipudi, D. Lam, J. Xu, and R. Martín-Martín. OopsieVerse: A safety benchmark with damage-aware simulation for robot manipulation. arXiv preprint arXiv:2606.31993, 2026.
- [5] R. Cui, Z. Zhang, J. Pang, H. Chi, J. Guo, S. Zhang, S. Xie, X. Jin, Y. Mu, J. Yang, G. Yao, X. Zhan, Y.-Q. Zhang, and H. Zhao. LIBERO-Safety: A comprehensive benchmark for physical and semantic safety in vision-language-action models. arXiv preprint arXiv:2606.23686, 2026.
- [6] M. Lyu, Y. Sun, Y. Jia, S. Shen, M. Sha, H. Li, F. Zhao, and Y. Zeng. ForesightSafety-VLA: A unified diagnostic safety benchmark for vision-language-action models. arXiv preprint arXiv:2606.27079, 2026.
- [7] Z. Song, H. Liu, X. Wang, S. Li, X. Li, Y. Ji, L. Gao, J. Zhang, X. Meng, X. Chang, and X. Chen. The evaluation bottleneck of vision-language-action models: A evaluation-centric survey. Preprints.org, 2026. [doi:10.20944/preprints202606.0425.v1](https://doi.org/10.20944/preprints202606.0425.v1).
- [8] S. Han and H. Choi. ROEP: A robotics-oriented evaluation protocol for deployment-facing vision-language-action manipulation policies. *Sensors*, 26(15):4757, 2026. [doi:10.3390/s26154757](https://doi.org/10.3390/s26154757).
- [9] A. Bauer, M. Leucker, and C. Schallhart. Runtime verification for LTL and TLTL. *ACM Transactions on Software Engineering and Methodology*, 20(4):14:1–14:64, 2011. [doi:10.1145/2000799.2000800](https://doi.org/10.1145/2000799.2000800).
- [10] J. Huang, C. Erdogan, Y. Zhang, B. Moore, Q. Luo, A. Sundaresan, and G. Rosu. ROSRV: Runtime verification for robots. In *Runtime Verification*, volume 8734 of *Lecture Notes in Computer Science*, pages 247–254, 2014. [doi:10.1007/978-3-319-11164-3_20](https://doi.org/10.1007/978-3-319-11164-3_20).
- [11] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.