TEXTMANI: A CONSTRAINT-BASED FRAMEWORK FOR EVALUATING AND REPAIRING LLM REASONING FROM LANGUAGE TO ROBOTIC ACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Translating large language model (LLM) reasoning into robotic actions requires an interface between semantics and execution. This interface should make intermediate decisions inspectable and editable.Moreover, task success rates alone provide limited insight into task progress and points of failure throughout this process. We introduce TextMani, a unified framework for language-to-action execution, evaluation, and repair that uses explicit constraints as a shared intermediate representation. Given natural language instructions and scene features, TextMani generates execution targets through subgoal decomposition, feature selection, constraint formulation, and mathematical solving. Planning and control modules then translate these targets into robot actions. By linking intermediate artifacts to execution evidence, the framework supports process-level analysis based on physical subgoal progress and enables targeted repair following failures. Guided by execution feedback, it selects stages or constraints for modification, preserves unaffected content, and updates dependent downstream artifacts before re-execution. We evaluate 15 models across 18 simulated tasks and 10 models across six real-world robotic tasks. On real robots, TextMani improves task success from 27.2% with direct numerical-target prediction to 36.3%. Feedback-driven repair is also effective: Task-specialist repair recovers 47.7% of failed tasks compared with 24.5% for self-repair. Blind diagnosis reveals that failure recognition is the primary bottleneck.These results show that explicit executable specifications provide a common interface not only for comparing heterogeneous foundation models, but also for exposing where execution fails and enabling targeted physical recovery.

1 INTRODUCTION

Large language models (LLMs) and multimodal foundation models are increasingly used as high-level decision makers for embodied agents. Given a natural-language instruction and a representation of the task scene, these models can decompose tasks, reason about objects and spatial relations, and propose robot behaviors. Existing systems have connected such reasoning to physical control through skill selection, executable programs, language-mediated planning, and spatially grounded representations (Ichter et al., 2023; Liang et al., 2023; Huang et al., 2023b;a). Yet understanding a task semantically is not the same as specifying an executable robot action. Physical execution ultimately requires concrete geometric quantities, motion targets, and control interfaces, while a free-form reasoning leaves the mapping from semantic decisions to physical behavior largely implicit. This gap becomes particularly important when execution fails. Two models may both fail the same task while exhibiting fundamentally different behavior: one may generate a physically executable plan and reach the final interaction, whereas another may already fail when selecting task-relevant physical evidence or specifying the desired relation. Fine-grained agent benchmarks have similarly shown that final success alone can obscure intermediate progress and interactive capabilities (Liu et al., 2024; Ma et al., 2024; Wang et al., 2024). Moreover, using action trajectories alone as process-level evidence can encourage shortcut matching to interface- or sequence-specific patterns and can misjudge alternative yet valid executions, motivating evaluation grounded in task-relevant states and subgoals rather than trajectory similarity alone (Gu et al., 2026; Kim et al., 2025; Yow et al., 2024). Process-level evaluation therefore requires intermediate representations that are both interpretable and operational: they should expose task-relevant decisions while mapping unambiguously to executable robot behavior.Recent work has begun to use structured constraints as

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.

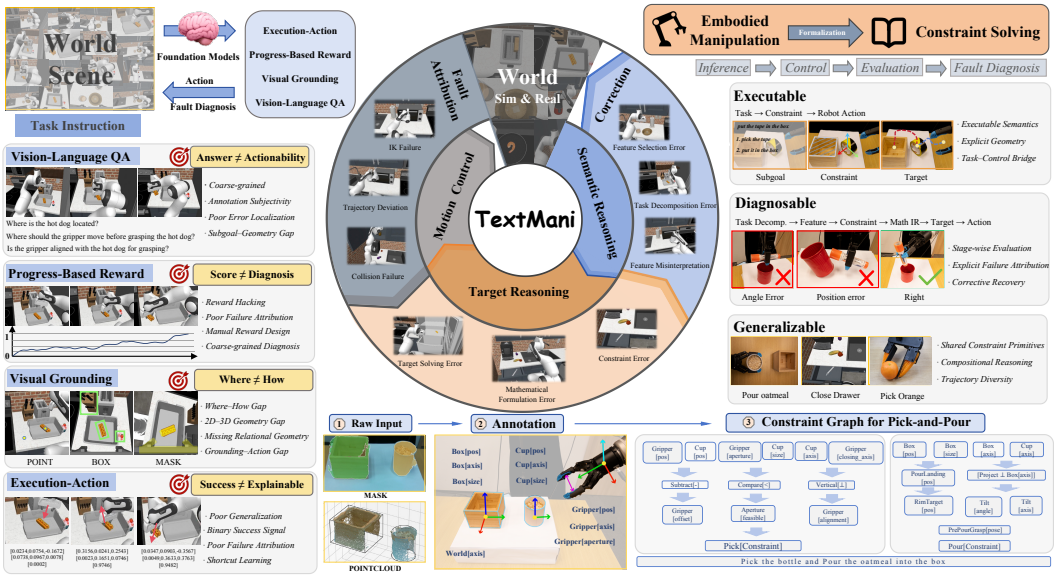


Figure 1: **Overview of TextMani.** *(Left)*Existing interfaces leave the mapping from semantic reasoning to physical execution largely implicit. *(Center)*TextMani makes this mapping explicit through constraints, coupling forward execution with backward diagnosis and repair. *(Right)*The resulting interface is executable, diagnosable, and generalizable, with explicit constraints grounded from scene observations.

Table 1: Comparison of structured constraint interfaces for language-guided robot manipulation; detailed evaluation criteria and method-specific distinctions are provided in Appendix.

Method	Gen.	Sep.	SG-Prog.	Diag.	Attr.	Repair	X-Model
VoxPoser (Huang et al., 2023a)	✓	–	–	–	–	–	–
ReKep (Huang et al., 2025)	✓	–	–	–	–	–	–
GeoManip (Tang et al., 2025)	✓	✓	–	–	–	✓[a]	–
PRoC3S (Curtis et al., 2025)	✓	–	–	–	–	✓[b]	–
TextMani	✓	✓	✓	✓	✓	✓[c]	✓

interfaces between foundation-model reasoning and robot control. VoxPoser grounds language-inferred affordances and constraints into 3D value maps (Huang et al., 2023a); ReKep generates relational keypoint cost functions and optimizes robot actions satisfying them (Huang et al., 2025); GeoManip generates stage-specific geometric constraints for trajectory optimization (Tang et al., 2025); and PRoC3S combines LLM-generated parameterized programs with continuous constraint satisfaction (Curtis et al., 2025). Despite this progress, three questions remain open. First, existing constraint-based manipulation systems are typically instantiated and evaluated with a specific foundation model, while differing in their representations, solvers, and execution interfaces, leaving no common model-agnostic framework for translating foundation-model reasoning into physically executable specifications. (Huang et al., 2023a; 2025; Tang et al., 2025; Curtis et al., 2025).Second, we still lack a systematic understanding of how current foundation models perform when evaluated under the same executable interface: whether they can identify task-relevant physical evidence, formalize the intended relations, and produce downstream planning and execution. Third, even when such specifications can be executed, it remains unclear how well these models can recognize, localize, and repair their own specification errors after physical execution fails.To address these questions, we introduce **TextMani**, a unified constraint-based framework for language-guided manipulation. TextMani explicitly decomposes the manipulation problem into task-conditioned physical evidence, physical subgoals, semantic constraints, executable mathematical representations, and instantiated robot targets.These intermediate artifacts are retained throughout execution, making the language-to-action process inspectable and quantitatively evaluable stage by stage, while enabling execution failures to be localized and repaired using execution evidence.By standardizing these intermediate representations and the downstream execution interface across models, TextMani decouples the evaluation protocol from model-specific architectures, allowing diverse foundation models to

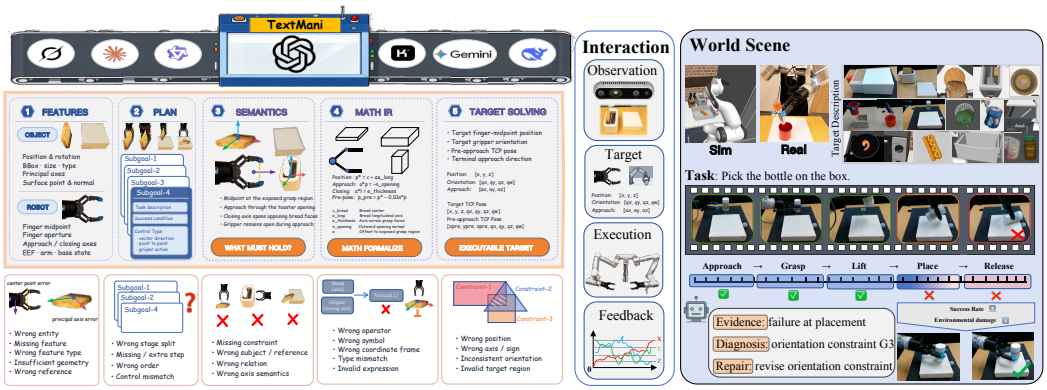


Figure 2: **Overview of TextMani.** *(Left)*A shared constraint-based interface structures manipulation from task-relevant physical evidence to executable robot targets. *(Right)*Execution evidence is traced back to addressable specification components for failure localization and scoped repair, closing the execution–diagnosis–repair loop.

be compared under the same specification and execution pipeline.Detailed comparisons with prior constraint-based manipulation and failure-recovery approaches are provided in Appendix. We evaluate these questions through three complementary settings: executable specification generation, blind failure diagnosis, and scoped specification repair.We analyze execution, diagnosis, and repair as distinct but related capabilities, asking whether models that generate stronger initial plans are also better at recognizing and correcting failures. Our evaluation covers 15 frontier-model configurations in simulation and 10 models across six real-robot manipulation tasks spanning grasp-and-place, insertion, pouring, and articulated-object interaction.

Our contributions are as follows:

- **A common constraint-based language-to-action interface.** We introduce TextMani, which decomposes manipulation into physical subgoals, task-conditioned evidence, semantic constraints, executable mathematical representations, and state-conditioned targets, providing a common execution interface across heterogeneous foundation models.
- **A unified execution-grounded diagnosis and repair protocol.** We establish a common failure-handling protocol that aligns execution evidence with failure recognition and localization, and uses a unified constraint schema to support constraint-level attribution, traceable revision, and localized repair under a consistent evaluation protocol.
- **A systematic multi-model evaluation and capability analysis.** We evaluate frontier foundation models in simulation and on real robots across execution, diagnosis, localization, and repair, and characterize their capability profiles across models and manipulation settings.

2 Manipulation as Executable Formalization

2.1 Forward Modeling: From Language to Executable Manipulation

We formulate embodied manipulation as a *gripper-centric physical constraint-solving problem* conditioned on a task instruction u and the observed physical state x. Let $\mathcal{M} = \{f_1, \ldots, f_N\}$ denote the foundation models. TextMani allows each model to independently decompose the task into its own ordered sequence of physically meaningful subgoals:

$$\mathcal{S}^{\mathrm{SG},(m)} = \Phi_{\mathrm{SG}}\left(f_m; u, x_1\right) = \left(s_1^{(m)}, \ldots, s_{K_m}^{(m)}\right), \qquad m = 1, \ldots, N. \tag{1}$$

Each $s_k^{(m)}$ defines an intermediate physical objective, with boundaries induced by changes in physical objective, control mode, contact mode, or grasp state.

For each model-specific subgoal, TextMani factorizes the specification into three explicit stages: task-relevant physical evidence ($F_k^{(m)}$), semantic manipulation constraints ($C_k^{(m)}$) defined over that evidence, and an executable mathematical IR ($M_k^{(m)}$) that formalizes these constraints for numerical computation. Although their contents may differ across models and subgoals, all model-generated specifications conform to the same structured specification space $\mathcal{Z}$:

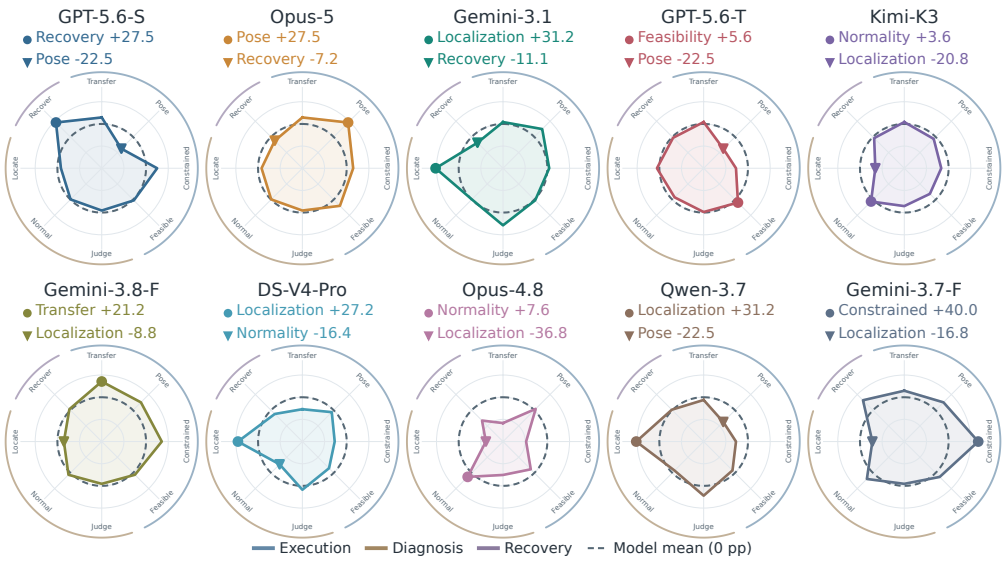


Figure 3: Relative capability profiles of ten real-robot models across execution, diagnosis, and repair dimensions, with detailed dimension definitions and evaluation protocols provided in Appendix.

$$Z_k^{(m)} = f_m\left(u, s_k^{(m)}, x_k^{(m)}\right) = \left(F_k^{(m)}, C_k^{(m)}, M_k^{(m)}\right) \in \mathcal{Z}, \qquad m=1,\ldots,N,\ k=1,\ldots,K_m. \tag{2}$$

Across independently generated subgoals, these outputs therefore constitute model-specific solutions expressed under a common specification contract.

$$\mathbf{Z} = \left\{ Z_k^{(m)}; \middle| ; m = 1, \ldots, N, ; k = 1, \ldots, K_m \right\} \subseteq \mathcal{Z}. \tag{3}$$

This common contract defines a fixed boundary between model reasoning and physical execution:

$$\begin{aligned} T_k^{(m)} &= \mathcal{B}_{\text{shared}}\left(Z_k^{(m)}, x_k^{(m)}\right), \\ a_k^{(m)} &= \pi_{\text{ctrl}}\left(T_k^{(m)}, x_k^{(m)}\right), \qquad m=1,\ldots,N, k=1,\ldots,K_m. \end{aligned} \tag{4}$$

Here, $T_k^{(m)}$ denotes the numerical robot target instantiated from $Z_k^{(m)}$, $a_k^{(m)}$ the resulting control action, and $\mathcal{B}_{\text{shared}}$ denotes the common state-grounding and numerical-solving backend. This separation isolates model-specific reasoning from downstream computation, enabling TextMani to compare foundation models under a common grounding, solving, and control pipeline.

2.2 Backward Modeling: From Execution Failure to Scoped Correction

The backward process traces a failed execution back through the specification that produced it—from the executed target to its mathematical realization and semantic constraint. Unique constraint addresses make this reverse trace explicit, enabling specification-level localization.To make this reverse trace addressable, each constraint retains a unique identity across its semantic and executable forms. For model m, subgoal k, and constraint identifier i, let

$$q_{k,i}^{(m)} = (m, k, i), \qquad g_{k,i}^{(m,r),[q_{k,i}^{(m)}]} \xrightarrow{\text{Compile}} \mu_{k,i}^{(m,r),[q_{k,i}^{(m)}]}, \tag{5}$$

where r indexes the execution–repair round. The address $q_{k,i}^{(m)}$ is shared by a semantic constraint and its executable realization, providing a stable reference across revision rounds.

Given execution evidence $E^{(m,r)}$ and the corresponding specification $Z^{(m,r)}$, diagnosis identifies the subgoal and constraint-address set requiring intervention:

$$\left(\hat{k}^{(m,r)}, \hat{\mathcal{Q}}^{(m,r)}\right) = \mathcal{D}_{\text{shared}}\left(E^{(m,r)}, Z^{(m,r)}\right). \tag{6}$$

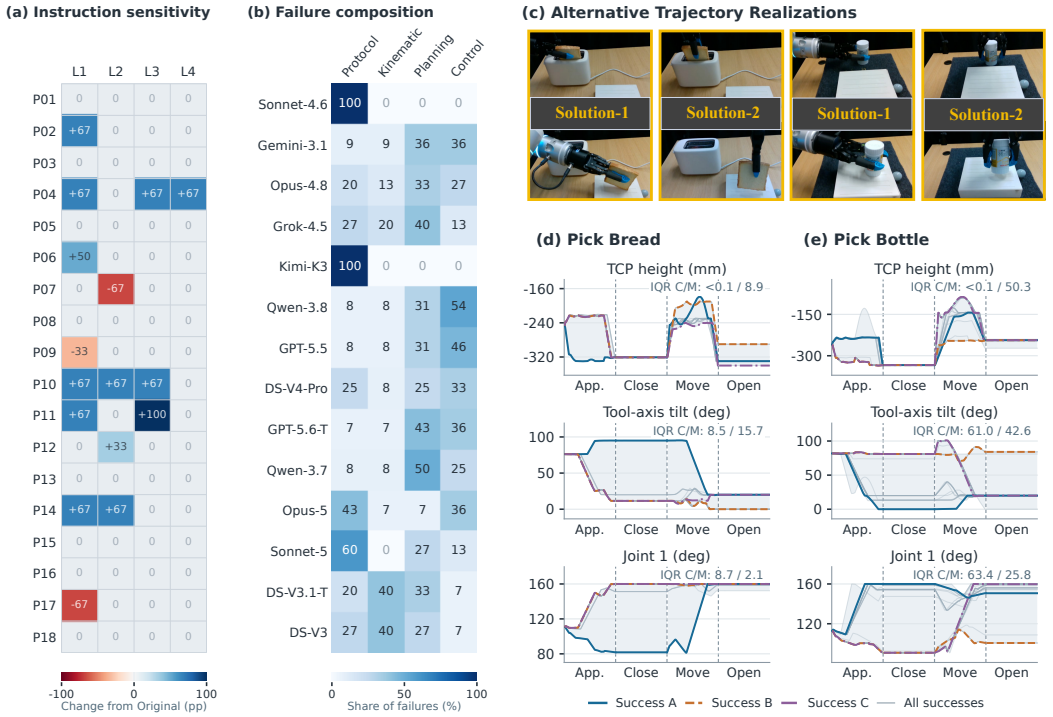


Figure 4: (a) Instruction-induced changes in physical-subgoal progress across 18 simulated tasks. (b) Initial failure composition across 15 models. (c) Alternative successful real-robot executions. (d,e) Phase-aligned trajectories for successful Pick Bread and Pick Bottle executions, showing selective convergence in task-critical variables while preserving motion diversity. Details are provided in Appendix.

The diagnosed scope is then applied to the specification that produced the failed execution:

$$Z_{\hat{k}}^{(m,r+1)} = \mathcal{D}_{\text{shared}}\left(Z_{\hat{k}}^{(m,r)}, E^{(m,r)}; \hat{\mathcal{Q}}^{(m,r)}\right), \tag{7}$$

while constraints outside the diagnosed scope are preserved:

$$g_{\hat{k},j}^{(m,r+1)} = g_{\hat{k},j}^{(m,r)}, \qquad \forall, q_{\hat{k},j}^{(m)} \notin \hat{\mathcal{Q}}^{(m,r)}. \tag{8}$$

Thus, diagnosis converts execution evidence into an addressable intervention scope, and repair revises that scope while preserving unaffected constraints. The backward model therefore mirrors the forward model: forward execution maps a structured specification to physical behavior, whereas backward correction maps observed execution evidence back to a scoped specification revision.

3 Experiments

We evaluate TextMani along three capabilities enabled by the shared specification interface: whether heterogeneous foundation models can produce executable specifications under a common grounding and control pipeline, whether execution evidence supports blind failure recognition and localization, and whether a known failure scope can be revised to improve subsequent execution. Simulation provides process-level evaluation of initial execution and multi-round recovery, while real-robot experiments evaluate physical execution, comparison with direct target prediction, blind diagnosis, and known-subgoal repair. We further analyze successful execution diversity, instruction sensitivity, and runtime overhead.

3.1 Experimental Setup

Simulation Benchmark. We build a custom 18-task language-conditioned manipulation benchmark using RoboCasa assets and simulation environments, covering object relocation, constrained

Table 2: Multi-model simulation performance across initial execution, recovery budgets, repair effects, and downstream execution stages. **Best**/**Second best** denote the top two results within each comparison group. Full metric definitions and evaluation details are provided in Appendix.

Model	Initial		≤RA1 Best		≤RA2 Best		Repair Effects					Stage Statistics			
	SGP ↑	TSR ↑	SGP ↑	TSR ↑	SGP ↑	TSR ↑	FRR ↑	AR ↑	ΔP ↑	RA1-AR ↑	RA2-AR ↑	IK ↑	P—IK ↑	E—P ↑	G—C ↑
GPT-5.6-Sol	20.8	**5.6**	24.5	5.6	28.3	5.6	0.0	6.1	+7.8	5.9	6.2	78.8	53.7	59.1	83.3
GPT-5.6-Terra	24.1	**5.6**	24.1	5.6	35.2	**16.7**	**11.8**	9.1	+11.8	0.0	**18.8**	81.1	58.1	56.0	85.0
GPT-5.5	24.5	**5.6**	24.5	5.6	35.8	**16.7**	**11.8**	8.8	+11.8	0.0	17.6	84.7	60.0	53.3	77.3
Claude Opus 5	16.7	**5.6**	18.5	**11.1**	24.1	**16.7**	**11.8**	7.1	+7.8	5.9	9.1	**92.1**	60.0	57.1	76.9
Claude Opus 4-8	20.4	**5.6**	27.8	**16.7**	38.9	**16.7**	**11.8**	**16.7**	**+19.6**	11.8	**23.1**	87.8	60.5	**69.2**	75.0
Claude Sonnet 5	14.8	0.0	22.2	**11.1**	22.2	11.1	5.6	7.4	+5.6	11.1	0.0	89.7	53.8	57.1	**92.9**
Claude Sonnet 4.6	**29.6**	**11.1**	**37.0**	**11.1**	**44.4**	**22.2**	**12.5**	**12.9**	**+16.7**	**12.5**	13.3	87.3	62.5	63.3	85.7
Gemini 3.1 Pro	**29.6**	**11.1**	35.2	**16.7**	**40.7**	**22.2**	**12.5**	9.7	+12.5	6.2	13.3	88.7	**72.3**	44.1	91.3
Kimi K3	**31.5**	**5.6**	31.5	5.6	38.9	5.6	0.0	6.2	+7.8	0.0	13.3	**96.4**	61.1	54.5	81.5
Qwen3.8 Max	24.1	**5.6**	25.9	**11.1**	38.9	**16.7**	**11.8**	11.8	+15.7	5.9	17.6	82.8	62.5	46.7	86.4
Qwen3.7 Max	24.1	**5.6**	27.8	5.6	29.6	11.1	5.9	6.2	-2.0	5.9	6.7	84.2	**66.7**	65.6	71.4
DeepSeek V4 Pro	20.4	**5.6**	29.6	**11.1**	35.2	**16.7**	**11.8**	12.5	+15.7	11.8	13.3	81.6	65.0	50.0	84.2
DeepSeek V3.1	14.8	0.0	14.8	0.0	18.5	0.0	0.0	3.3	+3.7	0.0	8.3	61.4	40.7	**72.7**	90.0
DeepSeek V3	7.4	0.0	7.4	0.0	7.4	0.0	0.0	0.0	+0.0	0.0	0.0	57.1	45.0	55.6	50.0
Grok 4.5	24.1	**5.6**	**38.9**	**16.7**	38.9	**16.7**	**11.8**	11.8	+15.7	**23.5**	0.0	86.8	65.2	43.3	**91.7**
Pooled	**21.8**	**5.2**	**26.0**	**8.9**	**31.8**	**13.0**	**7.8**	**8.7**	**+9.9**	**6.6**	**11.2**	**83.2**	**60.6**	**55.5**	**82.7**

insertion/extraction, semantic placement, and articulated-object interaction. For each task, SceneLock fixes the instruction, scene annotations, simulator state, object configuration, and robot initialization across the 15 evaluated foundation-model configurations. We evaluate initial execution and recovery under fixed repair budgets using physical-subgoal progress and task success.

Real-Robot Benchmark. The real-robot benchmark contains six tasks—drawer opening/closing, bottle and bread grasping, pen insertion, and cereal pouring—evaluated on a common panel of 10 models. TextMani and Direct use the same downstream validation and execution components; for each model–task pair, we conduct five TextMani trials and three Direct trials, yielding 300 and 180 execution conditions, respectively.

3.2 Can Foundation Models Produce Executable Specifications?

We first examine whether heterogeneous foundation models can produce physically executable specifications through the shared TextMani interface. We evaluate this question from four perspectives: multi-model execution performance, comparison with direct numerical-target prediction, diversity among successful executions, and sensitivity to task specification.

Multi-Model Execution Performance. Across the common 10-model real-robot panel, TextMani achieves a task success rate of (36.3%), with Claude Opus 5 and Gemini 3.7 Flash both reaching (60.0%). In the original simulation cohort, initial execution reaches (21.8%) consecutive GT-Subgoal progress and (5.2%) task success. These results establish that heterogeneous foundation models can generate specifications that are physically executable through the same TextMani interface, while the gap between partial progress and full-task success highlights the additional information captured by process-level evaluation.Stage-wise statistics show that execution loss is distributed across multiple downstream stages. Among 740 simulation planning records, (83.2%) yield valid IK solutions; of these, (60.6%) produce valid paths, and (55.5%) of valid-path records successfully complete the corresponding subgoal. This decomposition shows that reaching a physically grounded target does not guarantee successful downstream planning or execution, motivating stage-wise evaluation beyond terminal task success.

Comparison with the Direct-Target Workflow. This comparison examines whether the shared constraint interface can more effectively translate language-model outputs into successful real-world manipulation. Over the common set of models and tasks, TextMani achieves an overall real-robot success rate of 36.3%, compared with 27.2% for Direct, corresponding to a 9.1 percentage-point improvement. This advantage is observed for 8 of the 10 models. These results show that the TextMani workflow, which combines constraint representation, target solving, and kinematic verification, can not only interface with different models but also achieve a higher task completion rate than directly predicting numerical targets on this set of tasks, providing evidence for the practical execution value of the shared interface.The improvement reaches 20.0 percentage points for grasp-and-transport tasks and 12.0 percentage points for alignment/pose tasks, indicating that the advantage of

Table 3: Real-robot comparison across execution, blind diagnosis, and known-subgoal repair on the common 10-model panel. **Best**/**Second best** denote the top two results within each comparison group; metric definitions, denominators and filtering rules are provided in Appendix.

Model	Execution (TextMani)				Execution (Direct)				Inspection			Repair (Self)			Repair (Specialist)		
	TSR ↑	G&P ↑	A&O ↑	C-P/P ↑	TSR ↑	G&P ↑	A&O ↑	C-P/P ↑	Acc. ↑	FPR ↓	JSG ↑	LPR ↑	TRR ↑	ΔP ↑	LPR ↑	TRR ↑	ΔP ↑
GPT-5.6-Sol	40.0	**70.0**	0.0	**50.0**	22.2	33.3	0.0	33.3	64.0	16.0	44.0	58.8	52.9	+52.9	52.9	47.1	+42.7
GPT-5.6-Terra	26.7	**70.0**	0.0	10.0	11.1	16.7	0.0	16.7	66.0	20.0	52.0	33.3	23.8	+33.3	47.6	47.6	+55.6
Claude Opus 5	**60.0**	**80.0**	**50.0**	**50.0**	27.8	16.7	**50.0**	16.7	64.0	12.0	44.0	18.2	18.2	+30.0	63.6	63.6	+40.0
Claude Opus 4-8	13.3	0.0	40.0	0.0	11.1	33.3	0.0	0.0	52.0	**0.0**	12.0	5.6	5.6	+3.8	27.8	27.8	+33.3
Gemini 3.1 Pro	**43.3**	40.0	40.0	**50.0**	33.3	**66.7**	**33.3**	0.0	**84.0**	16.0	**80.0**	14.3	14.3	+8.3	14.3	14.3	+9.1
Gemini 3.8 Flash	**43.3**	**70.0**	40.0	20.0	**66.7**	**66.7**	**33.3**	**100.0**	64.0	**4.0**	40.0	72.7	27.3	+50.0	54.5	45.5	+50.0
Gemini 3.7 Flash	**60.0**	60.0	**60.0**	**60.0**	**44.4**	**50.0**	0.0	**83.3**	64.0	**0.0**	32.0	55.6	44.4	+33.3	55.6	55.6	+44.4
Kimi K3	36.7	**80.0**	20.0	10.0	16.7	0.0	16.7	33.3	58.0	8.0	28.0	23.1	23.1	+33.3	30.8	30.8	+23.3
Qwen3.7 Max	20.0	50.0	0.0	10.0	16.7	33.3	0.0	16.7	**80.0**	20.0	**80.0**	42.1	26.3	+37.5	78.9	73.7	+75.5
DeepSeek V4 Pro	20.0	30.0	20.0	10.0	22.2	33.3	16.7	16.7	72.0	32.0	**76.0**	18.2	18.2	+13.0	63.6	63.6	+65.7
Pooled	**36.3**	**55.0**	**27.0**	**27.0**	**27.2**	**35.0**	**15.0**	**31.7**	**66.8**	**12.8**	**48.8**	**32.3**	**24.5**	**+29.3**	**49.7**	**47.7**	**+47.1**

TextMani is more pronounced under these manipulation requirements. In contrast, performance on constrained push/pull tasks is 4.7 percentage points lower, and Gemini 3.8 Flash also performs better with Direct than with TextMani (66.7% vs. 43.3%), showing that the gains are both task- and model-dependent. Overall, TextMani provides a more effective language-to-action workflow than direct numerical-target prediction on this benchmark, improving real-robot task success across most evaluated models. The results suggest that explicitly structuring model outputs before numerical grounding and execution is particularly beneficial for grasp-and-transport and alignment/pose manipulation, while the remaining task- and model-dependent exceptions indicate that the gain should be attributed to the complete workflow rather than to any single constraint component.

Diversity of Successful Trajectories. Successful executions show selective convergence, not global trajectory convergence. Across 34 successful bottle-grasping and 21 successful bread-grasping executions, trajectories are aligned by approach, gripper closing, transport, and opening phases. As shown in Figure 4(d,e), task-critical variables can become tightly concentrated at specific interaction events—for example, TCP height around gripper closure—while other variables, including tool-axis tilt and joint configuration, remain substantially dispersed across successful executions.Crucially, convergence in one task-relevant dimension does not collapse the motion to a unique trajectory. The highlighted successful executions reach the same task outcome through visibly different orientations, joint configurations, and intermediate paths. Thus, TextMani's constraint-based formulation is compatible with a set of successful realizations: constraints restrict the physical relations required for task completion while leaving unconstrained degrees of freedom and intermediate motion available for alternative solutions. The shaded regions in Figure 4(d,e) show the empirical range of observed successes rather than a certified feasible region.

Sensitivity to Instruction Specification. We test how alternative task specifications affect execution by evaluating Original and L1–L4 instruction variants with GPT-5.6 Sol across the same 18 tasks. Compared with the Original instruction ((20.8%) GT-Subgoal progress; (5.6%) task success), L2 and L3 increase progress to (26.9%) and (34.6%), respectively, while both reach (11.1%) task success.Instruction details produce sparse, task-dependent gains. As shown in Figure 4(a), changing the task specification leaves many tasks unchanged, while producing large gains on a small subset and substantial regressions on others. For GPT-5.6 Sol, several variants improve aggregate GT-Subgoal progress and task success over the Original instruction, but the best intermediate progress does not necessarily yield the best final success.These results show that foundation models can exploit additional task information through the TextMani interface, but the benefit depends on which task details are provided, not simply on instruction length or detail. The selected instruction also remains executable across the broader 14-model evaluation, supporting instruction responsiveness across heterogeneous models without implying a uniform instruction-induced gain.

3.3 Can Models Diagnose Failures from Execution Evidence?

After evaluating execution, we next examine whether the exposed intermediate representations and execution traces can support the identification of task-relevant errors. We distinguish two stages of this process: whether an error is detected and whether it is localized to the corresponding subgoal.

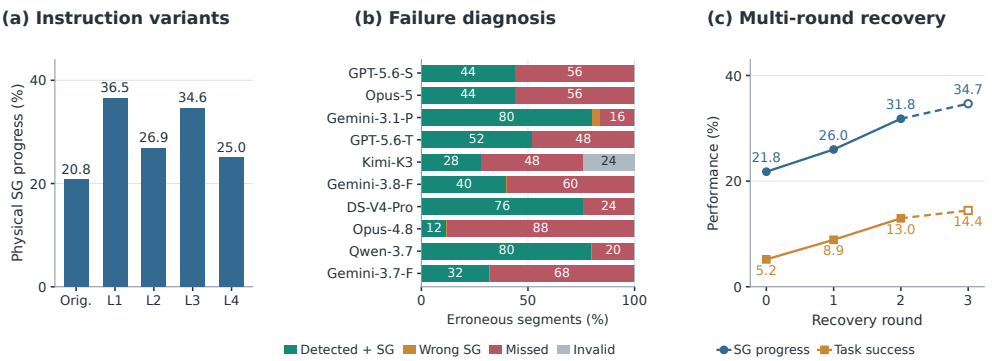


Figure 5: **Instruction sensitivity, blind diagnosis, and multi-round recovery.** (a) Initial physical-subgoal progress under five instruction formulations for GPT-5.6 Sol across 18 simulated tasks. (b) Blind diagnosis outcomes for ten models on 25 erroneous real-robot execution segments, separating correct detection/localization, incorrect localization, missed errors, and invalid responses. (c) Best-so-far SG progress and cumulative task success over successive simulation recovery rounds. Detailed normalization, denominators, and partial-round coverage are provided in Appendix.

Recognition of Normal and Erroneous Execution Segments. This experiment directly evaluates whether models can identify task-relevant errors from execution records. We stratified existing real-robot records by task and label and sampled 50 motion segments from distinct source episodes, comprising 25 normal and 25 erroneous cases. Ten models were given the same task description, geometry, compact constraints, optimization target, and sampled executed trajectory, and were asked to make a binary yes/no judgment. Human annotations, the Space flag, the identity of the source model, and successful reference executions were withheld. For each case, we uniformly sampled 32 trajectory points by record index while preserving their original timestamps; the input therefore does not contain the complete high-frequency trajectory.

Across all models, the overall judgment accuracy is 66.8%, with Gemini 3.1 Pro and Qwen3.7 Max reaching 84.0% and 80.0%, respectively. The ability to distinguish normal from erroneous executions without access to human labels provides evidence that the models possess some task-relevant error discrimination capability and that the execution records exposed by TextMani can support post-execution model inspection. The parseable-response rate is 97.4%, with a normal-case identification rate of 84.4%, an error recall of 49.2%, and a normal-case false-positive rate of 12.8%. These results further show that the discrimination capability is present but asymmetric: models identify normal executions more reliably than they detect errors, and missed errors remain a major limitation of diagnostic reliability.

From Detection to Subgoal Localization. Once an error is detected, subgoal localization is nearly lossless. The error-detection rate is $49.2\,\%$, while the joint rate requiring both correct detection and localization to the frozen subgoal is $48.8\,\%$, a decrease of only 0.4 percentage points. Consistent with Figure 5(b), most diagnostic failures therefore arise before localization, from errors that are not detected at all.Conditioned on successfully detected errors, subgoal matching reaches $99.2\,\%$. This conditional rate does not imply that $99.2\,\%$ of all failures are localized correctly, because undetected failures are excluded. Rather, it shows that once models recognize a task-relevant error, they almost always identify its corresponding frozen subgoal. The dominant bottleneck is therefore *failure recognition*, not subgoal selection after detection. The frozen subgoal serves as a process-level localization reference and does not by itself identify the exact erroneous constraint or causal failure mechanism.

3.4 Can Failed Specifications Be Repaired?

We further examine whether these explicit intermediate representations can be locally modified to change execution outcomes. The real-robot experiments evaluate physical repair given a known failed subgoal, whereas the simulation experiments evaluate the existing recovery pipeline in terms of progress, final outcomes, and failure transitions under different recovery budgets. We report the two settings separately.

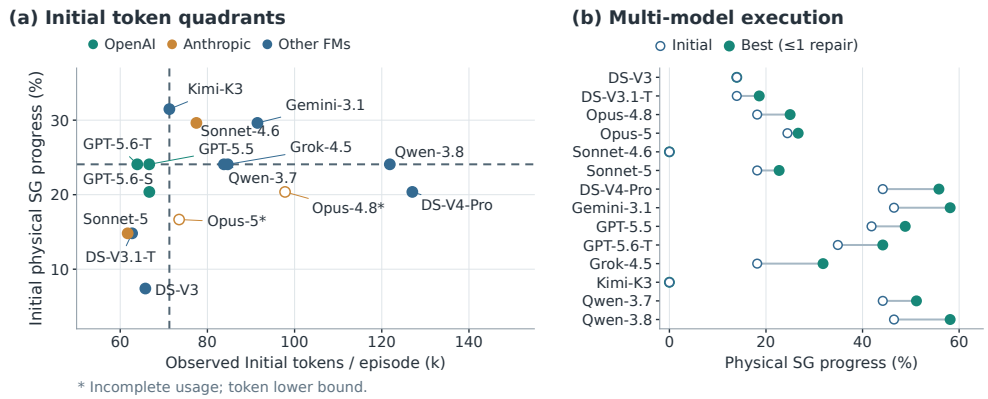


Figure 6: **Token usage and execution performance across foundation models.** (a) Initial token usage versus physical-subgoal progress across 15 models in the 18-task simulation evaluation. (b) Initial and best SG progress under a one-repair budget for the separate 14-model selected-instruction evaluation. Detailed cohort definitions, token-record completeness, and plotting conventions are provided in Appendix.

Real-Robot Repair. We evaluate subgoal-scoped repair on 155 confirmed failed TextMani executions. Given the failed subgoal, *Self* uses the original generating model for revision, whereas *Task-specialist* uses the model selected and frozen for that task based on prior task performance. In both settings, the failed subgoal specification is regenerated while the completed prefix and subgoal ordering are preserved, and the revised specification is compiled, solved, and executed through the same downstream pipeline.Failed specifications can be physically repaired, and repair capability is strongly model dependent. Self recovers $24.5\,\%$ of failed tasks, whereas Task-specialist reaches $47.7\,\%$, a gain of 23.2 percentage points. On the subset with directly comparable pre- and post-repair progress, mean progress increases by 29.3 points for Self and 47.1 points for Task-specialist. These results demonstrate that selecting repairers based on prior task-execution performance improved full-task recovery over Self repair on the same assigned failure cases. This result supports task-specific execution competence as a useful criterion for repairer selection.

Multi-round Recovery in Simulation. On the fixed set of 270 simulation conditions, the best consecutive GT-subgoal progress achieved within the available budget is 21.8%, 26.0%, and 31.8% for Initial, up to RA1, and up to RA2, respectively. The corresponding rates of achieving task success at least once are 5.2%, 8.9%, and 13.0%. Increasing the recovery budget therefore allows more conditions to reach greater progress or achieve task completion at least once.Progress gains do not necessarily translate into additional full-task successes. For example, Kimi K3 improves its best SG progress from 17/54 to 21/54, while its task success remains unchanged at 1/18. Consecutive subgoal progress therefore complements binary task success by revealing recovery gains in model–task combinations that do not reach complete task execution.

4 Conclusion

We introduced **TextMani**, a constraint-based language-to-action framework that provides heterogeneous foundation models with a shared executable specification interface and links the resulting specifications to execution evidence for diagnosis and revision. Across 15 model configurations in simulation and 10 models on six real-robot tasks, TextMani enables systematic evaluation beyond terminal task success through physical-subgoal progress and stage-wise execution signals. On real robots, the structured workflow improves overall task success over direct numerical-target prediction, while successful executions retain diverse trajectories rather than collapsing to a single reference motion. Blind diagnosis further shows that failure recognition, rather than localization after detection, is the main diagnostic bottleneck. Given a known failed subgoal, specification revision can recover failed physical executions, and task-specific execution performance provides a useful signal for selecting effective repairers.Future work should close the remaining gap between these stages: improving reliable failure detection and extending the framework to longer-horizon tasks and broader physical environments.

AI USE STATEMENT

We used generative AI tools to assist with literature search, code debugging, manuscript language polishing, figure/table preparation, and LaTeX formatting. All AI-assisted outputs, including citations, code, analyses, and manuscript text, were manually reviewed and verified by the authors. The authors made all final methodological and scientific decisions and take full responsibility for the content of this paper.

ETHICS STATEMENT

This work studies robotic manipulation in simulation and controlled real-robot environments and does not involve human subjects or personal data. All physical experiments were conducted in a controlled laboratory setting.

REPRODUCIBILITY STATEMENT

We provide detailed metric definitions, model configurations, and additional experimental results in the appendix. The main paper specifies the shared representation, execution, diagnosis, and repair procedures used across models. Upon acceptance, we will publicly release the implementation, task configurations, evaluation code, and supplementary artifacts required to reproduce the reported simulation and real-robot experiments.

REFERENCES

Alibaba Cloud. Qwen3.7-Max Model Documentation. https://www.alibabacloud.com/help/en/model-studio/qwen3-7-max, 2026a. Accessed: 2026-09-26.

Alibaba Cloud. Qwen3.8-Max Model Documentation. https://www.alibabacloud.com/help/en/model-studio/qwen3-8-max, 2026b. Accessed: 2026-09-26.

Anthropic. Introducing Claude Opus 4.8. https://www.anthropic.com/news/claude-opus-4-8, May 2026a. Accessed: 2026-09-26.

Anthropic. Introducing Claude Opus 5. https://www.anthropic.com/news/claude-opus-5, July 2026b. Accessed: 2026-09-26.

Anthropic. Introducing Claude Sonnet 4.6. https://www.anthropic.com/news/claude-sonnet-4-6, February 2026c. Accessed: 2026-09-26.

Anthropic. Introducing Claude Sonnet 5. https://www.anthropic.com/news/claude-sonnet-5, June 2026d. Accessed: 2026-09-26.

Shraddha Barke, Arnav Goyal, Alind Khare, Avaljot Singh, Suman Nath, and Chetan Bansal. Agentrx: Diagnosing ai agent failures from execution trajectories. *arXiv preprint arXiv:2602.02475*, 2026.

Aidan Curtis, Nishanth Kumar, Jing Cao, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Trust the proc3s: Solving long-horizon robotics problems with llms and constraint satisfaction. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 1362–1383. PMLR, 2025.

DeepSeek. DeepSeek-V3 Release. https://www.deepseek.com/news/deepseek-v3/, December 2024. Accessed: 2026-09-26.

DeepSeek. DeepSeek-V3.1 Terminus Update. https://deepseek.com/news/v3-1-terminus/, September 2025. Accessed: 2026-09-26.

DeepSeek. DeepSeek-V4-Pro GA Release. https://api-docs.deepseek.com/news/news260813/, August 2026. Accessed: 2026-09-26.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *International Conference on Learning Representations*, 2025.

Google DeepMind. Gemini 3.1 Pro Model Card. `https://deepmind.google/models/model-cards/gemini-3-1-pro/`, February 2026a. Accessed: 2026-09-26.

Google DeepMind. Gemini 3.7 Flash Model Card. `https://deepmind.google/models/model-cards/gemini-3-7-flash/`, August 2026b. Accessed: 2026-09-26.

Google DeepMind. Gemini 3.8 Flash: Model Card. `https://deepmind.google/models/model-cards/gemini-3-8-flash/`, September 2026c. Accessed: 2026-09-26.

Weizheng Gu, Chengze Li, Zhuohao Yu, Mengyuan Sun, Zhibang Yang, Wei Wang, Hongrui Jia, Shikun Zhang, and Wei Ye. What do agents learn from trajectory-sft: Semantics or interfaces? *arXiv preprint arXiv:2602.01611*, 2026.

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. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 540–562, 2023a.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Tomas Jackson, Noah Brown, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782, 2023b.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Proceedings of the 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 4573–4602, 2025.

Brian Ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T. Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In *Proceedings of the 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318, 2023.

Wonjoong Kim, Sangwu Park, Yeonjun In, Sein Kim, Dongha Lee, and Chanyoung Park. Beyond the final answer: Evaluating the reasoning trajectories of tool-augmented agents. *arXiv preprint arXiv:2510.02837*, 2025.

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 *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500, 2023. doi: 10.1109/ICRA48891.2023.10160591.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, 2024.

Zeyi Liu, Arpit Bahety, and Shuran Song. Reflect: Summarizing robot experiences for failure explanation and correction. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 3468–3484, 2023.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. Agentboard: An analytical evaluation board of multi-turn llm agents. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Moonshot AI. Kimi K3: A new frontier in intelligence. https://www.kimi.com/news/kimi-k3, July 2026. Accessed: 2026-09-26.

OpenAI. GPT-5.5 Model. https://developers.openai.com/api/docs/models/gpt-5.5, 2026a. Accessed: 2026-09-26.

OpenAI. GPT-5.6 Sol Model. https://developers.openai.com/api/docs/models/gpt-5.6-sol, 2026b. Accessed: 2026-09-26.

OpenAI. GPT-5.6 Terra Model. https://developers.openai.com/api/docs/models/gpt-5.6-terra, 2026c. Accessed: 2026-09-26.

Weiliang Tang, Jia-Hui Pan, Yun-Hui Liu, Masayoshi Tomizuka, Li Erran Li, Chi-Wing Fu, and Mingyu Ding. Geomanip: Geometric constraints as general interfaces for robot manipulation. *arXiv preprint arXiv:2501.09783*, 2025.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *International Conference on Learning Representations*, 2024.

xAI. Introducing Grok 4.5. https://x.ai/news/grok-4-5, July 2026. Accessed: 2026-09-26.

J-Anne Yow, Neha Priyadarshini Garg, Manoj Ramanathan, and Wei Tech Ang. ExTraCT: Explainable trajectory corrections for language-based human-robot interaction using textual feature descriptions. *Frontiers in Robotics and AI*, 11:1345693, 2024. doi: 10.3389/frobt.2024.1345693.

5 Appendix

6 Related Work

Foundation models for robotic planning and control. Foundation models have been connected to robot control through a variety of intermediate interfaces. SayCan combines language-model planning with learned skill affordances (Ichter et al., 2023); Code as Policies generates executable programs that invoke perception and control APIs (Liang et al., 2023); Inner Monologue incorporates execution and environment feedback for closed-loop replanning (Huang et al., 2023b); and VoxPoser grounds language-inferred affordances and constraints into composable 3D value maps for trajectory synthesis (Huang et al., 2023a). Together, these works show that intermediate representations can bridge foundation-model reasoning and physical control. TextMani instead focuses on whether this interface can be made simultaneously executable, inspectable, and revisable, so that failures can be localized and corrected at the level of the generated specification.

Structured constraint interfaces for manipulation. Recent work has explored structured constraints as interfaces between foundation-model reasoning and robot control. ReKep represents manipulation goals with VLM-generated relational keypoint constraints and optimizes robot actions to satisfy them (Huang et al., 2025); GeoManip constructs stage-specific geometric specifications and executable objectives for trajectory optimization (Tang et al., 2025); and PRoC3S combines LLM-generated parameterized programs with environmental constraints in a continuous constraint-satisfaction formulation (Curtis et al., 2025). These approaches demonstrate different ways of translating high-level model outputs into physically grounded action specifications, but are typically instantiated around particular models, representations, solvers, and execution interfaces. TextMani provides a common constraint-based interface that enables heterogeneous foundation models to be compared under a shared specification and execution protocol. Building on this interface, we establish a unified failure-handling protocol that aligns execution evidence with failure recognition and localization, while supporting constraint-level attribution, traceable revision, and localized repair.

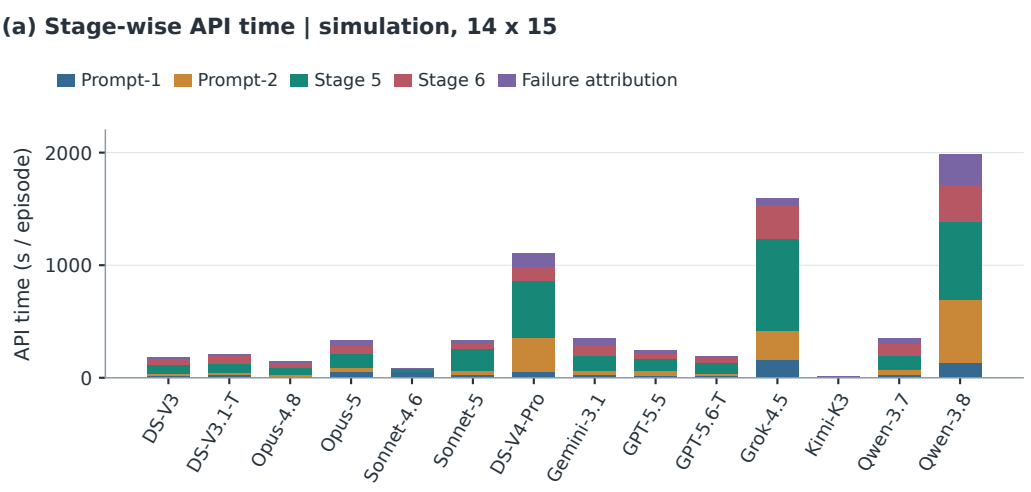


Figure 7: Resource use in two separately identified simulation experiments. (a) Mean recorded API latency by stage over Initial plus RA1 in the 14-model, 15-task selected-instruction cohort, divided by 15 planned conditions per model. (b) Observed Initial tokens versus mean Initial physical-SG progress in the original 15-model, 18-task cohort. Dashed splits are medians of the 13 complete-usage models. Opus-4-8 and Opus-5 have incomplete usage (99.4% and 93.3% of calls), so hollow-marker token coordinates are lower bounds. These panels do not form a paired latency-token comparison; token counts are not monetary costs.

Process-level evaluation and failure repair. Recent work has shown that interactive-agent performance cannot be fully characterized by final success alone. AgentBench analyzes recurrent failure modes across interactive environments (Liu et al., 2024); AgentBoard introduces fine-grained progress-based evaluation (Ma et al., 2024); and MINT demonstrates that strong single-turn performance does not necessarily translate to effective feedback-driven interaction (Wang et al., 2024). In robotics, REFLECT uses structured experience summaries for failure explanation and correction, AHA explicitly detects and reasons about manipulation failures, and AgentRx derives constraints from execution trajectories to localize critical failures (Liu et al., 2023; Duan et al., 2025; Barke et al., 2026). While prior work reasons over execution traces or derives diagnostic constraints after failures occur, TextMani uses constraints as the action-generating interface itself: the same structured specification is compiled into robot targets before execution and later reused for failure attribution and localized repair.

Evaluated foundation models. Across all simulation and real-robot experiments, we evaluate 17 unique foundation-model configurations from seven providers: GPT-5.6 Sol, GPT-5.6 Terra, and GPT-5.5 from OpenAI (OpenAI, 2026b;c;a); Claude Opus 5, Claude Opus 4.8, Claude Sonnet 5, and Claude Sonnet 4.6 from Anthropic (Anthropic, 2026b;a;d;c); Gemini 3.1 Pro, Gemini 3.7 Flash, and Gemini 3.8 Flash from Google DeepMind (Google DeepMind, 2026a;b;c); Kimi K3 from Moonshot AI (Moonshot AI, 2026); Qwen3.7 Max and Qwen3.8 Max from Alibaba Cloud (Alibaba Cloud, 2026a;b); DeepSeek V4 Pro, DeepSeek V3.1-Terminus, and DeepSeek V3 from DeepSeek (DeepSeek, 2026; 2025; 2024); and Grok 4.5 from xAI (xAI, 2026). The main simulation evaluation uses the corresponding 15-model subset, while the real-robot evaluation uses a common 10-model panel; exact API endpoints, snapshots, and inference configurations are reported with the experimental configuration files.

Detailed criteria for Table 1. Table 1 compares structured constraint interfaces for language-guided robot manipulation. Gen. denotes foundation-model generation of a structured task specification; Sep. indicates that semantic specifications and executable mathematical formulations are represented as distinct stages; SG-Prog. denotes quantitative evaluation of execution progress over explicit physical subgoals; Diag. denotes blind failure recognition and localization from execution evidence; Attr. indicates that execution evidence is associated with specific components of the generated specification, enabling constraint-level failure attribution and traceable revision; Repair

Table 4: Initial real-robot execution with TextMani and the Direct workflow. [Keys: **Best**/**Second best** within the stated comparison group.]

Model	TextMani							Direct						
	Open Drw.	Close Drw.	Pick Bottle	Pick Bread	Insert Pen	Pour Cereal	TSR ↑	Open Drw.	Close Drw.	Pick Bottle	Pick Bread	Insert Pen	Pour Cereal	TSR ↑
GPT-5.6 Sol	**60.0%**	40.0%	60.0%	**80.0%**	0.0%	0.0%	40.0%	33.3%	33.3%	33.3%	**33.3%**	**0.0%**	0.0%	22.2%
Claude Opus 5	**40.0%**	**60.0%**	**80.0%**	**80.0%**	**100.0%**	0.0%	**60.0%**	33.3%	0.0%	33.3%	0.0%	**0.0%**	**100.0%**	27.8%
Gemini 3.1 Pro	20.0%	**80.0%**	**80.0%**	0.0%	40.0%	40.0%	**43.3%**	0.0%	0.0%	**66.7%**	**66.7%**	**33.3%**	33.3%	33.3%
GPT-5.6 Terra	0.0%	20.0%	**100.0%**	40.0%	0.0%	0.0%	26.7%	0.0%	33.3%	33.3%	0.0%	**0.0%**	0.0%	11.1%
Kimi K3	20.0%	0.0%	**80.0%**	**80.0%**	40.0%	0.0%	36.7%	**66.7%**	0.0%	0.0%	0.0%	**33.3%**	0.0%	16.7%
Gemini 3.8 Flash	20.0%	20.0%	**80.0%**	**60.0%**	0.0%	**80.0%**	**43.3%**	**100.0%**	**100.0%**	**100.0%**	**33.3%**	**0.0%**	**66.7%**	**66.7%**
DeepSeek V4 Pro	20.0%	0.0%	40.0%	20.0%	40.0%	0.0%	20.0%	33.3%	0.0%	**66.7%**	0.0%	**33.3%**	0.0%	22.2%
Claude Opus 4-8	0.0%	0.0%	0.0%	0.0%	**80.0%**	0.0%	13.3%	0.0%	0.0%	0.0%	**66.7%**	**0.0%**	0.0%	11.1%
Qwen3.7 Max	20.0%	0.0%	60.0%	40.0%	0.0%	0.0%	20.0%	0.0%	33.3%	**66.7%**	0.0%	**0.0%**	0.0%	16.7%
Gemini 3.7 Flash	**40.0%**	**80.0%**	**100.0%**	20.0%	20.0%	**100.0%**	**60.0%**	**100.0%**	**66.7%**	33.3%	**66.7%**	**0.0%**	0.0%	**44.4%**
Mean	**24.0%**	**30.0%**	**68.0%**	**42.0%**	**32.0%**	**22.0%**	**36.3%**	**36.7%**	**26.7%**	**43.3%**	**26.7%**	**10.0%**	**20.0%**	**27.2%**

The common ten-model panel contains six tasks, with five TextMani and three Direct conditions per model-task pair (300/180 in total). Astra is excluded; the original Direct panel had 18 additional Astra conditions, and 30 supplemental TextMani Astra conditions are excluded.
Generation and preparation failures remain in the formal denominators. Missing physical labels (33 TextMani and 54 Direct conditions) remain under the confirmed full-budget rule. Space stops take precedence, with confirmed task-terminal and cleanup exceptions retained.
Equal sample counts within each workflow make overall success equal to the equally weighted model-task average. The September 24 supplemental runs, including Gemini, replace the final execution for the same formal condition according to the manifest.
Best and second-best values are marked within each metric column, excluding totals; ties use displayed precision and constant columns are unmarked.

denotes feedback-driven revision of the generated specification; and X-Model denotes evaluation of heterogeneous foundation models for execution, diagnosis, and repair under a shared specification and execution protocol. GeoManip supports failure-conditioned adjustment of geometric specifications but does not evaluate blind constraint-level attribution and localized repair as separate capabilities. PRoC3S uses constraint-violation feedback from unsatisfiable CCSPs to generate a new program/constraint problem rather than localizing and revising an execution-grounded specification component. TextMani evaluates localized repair separately from blind failure recognition and localization, with execution evidence aligned to explicit components of the generated specification.

Detailed description of Figure 3. Figure 3 presents relative capability profiles for ten real-robot models across eight dimensions. Each score is centered by the equal-weight mean of the ten models, and the displayed radius equals 60 plus the corresponding percentage-point deviation from that mean; the dashed circle therefore marks the model mean rather than an absolute performance threshold. Only the largest positive and negative deviations are labeled, while numeric spoke labels refer to the shared legend. The eight dimensions are: **(1) Object Transfer**, initial task success on bottle and bread pick-and-place; **(2) Pose-Sensitive Manipulation**, initial task success on pen insertion and cereal pouring; **(3) Constrained Manipulation**, initial task success on drawer opening and closing; **(4) Feasible Target Generation**, the proportion of plans passing IK checks for all required motion steps; **(5) State Discrimination**, overall accuracy in distinguishing normal from erroneous execution; **(6) Normality Recognition**, correct no-error judgments on normal execution segments; **(7) Failure Localization**, joint error detection and agreement with the reference physical subgoal; and **(8) Task Recovery**, full-task success after the original model performs Self repair at a known failed subgoal. Dimensions 1–4 pool TextMani and Direct trials at their recorded 5:3 ratio, while Dimensions 5–7 retain invalid or missing responses in their denominators. Because these dimensions are

computed from distinct evaluation sets and denominators, the radar plot is intended to compare relative capability profiles rather than define a difficulty-calibrated universal ability scale or an overall model ranking.

Detailed description of Table 2. Simulation results: initial execution, recovery budgets, and stage performance. [Keys: **Best**/**Second best** within the stated comparison group.]

Rates are percentages; progress changes are in percentage points (pp). SGP denotes GT subgoal progress; TSR denotes task success rate. FRR denotes final recovery rate, AR denotes repair-transition advance rate, and ΔP denotes the change in GT subgoal progress in percentage points. RA1-AR and RA2-AR denote advance rates for the first and second repair attempts, respectively. IK denotes the valid inverse-kinematics rate; P—IK denotes the valid-path rate conditional on valid IK; E—P denotes the successful-execution rate conditional on a valid path; and G—C denotes the successful-grasp rate conditional on an actual close attempt.

GT subgoal progress is a micro-average of contiguous prefix completion; budgeted task success counts at least one success within the budget. Final recovery is evaluated only on initially failed episodes that entered repair.

Advance rates use repair transitions; a transition without a scorable execution retains its preceding progress. Stage rates use planning records, valid IK, valid paths, and actual close attempts, respectively, including cumulative records.

Best and second-best values are marked within each metric column, excluding totals; ties use displayed precision and constant columns are unmarked.

Detailed caption for Figure 4. Task-level instruction effects, failure composition, and diverse successful trajectories. (a) Changes in initial physical-subgoal progress relative to Original, in percentage points, for instruction variants L1–L4 across 18 simulated tasks using GPT-5.6 Sol. Gray zeros indicate no change. (b) Initial failure composition across 15 models in the 18-task simulation evaluation. Percentages use each model's complete failure set; omitted categories remain in the denominator. Control denotes control and interaction failures. (c) Illustrative real-robot examples of alternative successful manipulation realizations. (d,e) TCP height, tool-axis tilt, and Joint 1 angle for 21 successful Pick Bread executions and 34 successful Pick Bottle executions. Three representative successful executions are highlighted consistently across variables; the remaining trajectories appear as faint lines. Shading denotes the empirical range of successful trajectories. Approach, gripper closing, transport, and gripper opening are normalized separately to align execution phases. IQR C/M reports the phase-averaged pointwise interquartile range during closing and transport, in the plotted units. Successful executions exhibit concentrated TCP heights near gripper closure while retaining variation in tool orientation and joint trajectories.

Detailed caption for Figure 5. Instruction sensitivity, failure diagnosis, and multi-round recovery. (a) Initial physical-subgoal (SG) progress under five instruction formulations for GPT-5.6 Sol across 18 simulated tasks. Progress is normalized by the reference SG counts: 53 for Original and 52 for L1–L4. (b) Blind diagnosis by ten models on the same 25 erroneous real-robot execution segments. Stacked bars distinguish joint error detection and correct SG localization, detected errors with incorrect localization, missed errors, and invalid responses. (c) Best-so-far SG progress and cumulative task success over successive recovery rounds in simulation, evaluated over 808 reference SGs and 270 conditions. Round 0 denotes initial execution. The dashed final segment and hollow markers indicate partial third-round coverage: 90 conditions were observed, while the remaining conditions retain their previous best outcomes. Panels report separate experimental settings.

Detailed caption for Figure 6. Token usage and execution performance across foundation models. (a) Observed token usage per episode versus initial physical-subgoal (SG) progress in the 15-model, 18-task simulation evaluation. Colors indicate model families; dashed lines mark the medians across models with complete usage records. Hollow markers and asterisks indicate incomplete token records, making the corresponding token counts lower bounds. (b) Initial and best SG progress with at most one repair in the separate 14-model, 15-task selected-instruction evaluation. Connected markers show the improvement attained within this repair budget. The two panels use separate task cohorts.

Detailed description of Table 3. Real-robot results: execution, inspection, and known-subgoal repair. [Keys: **Best**/**Second best** within the stated comparison group.]

Rates are percentages and progress changes are measured in percentage points (pp). G&P denotes pick bottle and bread; A&O denotes insert pen and pour cereal; C-P/P denotes open and close drawer. JSG denotes the Joint SG metric and requires both error detection and agreement with the frozen subgoal reference. ΔP denotes the change in subgoal progress, measured in percentage points.

Initial execution uses 300 TextMani and 180 Direct conditions on the common ten-model panel. Conditions without physical labels (33/54) remain in the respective denominators and contribute no confirmed successes; no physical failure labels are imputed.

Inspection uses the same 50 cases per model: 25 normal and 25 erroneous cases. JSG requires error detection and agreement with the frozen subgoal reference; this reference is not an independent causal first-error label. There are 13/500 invalid yes/no judgments.

Repair assigns each of 155 original failures to Self and Specialist. Rows identify the failed-plan source model; Specialist results do not describe the named model as repairer. Inspection and repair are separate evaluations.

Progress change uses 255/310 comparable repair conditions; 55 are excluded from that mean. Confirmed Space-stop, terminal-task, cleanup, continuation, and supplemental-run selection rules are retained.

Best and second-best values are marked within each metric column, excluding totals; ties use displayed precision and constant columns are unmarked. Repair columns are not ranked across failure-source models.