REPOMIRAGE: DO CODE AGENTS REALLY UNDERSTAND REPOSITORY STRUCTURES?

Hanyu Li[1*] **Yichi Zhang**[2*] **Speed Zhu**[3] **Yinpeng Dong**[2]

[1] Beijing University of Posts and Telecommunications [2] Tsinghua University [3] Tencent

ABSTRACT

Despite the impressive progress of code agents on code generation benchmarks, their robustness in understanding repository structure is overlooked. This is critical as real repositories often come with noisier and less informative structure cues, such as messy layouts and misleading naming conventions. We present REPOMIRAGE, an automated perturbation framework that probes this robustness gap by producing function-preserving variants of SWE-bench Verified tasks and evaluating agents under repository-level test-time shifts. Multiple levels of perturbations, from prompts to structures, are imposed on the original tasks while maintaining functionality. On 158 curated instances, these shifts consistently degrade performance and make target localization harder, with the resolution rate of GPT-4o dropping from 32.91% to 7.59%, exposing a robustness gap in current code agents.

1 INTRODUCTION

Large Language Model (LLM) agents have rapidly evolved into practical code agents that can navigate codebases, localize bugs, and produce executable patches (Dong et al., 2025; Luo et al., 2025). Recent benchmarks (Yang et al., 2025b; Fu et al., 2025) increasingly target repository-level software engineering, where agents must comprehend project structures and handle complex issues. This ability is desired because realistic problems often require identifying the correct modification scope across multiple files and dependencies, rather than producing a locally correct change in isolation.

Despite continuous progress in code agents for coding (Yang et al., 2024), strong performance on benchmarks does not necessarily imply their potential for real-world deployments concerning repository understanding. Most benchmarks are built from open-source projects with a relatively clean structure, where agents may succeed via shortcuts such as keyword matching or inferring edit locations from semantically aligned file and symbol names (Le Hai et al., 2025; Gu et al., 2025). However, when it comes to assisting common users, cluttered and ambiguous structures are prevalent, making such shortcuts unavailable. This raises a central question: *Do code agents really understand repository structures?* If success is driven by shallow heuristics, evaluations may overestimate the reliability of these agents in realistic settings, where test-time distribution shifts in structure can be noisy, incomplete, or misleading, posing a practical robustness challenge.

In this paper, we aim to test whether agents can maintain performance when the repository structure becomes noisier or less informative, which requires stronger repository understanding rather than reliance on shortcut cues. To address this, we introduce REPOMIRAGE, an automated and reproducible perturbation framework that stress-tests repository understanding, creating function-preserving variants of SWE-Bench Verified tasks (Chowdhury et al., 2024) to evaluate agents under such repository-level distribution shifts. However, naive repository perturbations are insufficient, as even minor edits can silently break packaging, imports, or runtime behaviors, rendering originally solvable tasks unsolvable. REPOMIRAGE applies controllable perturbations at multiple levels to minimize the impacts of perturbations to code completeness, which is further enforced through consistency filtering for each sample. This results in a curated subset of transformed tasks that remain correct by construction, enabling reliable evaluation across agents.

On the curated 158-instance REPOMIRAGE dataset, perturbations at multiple repository levels leading to test-time structure shifts consistently degrade code agent performance and make debugging

*Equal Contribution. (mail to: thestudent@bupt.edu.cn, zyc22@tsinghua.edu.cn)

localization notably harder, underscoring current agents' limited ability to understand, navigate, and modify repositories. Resolution rates decline across representative models, from modest degradations for stronger models (GPT-5: 62.66% to 55.70%) to severe collapses for weaker ones (GPT-4o: 32.91% to 7.59%). Ablation study further shows that REPOMIRAGE sharply lowers the chance of locating the correct target files and increases the steps for agents to reach them.

Our contributions are three-fold:

- Conceptually, we identify a repository-level robustness failure mode for code agents, where their performance can be brittle under test-time structural shifts, exposing limitations in repository understanding beyond patch synthesis.
- Technically, we develop REPOMIRAGE, an automated and reproducible perturbation framework that incorporates multi-level perturbations on repositories in SWE-Bench Verified and preserves the functionality, enabling controlled stress tests for code agents.
- Empirically, we evaluate code agents powered by mainstream LLMs on REPOMIRAGE, complemented by trajectory-level analyses, which reveal their common failure patterns when navigating and modifying perturbed repositories.

2 RELATED WORK

Code Benchmarks. Code benchmarks have evolved from early evaluations focused on standalone programming problems, such as HumanEval (Chen et al., 2021) and LiveCodeBench (Jain et al., 2024), toward more realistic settings that require agents to operate over entire software projects. Recent benchmarks, including SWE-bench (Jimenez et al., 2024) and SWE-bench-Pro (Deng et al., 2025), evaluate repository-level bug fixing under test-driven constraints, while others further emphasize multi-file reasoning (Rashid et al., 2025; Liu et al., 2023; Li et al., 2024; Fu et al., 2025; Yang et al., 2025b). Despite this progression, these benchmarks are constructed from open-source projects with well-organized structure, which may enable shortcut-based reasoning and overestimate agents' robustness to structural ambiguity.

Code Agent Robustness. There have been studies on the robustness of code agents under controlled perturbations, most overlooking the complex nature in repository understanding. Prior work examines sensitivity to natural language variations by rewriting problem descriptions while preserving intent, showing that even minor linguistic changes can affect model behavior (Mastropaolo et al., 2023; Chen et al., 2024). Other work applies semantics-preserving code transformations, such as variable renaming, control-flow rewrites, and dead-code insertion, to test whether models rely on surface code patterns (Li et al., 2023; Fan et al., 2023; Orvalho & Kwiatkowska, 2025; Lam et al., 2025). A limited number of recent studies have begun to consider robustness in broader codebase contexts (Lee et al., 2025). In this line, REPOMIRAGE investigates repository-level robustness by systematically weakening structural cues while preserving end-to-end functionality.

3 FRAMEWORK OF REPOMIRAGE

REPOMIRAGE is an automated and reproducible framework over repositories, as illustrated in Figure 1. Unlike prior work that applies perturbation on prompt or code to single files, REPOMIRAGE operates from the perspective within a repository context. We extend these script-level transformations to multiple levels of perturbations to examine the robustness of code agents.

Prompt-Level Perturbation. At the prompt level, we rewrite issue descriptions to reduce structural guidance. The rewritten issues adopt a more informal tone, introduce linguistic noise, and obscure references to repository structure, while preserving the task intent. This rewriting process is performed automatically using `GPT-4.1` with designed prompts to ensure semantic consistency.

Code-Level Perturbation. At the code level, we apply obfuscations to files modified by the gold patch, using AST transformations (Bielik & Vechev, 2020) to ensure semantic preservation. Specifically, we employ three types of obfuscation: (1) renaming local variables within functions; (2) performing conversions between `if-else` statements and ternary expressions; (3) injecting non-functional dead code. By constraining these transformations to local scopes and avoiding global symbols, the framework preserves executability while reducing surface code patterns.

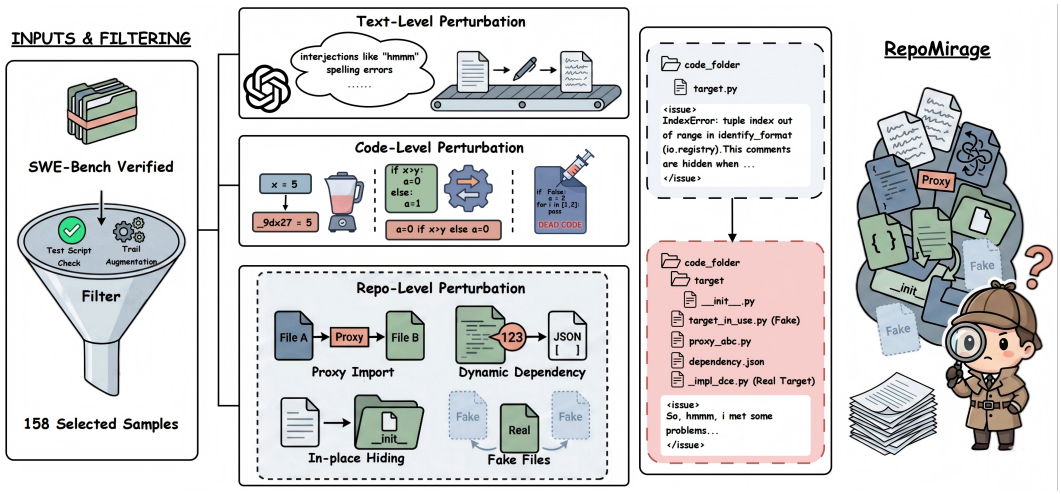

Figure 1: Framework workflow for REPOMIRAGE. We collect instances from SWE-bench Verified, filter candidates with testing scripts, then apply three level of perturbations. The pipeline outputs REPOMIRAGE dataset with 158 samples.

Repository-Level Perturbation. At the repository level, REPOMIRAGE introduces structure-preserving but signal-disrupting transformations that explicitly target cross-file reasoning. In practice, we implement four types of transformations. (1) *Proxy import*: direct import statements in a target file are rewritten to import from a newly created proxy module at the same directory level, which re-exports the original dependency. (2) *Dynamic dependency*: numeric constants in the target file are externalized into a configuration file and loaded at runtime, converting in-file constants into cross-file dependencies. (3) *In-place hiding*: a source file is transformed into a package by replacing it with a directory of the same name and an `__init__.py` that re-imports the original logic from a renamed file. (4) *Fake files*: additional files with similar names but incomplete content are introduced alongside the target file to create misleading structural cues.

Data Filtering. We start from the SWE-bench Verified subset (Chowdhury et al., 2024) and apply a filtering process. Only instances that admit safe perturbations are retained. For each candidate, perturbations are applied and validated via `PASS_TO_PASS` testing, which is the standard SWE-bench protocol in which all tests must pass both before and after the patch. After filtering, we obtain a subset of 158 instances for controlled repository-level evaluation.

Overall, REPOMIRAGE incorporates multi-level perturbations from the repository perspective under an automated pipeline, enabling stress testing of repository understanding in code agents.

4 EXPERIMENTS

Setup and Metrics. We evaluate REPOMIRAGE on the curated subset of SWE-bench Verified using the standard bash-only evaluation protocol. All LLMs operate within the same `mini-swe-agent` framework(Yang et al., 2024), interacting with the repository through command-line tools to generate and apply patches. An instance is considered resolved if all tests pass after the agent-generated patch. In addition to resolution rate, we report interaction statistics that reflect repository-level behavior, including the number of steps taken, the step at which the agent first locates the target file (Avg. LocStep), and the average number of opened files (Avg. OpenFiles).

Model Comparison. Table 1 reports both absolute resolution rates and relative drop rates under REPOMIRAGE. All models experience performance degradation, but the magnitude of robustness loss varies substantially. Stronger models such as `GPT-5` and `DeepSeek-V3.2`(Liu et al., 2025) exhibit relatively mild drops (11.11% and 8.14%, respectively), whereas weaker models suffer severe relative degradation, including `GPT-4o` (76.94%), `Gemini-2.5-Pro`(Comanici et al., 2025) (60.17%), `Qwen3-Coder-30B` (48.27%) and `Qwen3-30B`(Yang et al., 2025a) (74.12%). These results suggest that while stronger models are more resilient to repository-level perturbations, none are fully robust to the structural ambiguity introduced by REPOMIRAGE.

Table 1: Comparison of LLM agents on REPOMIRAGE, reporting resolution rate and procedure statistics. Origin denotes the resolution rate prior to perturbations. Located denotes the rate that agent locates the target file, Avg. LocStep denotes the number of steps to locate the target, and Avg. OpenFiles denotes the number of opened files.

Model Name	Origin	Resolved	Located	Avg. Step	Avg. LocStep	Avg. OpenFiles
GPT-4o	32.91	7.59$^{-76.94\%}$	25.32	38.50	11.25	18.25
GPT-5	62.66	55.70$^{-11.11\%}$	77.22	16.69	7.66	11.54
Gemini-2.5-Pro	53.80	21.43$^{-60.17\%}$	48.81	19.25	12.37	11.98
DeepSeek-V3.2-Reasoner	59.49	54.65$^{-8.14\%}$	68.99	47.66	19.26	17.44
Qwen3-Coder-30B	22.02	11.39$^{-48.27\%}$	39.24	43.37	14.52	15.56
Qwen3-30B	14.14	3.66$^{-74.12\%}$	16.46	86.99	10.77	10.46

Table 2: Ablation study of GPT-5 under different perturbation levels.

Perturbation Level	Resolved	Located	Avg. Step	Avg. LocStep	Avg. OpenFiles
Non-Perturbation	62.66	97.21	12.64	3.07	9.06
Text-Level	62.23	96.25	14.86	3.41	9.27
Code-Level	59.49	97.56	13.26	3.00	9.24
Repo-Level	58.23	78.77	16.46	8.12	10.92
RepoMirage	55.70	77.22	16.69	7.66	11.54

Ablation Study over Perturbation Levels. Table 2 reports an ablation study of GPT-5 under different perturbations. Text-level and code-level perturbations lead to modest performance changes. Repository-level perturbations alone do not lead to a large drop in resolution rate, but they substantially reduce localization accuracy and increase localization steps. This suggests that perturbations affect repository understanding. In the full REPOMIRAGE setting, effects accumulate, pointing to repository-level reasoning failures as a key factor behind unresolved cases.

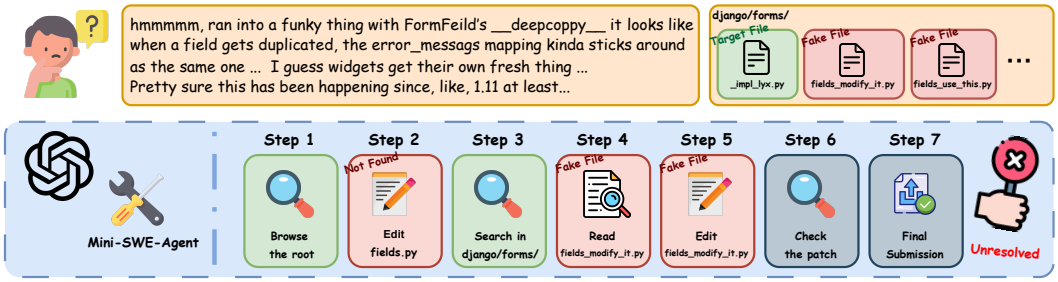

Figure 2: Case study over an unresolved sample of `GPT-5`.

Case Study. To qualitatively analyze agent behavior, we present a case study on `django__django-11880` transformed by REPOMIRAGE (Figure 2). Repository-level perturbations introduce proxy imports and fake files that obscure structural cues, diverting the agent toward misleading targets. As a result, the agent repeatedly edits incorrect files and fails due to wrong file selection. This case illustrates a common failure pattern: agents rely on surface patterns and struggle to recover once misled by indirection.

5 CONCLUSION

REPOMIRAGE studies the robustness of existing code agents in understanding repository structures, which significantly limits their real-world utility. With REPOMIRAGE, we show that this is a critical issue. Across diverse LLM agents, repository-level perturbations consistently increase localization difficulty, even when functionality is preserved. Strong models often produce valid patches yet fail due to incorrect file selection, indicating reliance on surface-level structural cues. Overall, REPOMIRAGE provides a controlled stress test for repository-level robustness, and we hope it encourages future work on explicit repository modeling for long-horizon code agents.

REFERENCES

Pavol Bielik and Martin Vechev. Adversarial robustness for code. In *International Conference on Machine Learning*, pp. 896–907. PMLR, 2020.

Junkai Chen, Li Zhenhao, Hu Xing, and Xia Xin. Nlperturbator: Studying the robustness of code llms to natural language variations. *ACM Transactions on Software Engineering and Methodology*, 2024.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Neil Chowdhury, James Aung, Chan Jun Shern, Oliver Jaffe, Dane Sherburn, Giulio Starace, Evan Mays, Rachel Dias, Marwan Aljubeh, Mia Glaese, Carlos E. Jimenez, John Yang, Leyton Ho, Tejal Patwardhan, Kevin Liu, and Aleksander Madry. Introducing SWE-bench verified, 2024. URL https://openai.com/index/introducing-swe-bench-verified/.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Xiang Deng, Jeff Da, Edwin Pan, Yan He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, Karmini Sampath, Maya Krishnan, Srivatsa Kundurthy, Sean M. Hendryx, Zifan Wang, Chen Bo Calvin Zhang, Noah Jacobson, Bing Liu, and Brad Kenstler. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? 2025. URL https://api.semanticscholar.org/CorpusID:281421060.

Yihong Dong, Xue Jiang, Jiaru Qian, Tian Wang, Kechi Zhang, Zhi Jin, and Ge Li. A survey on code generation with llm-based agents. *arXiv preprint arXiv:2508.00083*, 2025.

Zhiyu Fan, Xiang Gao, Martin Mirchev, Abhik Roychoudhury, and Shin Hwei Tan. Automated repair of programs from large language models. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1469–1481. IEEE, 2023.

Kelin Fu, Tianyu Liu, Zeyu Shang, Yingwei Ma, Jian Yang, Jiaheng Liu, and Kaigui Bian. Multi-docker-eval: Ashovel of the gold rush'benchmark on automatic environment building for software engineering. *arXiv preprint arXiv:2512.06915*, 2025.

Wenchao Gu, Juntao Chen, Yanlin Wang, Tianyue Jiang, Xingzhe Li, Mingwei Liu, Xilin Liu, Yuchi Ma, and Zibin Zheng. What to retrieve for effective retrieval-augmented code generation? an empirical study and beyond. *arXiv preprint arXiv:2503.20589*, 2025.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VTF8yNQM66.

Man Ho Lam, Chaozheng Wang, Jen-tse Huang, and Michael R. Lyu. Codecrash: Stress testing llm reasoning under structural and semantic perturbations. *arXiv preprint arXiv:2504.14119*, 2025.

Nam Le Hai, Dung Manh Nguyen, and Nghi DQ Bui. On the impacts of contexts on repository-level code generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1496–1524, 2025.

Hyunji Lee, Minseon Kim, Chinmay Singh, Matheus Pereira, Atharv Sonwane, Isadora White, Elias Stengel-Eskin, Mohit Bansal, Zhengyan Shi, Alessandro Sordoni, et al. Gistify! codebase-level understanding via runtime execution. *arXiv preprint arXiv:2510.26790*, 2025.

Jia Li, Ge Li, Xuanming Zhang, Yunfei Zhao, Yihong Dong, Zhi Jin, Binhua Li, Fei Huang, and Yongbin Li. Evocodebench: An evolving code generation benchmark with domain-specific evaluations. volume 37, pp. 57619–57641, 2024.

Zongjie Li, Chaozheng Wang, Zhibo Liu, Haoxuan Wang, Dong Chen, Shuai Wang, and Cuiyun Gao. Cctest: Testing and repairing code completion systems. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 1238–1250. IEEE, 2023.

Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

Tianyang Liu, Canwen Xu, and Julian McAuley. Repobench: Benchmarking repository-level code auto-completion systems. *arXiv preprint arXiv:2306.03091*, 2023.

Jane Luo, Xin Zhang, Steven Liu, Jie Wu, Jianfeng Liu, Yiming Huang, Yangyu Huang, Chengyu Yin, Ying Xin, Yuefeng Zhan, et al. Rpg: A repository planning graph for unified and scalable codebase generation. *arXiv preprint arXiv:2509.16198*, 2025.

Antonio Mastropaolo, Luca Pascarella, Emanuela Guglielmi, Matteo Ciniselli, Simone Scalabrino, Rocco Oliveto, and Gabriele Bavota. On the robustness of code generation techniques: An empirical study on github copilot. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 2149–2160. IEEE, 2023.

Pedro Orvalho and Marta Kwiatkowska. Are large language models robust in understanding code against semantics-preserving mutations? *arXiv preprint arXiv:2505.10443*, 2025.

Muhammad Shihab Rashid, Christian Bock, Yuan Zhuang, Alexander Buchholz, Tim Esler, Simon Valentin, Luca Franceschi, Martin Wistuba, Prabhu Teja Sivaprasad, Woo Jung Kim, et al. Swe-polybench: A multi-language benchmark for repository level evaluation of coding agents. *arXiv preprint arXiv:2504.08703*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

John Yang, Carlos E Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik R Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2405.15793.

John Yang, Kilian Lieret, Joyce Yang, Carlos E Jimenez, Ofir Press, Ludwig Schmidt, and Diyi Yang. Codeclash: Benchmarking goal-oriented software engineering. *arXiv preprint arXiv:2511.00839*, 2025b.