2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Grounding Language Models with Semantic Digital Twins for Robotic Planning

Mehreen Naeem[1], Andrew Melnik[1], Michael Beetz[1]

***Abstract*—We introduce a novel framework that integrates Semantic Digital Twins (SDTs) with Large Language Models (LLMs) to enable adaptive and goal-driven robotic task execution in dynamic environments. The system decomposes natural language instructions into structured action triplets, which are grounded in contextual environmental data provided by the SDT. This semantic grounding allows the robot to interpret object affordances and interaction rules, enabling action planning and real-time adaptability. In case of execution failures, the LLM utilizes error feedback and SDT insights to generate recovery strategies and iteratively revise the action plan. We evaluate our approach using tasks from the ALFRED benchmark, demonstrating robust performance across various household scenarios. The proposed framework effectively combines high-level reasoning with semantic environment understanding, achieving reliable task completion in the face of uncertainty and failure.**

I. Introduction

Exploring dynamic environments has revolutionized how robots interpret and execute complex tasks by leveraging advanced natural language processing capabilities [1]. This allows robots to decompose high-level instructions into actionable steps and adapt dynamically [2], as seen in frameworks like RePLan [3][4], which enables online re-planning in response to unexpected obstacles. However, challenges such as safety concerns and model hallucinations (where models generate plausible but incorrect outputs) persist. To address these, some work incorporates safety checks into the planning process[5][6], while others use scene graphs to ground LLMs in the physical environment[7], enhancing plan feasibility by connecting language models to real-world contexts through sensor data or computer vision.

Traditional Digital Twins [8] primarily focus on geometric and physical representations of the environment, such as objects, meshes, and features, whereas Semantic Digital Twins (SDTs) incorporate real-time data [9]. It mirrors a building's systems and components of DT by adding layer of rich contextual and semantic information about dynamic surroundings by integrating the objects' text descriptions, rules, and interaction properties[10][11].Various approaches have been used to integrate SDTs with LLM agent in affordance-based scene representations to enable large-scale task planning in complex environments[12][13].

The fusion of LLM and SDTs offers a promising path to creating robotic systems capable of operating in dynamic environments. The relevance of this collaboration is to support a robot's ability by effectively adjusting robot actions that rely on real-time environmental feedback[14][4]. However, the main challenge is to align the LLM response with the semantic structures represented in SDTs, ensuring seamless communication between the two systems. Our framework addresses these challenges essential for bridging high-level reasoning with real-world execution in dynamic environments. This paper discusses the necessary preliminaries, including related work on integrating LLMs and SDTs, workflow construction, experimental setup, and evaluation results. Finally, it concludes with potential directions for future research.

[1]Institute for Artificial Intelligence, University of Bremen, Bremen, Germany. Corresponding author mnaeem@uni-bremen.de

II. Related Work

A. Leveraging Semantic Digital Twins for Task Planning

Recently, there have been advancements in how robots plan tasks, especially when they don't have all the information [15, 16, 17, 18]. AutoGPT+P [19] is an example of robots using object affordances and large language models to make plans even when certain objects are missing from the environment. However, this method can be slow because it relies on external LLM models for affordance mapping. In contrast, our approach employs Semantic Digital Twins (SDTs) that encapsulate object-action rules. This allows agents to plan tasks in real time without the need for training or external object reasoner, enhancing responsiveness and reducing reliance on potentially unreliable external data sources.

B. Integrating LLMs and SDTs in Complex Environments

Approaches like SayPlan utilize 3D scene graphs to ground LLM-generated plans in large-scale, multi-room environments[20, 21, 22]. Similarly, the SMART-LLM framework demonstrates using LLMs for multi-agent task planning, translating high-level instructions into coordinated actions[23]. These methods, while scalable, often depend on extensive pre-training and may struggle with real-time adaptability[24]. Our method diverges by using SDTs that inherently understand object-action relationships. These enable agents to plan and adapt in complex environments, facilitating immediate responsiveness to dynamic changes and enhancing operational efficiency.

C. Failure Detection and Recovery

Traditional failure recovery systems in robotics often combine LLMs with SDTs to detect and adapt to errors[6]. For example, REFLECT [25] uses vision-language models (VLMs) [26] conditioned on failure-related queries to generate

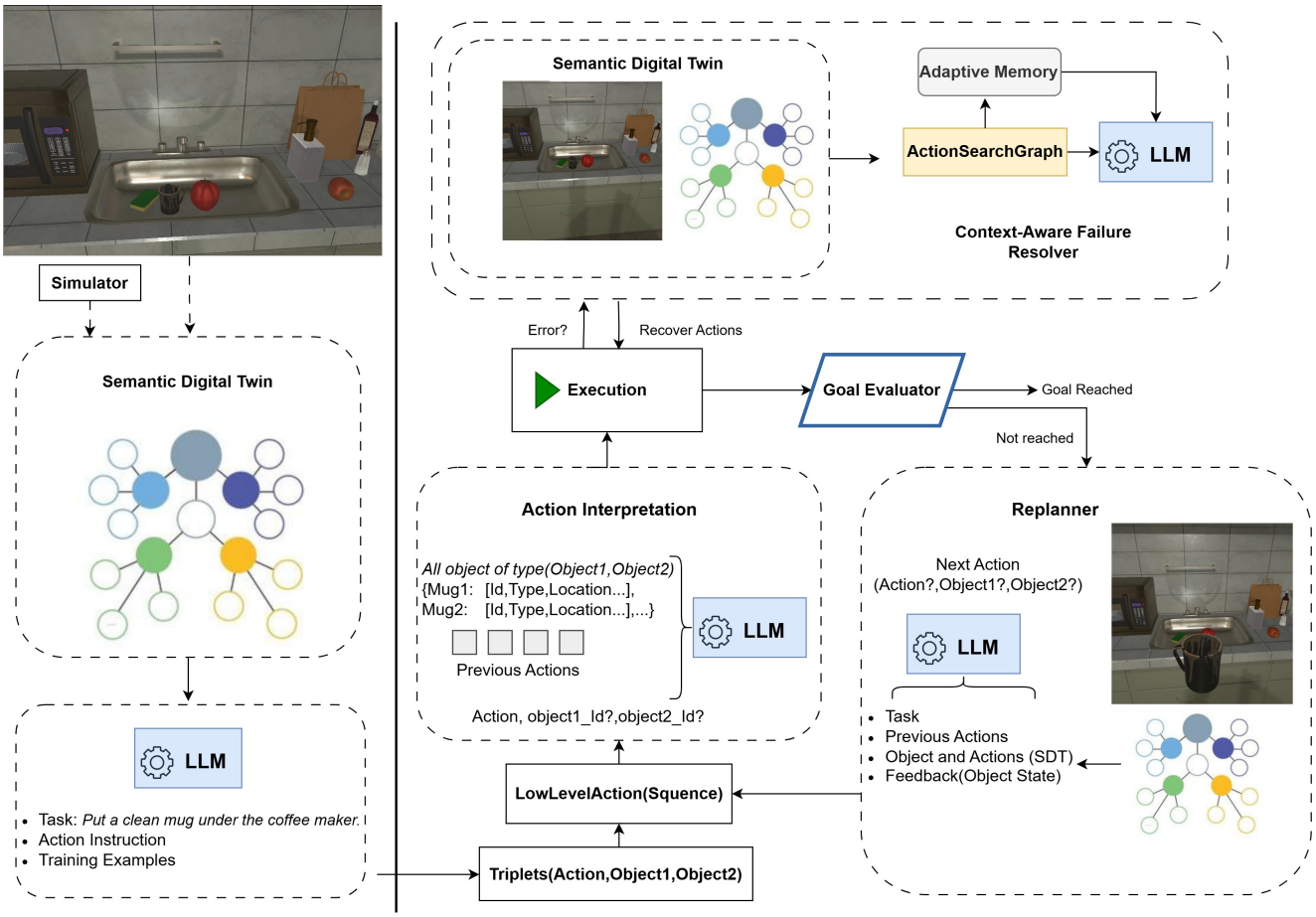

Fig. 1. LLM-based task planning with a Semantic Digital Twin. The SDT provides real-time context, enabling the LLM to generate and adapt action for goal-directed planning and execution.

summaries and re-plan tasks. Recent work such as LoTa-Bench has brought significant attention to evaluating language-based task planners in failure-prone environments[27]. LoTa-Bench provides a benchmark to assess how well LLM-based agents detect, describe, and recover from execution failures. It emphasizes querying large models during execution to reason about error types and generate recovery strategies. While LoTa-Bench has made valuable contributions in standardizing failure scenarios and offering metrics for recovery evaluation, its reliance on external querying and LLM-based introspection introduces latency and potential inconsistency during run-time execution.

III. Methodology

Enabling LLMs to operate in dynamic surroundings, reacting to changing robotic states and environments without excessive computational latency, our framework leverages a Semantic Digital Twin (SDT) to prompt a Large Language Model (LLM), enabling it to decompose the natural language description of a target task into a sequence of structured action triplets. These triplets act as step-by-step execution guidelines for the robot. Next, the agent grounds each action triplet by selecting the most contextually appropriate parameters to ensure smooth execution of the corresponding trajectory. The agent also handles potential failure cases at this step by leveraging an object-action knowledge semantic map to predict suitable recovery actions.

A. Semantic Digital Twin

The Semantic Digital Twin is a virtual representation of objects' dynamic states and properties, which are updated during execution. Our framework consists of two key components of the SDT for retrieving semantic knowledge: "Rules and Interaction Properties" and "Textual Descriptions." These components represent objects' responses to specific actions based on the surrounding context and conditions. Specifically, we utilize a structured set of rules, describing object affordances and the consequences of particular interactions. For example: Bottle, "Will break if subjected to sufficient force. Will fill up with water if placed under a running water source." We also define a structured set of permissible actions. For example, the object "Bottle" is annotated with the action properties ["Pickupable", "Fillable", "Breakable"]. These affordances indicate that the bottle can be picked up, filled (e.g., with water), or broken under certain conditions. Performing such a link explicitly to objects with corresponding

action capabilities enables the agent to create a concept map of a given task. By grounding the task description using a conceptual map derived from the Semantic Digital Twin (SDT) and leveraging the reasoning capabilities of Large Language Models (LLMs), we first predicted a structured set of action triplets for task execution and later handled the failure cases.

B. LLM-Powered Adaptive Planner

To provide Large Language Models (LLMs) with structured input for triplet prediction, we design an LLM-based agent that operates in a multi-step process. First, the agent scans the SDT environment and filters out only the objects that are relevant to the language task **L** at hand. Once these task-relevant objects are isolated, the agent queries a SDT, which supplies information about the possible actions that can be performed on or with each object, along with the actionable properties **P** associated with them (e.g., whether an object is fillable, openable, or can be picked up). With this contextual understanding, comprising the filtered set of objects, their actionable properties, and the predefined rules or contextual interactions **C** that govern permissible actions along with previous task examples **E**, the LLM agent is equipped to generate a structured output. Specifically, it predicts a sequence of action triplets, where each triplet *T(Action(a), SourceObj(o_s), TargetObj(o_t))* typically represents an action, the object it applies to, and other associated object. Training examples are task-context examples used to construct the prompt Fig. 5, providing the LLM agent with guidance on the expected type of response. Alongside this sequence, the agent also predicts the final goal condition these actions aim to achieve.

$$T(a, o_s, o_t) = \text{LLM}(L \mid E, C, P) \tag{1}$$

After establishing general guidelines for the task in the form of triplets, the next crucial step involves resolving the specific parameters of the predicted action triplets. In this context, resolving parameters means determining the most appropriate objects to associate with each high-level action in the sequence. To accomplish this, the LLM agent performs step-by-step reasoning during action execution in the Action Interpretation Engine. At each step, the agent takes into account three key inputs: (1) the current state SDT of the environment, (2) the task description, and (3) the history of previously executed actions. Using this information, the agent constructs a context-aware query to determine which object in the environment is the most suitable for the following high-level action in the triplet sequence.

C. Context-Aware Failure Handling and Replanning

If an action fails during execution, the Failure Resolver intervenes and attempts to resolve the issue by predicting alternative actions. It explores the environment and uses Action Search Graph to generate a map of available action pairs as shown in Fig. 2 and Fig. 3. The Failure Resolver Engine also incorporates adaptive memory, which tracks previously attempted solutions to avoid repeating the same ones for a specific failure point.

The Context-Aware Failure Resolver is designed to analyze errors and suggest appropriate recovery actions to address the failure. Action Search Graph takes a set of object **O**, a list of actions **A**, and filters the actions based on a condition function **Ca**.

$$A' = \{a \in A \mid C_a(O, a) = T\} \tag{2}$$

Here **O** be a list of object descriptions generated from SDT, **A**=[a1,a2,...,an] be a finite set of possible actions. **Ca(O,a)** is a boolean-valued condition function, which returns true if action a is valid for object **O** and false otherwise. The filtered action set A' consists of all actions a from the original set **A** for which the condition function **Ca(O,a)** evaluates to true. After generating all possible action pairs, they are sent to both the adaptive memory and the query generator. Adaptive memory is responsible for tracking the action pairs that have been attempted and providing feedback on them. The Query Generator then formulates a query (as illustrated in the Fig. 6) to the LLM agent to identify a potential solution. If the suggested action pair fails to resolve the error, a new query is generated, incorporating feedback from the previous attempt.

Once all predicted triplets have been executed successfully, the agent evaluates the task by comparing the current state with the goal state. If the goal has not yet been achieved, it gathers information on the actions performed so far, the current state of the objects, the task description, and feedback on the unmet goal conditions. This information is then used to prompt the LLM agent to generate additional action steps (in the form of triplets) needed to fulfill the remaining goal conditions.

IV. Experiment

We systematically evaluate our approach using examples from the ALFRED dataset[28] build on AI2-THOR[29], a benchmark designed for embodied AI agents that plan and execute actions to accomplish household tasks. These tasks include actions such as cleaning a mug, cutting vegetables, and cooling an apple in the refrigerator. For our evaluation, we selected tasks from three categories: Clean & Place, Heat & Place, and Cool & Place,as they present greater challenges in terms of reasoning and interaction complexity. The examples are drawn from the valid-seen split, which includes various interactive actions such as SliceObject, OpenObject, and ToggleOnObject. The SDT is generated from the behavior of each type of object that appears in ai2thor. https://ai2thor.allenai.org/ We use an SDT-assisted LLM to break down the language task description into triplets. Failure conditions are generated by altering the environment's ground truth from the dataset, such as making an object dirty or placing it inside a closed container.

In this work, we used the DeepSeek-r1(32B) model as a backbone of the LLM agent. The model was accessed locally through the OpenWebUI framework, which served as the interface for interaction with the LLM. Under the hood,

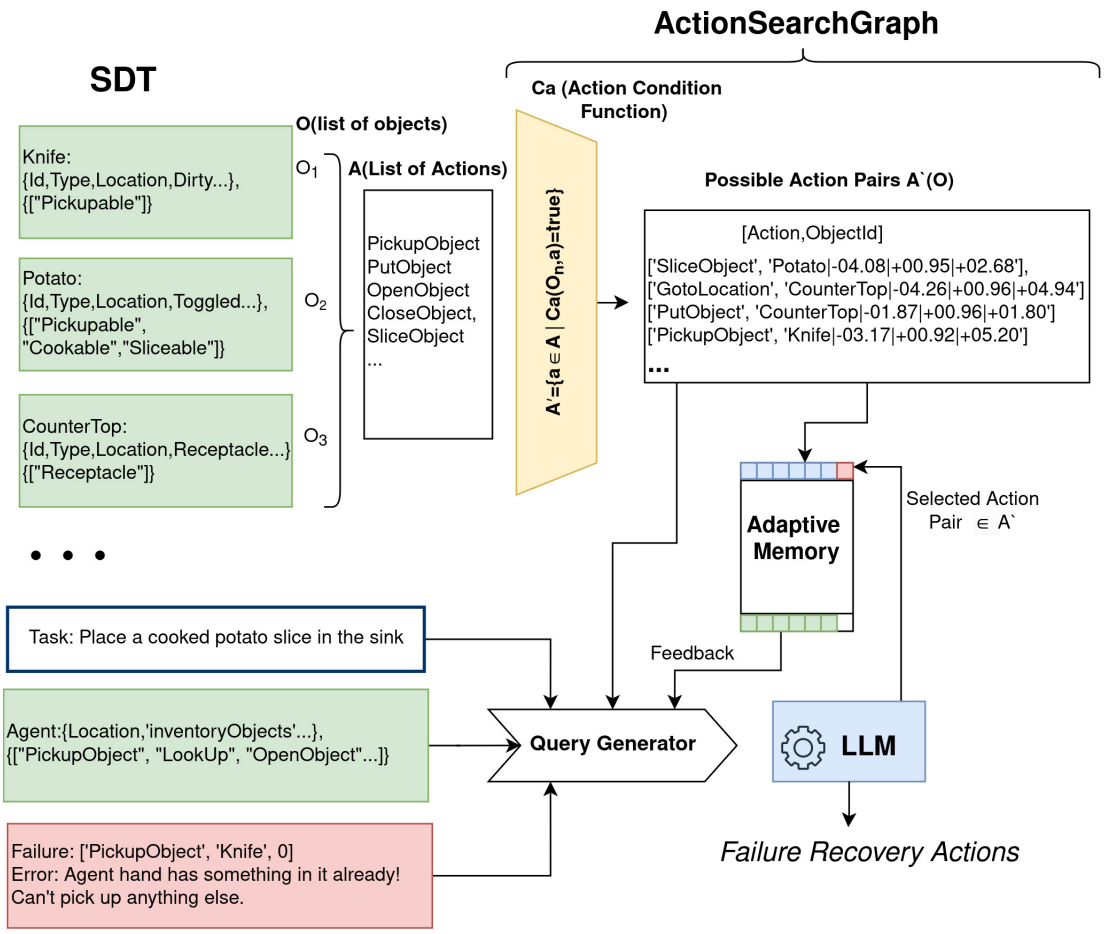

Fig. 2. Overview of the Context-Aware Failure Resolver system: SDT Object descriptions are processed with a condition function to filter possible actions from the action set (A). Filtered action pairs are passed to the Adaptive Memory and Query Generator.

OpenWebUI connects to the Ollama model runner, which provides a standardized Python and API interface for model execution. The model weights were obtained from the Hugging Face model repository and distributed through Ollama's model library for local deployment. The context window size for this model is 128k tokens, which allows for handling of long documents and extended multi-turn interactions. Furthermore, recognizing the effect of different ollama models, we also run the tasks on Llama3.3(70B) and GPT-oss(20B) with the same context window size of 128k tokens. All results reported in the experiments are based solely on the model's native reasoning capabilities, without training. The training examples mentioned in Fig. 1 are task-context examples used to construct the prompt, providing the LLM agent with guidance on the expected type of response.

V. Result

We selected a subset of 14 tasks for evaluation that are enactable by an agent. All tasks were successfully completed, demonstrating the robustness of the framework. Success was defined as achieving the specified goal condition. Although all tasks ultimately succeeded, several experienced failure cases during initial attempts. The system used an iterative correction mechanism for each failure case to refine the plan. For example, the task: Place a cooked potato slice in the sink, experienced two failure cases, each of them resolved in 1 iteration. We recorded the number of failure cases per task and the number of iterations required to resolve each failure across all tasks as shown in Table I. This allowed us to assess the planner's adaptability and the LLM's ability to generate corrective actions. Replanner iteration specifies how often the agent invoked the replanner after completing the sequence of predicted triplets in cases where the goal condition was still unmet. This does not refer to replanning after individual triplet failures, but after executing all planned steps and failing to reach the desired end state. A zero value means that the predicted triplets were sufficient to achieve the goal.The following Fig.4compares subgoal success rates across three setups: using triplets generated by the planner alone, using triplets with the failure resolver, and using triplets with the failure resolver followed by the replanner.

Based on the results, we observe that most failure cases stem from incorrect triplet predictions, while others are due to inappropriate object selection. For instance in task "Place a rinsed

TABLE I
SUMMARY OF TASK EXECUTION PERFORMANCE ON VARIOUS HOUSEHOLD TASKS

Task IDs	Task Description	No. Failure	Iteration Per Failure	Replanner Iteration	Success
1	Place a cooked potato slice in the sink	2	2	0	Yes
2	Put a cooked piece of potato in the sink.	0	0	2	Yes
3	Place a rinsed knife inside a drawer.	1	1	0	Yes
4	Slice an apple, cook it and set it on the counter	1	1	0	Yes
5	Place a clean knife in the drawer	0	0	0	Yes
6	Put a warm apple slice on the counter.	2	2	0	Yes
7	To cook a sliced tomato to throw it in the trash.	1	1	0	Yes
8	Put a chilled plate on the counter left of the sink.	2	2	0	Yes
9	Set a chilled bottle of wine on the table.	1	4	0	Yes
10	Put a clean mug under the coffee maker.	0	0	0	Yes
11	Put a clean cloth on the back of the toilet.	0	0	0	Yes
12	Put a wet sponge on the counter.	1	2	0	Yes
13	Cool a slice of bread and put it in the microwave.	0	0	0	Yes
14	Cut an apple, cool a piece and bring it to the table	1	1	2	Yes

knife inside a drawer", the agent attempts to place an object on the target but fails because the chosen target lacks sufficient space. In such scenarios, the agent needs to reconsider and revise its object selection. For tasks where the goal condition was not satisfied after initial execution, the Replanner Module was triggered. This component analyzed the current state and generated additional corrective actions using the LLM. We recorded the number of replanning iterations required per task. Notably, some tasks (such as the heating object examples) did not require any replanning, indicating that the initial plan and failure resolver were sufficient. However, if a goal condition is not met—for example, the agent performs the "heat the potato" action but misses the "slice" action, even though the goal requires the object to be both hot and sliced—then the replanner module steps in. It suggests the agent pick up a knife and perform the slicing action, thereby fulfilling the goal condition.

The purpose of our structured "Rules and Interaction Properties" goes beyond encoding general object behavior. These rules are explicitly designed to reflect the capabilities and constraints of specific robot agents and simulation environments. For example. While an LLM may know that a bottle can be picked up, filled, or broken, the actual feasibility of these interactions depends on the agent's embodiment and the simulation's physics fidelity. A bottle labeled as "Pickupable" in our framework means that it is semantically and mechanically feasible for a particular robot in a given simulator to perform the action — accounting for reachability, gripper type, force limits, etc. Our framework captures contextual dependencies and interaction consequences that may not be static or universally known. For example, whether an object breaks may depend on the surface it's dropped on, or whether it fills depends on proximity to a simulated water source — details that require structured specification rather than reliance on general commonsense priors. Thus, the rules provide grounded, operationalizable knowledge that is critical for agents operating in simulated or real environments where

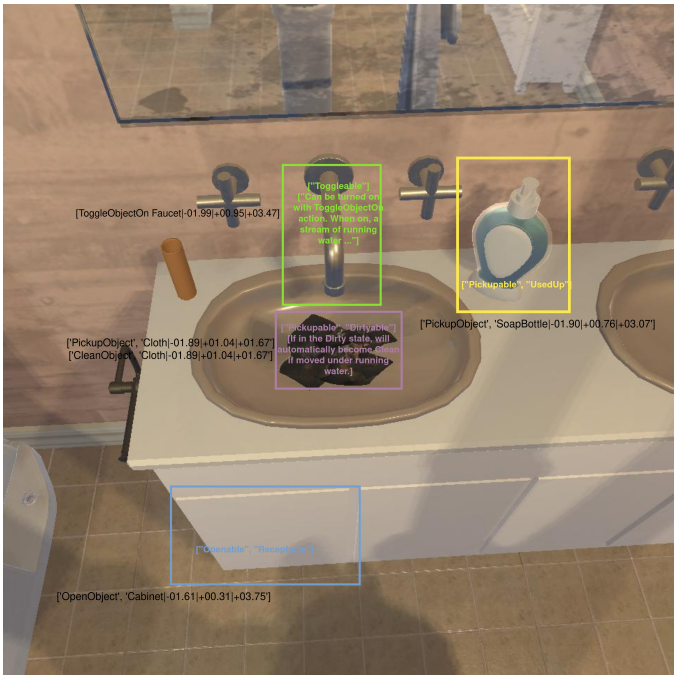

Fig. 3. Semantic Digital Twin based on "Rules and Interaction Properties" and "Textual Descriptions"

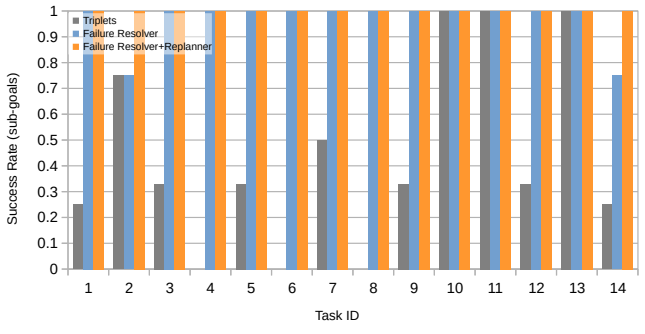

Fig. 4. Comparison of task (see Table I) success with and without Failure Resolver and Re-planner

affordances are not just about what "should" happen but what can happen under defined system constraints. The result of the individual impact on overall performance is presented in Table II. Without SDT, tasks achieve only a 57% success rate, despite requiring more failures and re-planning iterations. In contrast, SDT task planning improves performance in all metrics. Deepseek takes fewer failure iterations as compared to other listed models, whereas GPT-oss requires the least re-planning iterations to complete the task.

TABLE II
COMPARISON OF PERFORMANCE WITH AND WITHOUT SDT.

LLM	**Without SDT**	**SDT**		
		Deepseek	***Llama-3.3***	***GPT-oss***
Avg. failure itr. ↓	17.14	1.14	2.5	1.28
Avg. re-planning itr. ↓	1.57	0.28	0.4	0.2
Success rate ↑	57%	100%	92.86%	100%

VI. DISCUSSION

In earlier research, failures were typically addressed by summarizing the actions taken and then replanning the entire task from the beginning[25]. In contrast, LOTA-Benchmark [27] examined six failure cases and aimed to resolve them through demonstration-based feedback and example-driven replanning. REFLECT [25] and Replan [3] both support task verification and planning in a dynamic environment, but they lack a real-time failure resolver with SDT. In Sayplan [20], task planning is constrained by the 3D scene graph, which is built on static objects and lacks dynamic SDT. All of these approaches significantly restrict their adaptability to OpenAI models with a post-task failure resolver. In contrast, our method supports various features, including an open-source LLM, task verification, real-time failure resolver, and replanning in a dynamic environment by considering the executed actions' history.

TABLE III
FEATURE COMPARISON OF OUR FRAMEWORK WITH OTHER TASK PLANNING BENCHMARKS

Features	**REFLECT[25]**	**SayPlan**[20]	**RePlan**[3]	**Ours**
Adaptive Memory (tracks failed attempts)	✗	✗	✗	✓
Open-Source LLM Compatibility	✗	✗	✗	✓
Real-time Adaptation to Dynamic Changes	✓	✗	✓	✓
Real-time Failure Resolver	✗	✗	✓	✓

None of these approaches consider SDT based on object-actionable properties to predict context-aware actions. Our proposed method addresses failure cases such as the absence of visual grounding (e.g., attempting to interact with invisible or non-present objects) and object selection errors (e.g., choosing the wrong object). Moreover, the Action interpretation Engine helps execute the action triplets by determining whether interacting with an object logically follows previous actions. Its purpose is to ensure that each triplet is valid on its own and contributes meaningfully to accomplishing the overall task.This is achieved through the use of a Semantic Digital Twin-based ActionSearch Graph, which enhances the agent's understanding of object properties and contextual relevance.

VII. CONCLUSION

This work introduces an adaptive framework that synergistically combines Semantic Digital Twins (SDTs) with Large Language Models (LLMs) to enhance robotic task planning and execution in dynamic, real-world environments. By representing tasks as structured action triplets grounded in semantic and contextual information, our system enables execution, real-time error recovery, and iterative replanning. Experimental evaluations using the ALFRED dataset validate the effectiveness of our approach in handling complex household tasks, even in scenarios involving uncertainty or unexpected failures. Unlike prior methods, our framework leverages the rich semantics of SDTs to inform context-aware decision-making, eliminating reliance on external affordance mapping or retraining. This integration bridges the gap between high-level language reasoning and low-level robotic control, setting a new direction for autonomous agents capable of reliable, interpretable, and resilient behavior. Future work will explore expanding the semantic representation of environments, and incorporating multimodal sensory feedback for more nuanced task understanding.

ACKNOWLEDGMENTS

This research was partially funded by the German Research Foundation DFG, as part of Collaborative Research Center (Sonderforschungsbereich) "EASE - Everyday Activity Science and Engineering", University of Bremen.

REFERENCES

[1] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li, "Instruct2act: Mapping multi-modality instructions to robotic actions with large language model," 2023. [Online]. Available: https://arxiv.org/abs/2305.11176

[2] Y. Wang, Z. Xian, F. Chen, T.-H. Wang, Y. Wang, K. Fragkiadaki, Z. Erickson, D. Held, and C. Gan, "Robogen: Towards unleashing infinite data for automated robot learning via generative simulation," 2024. [Online]. Available: https://arxiv.org/abs/2311.01455

[3] M. Skreta, Z. Zhou, J. L. Yuan, K. Darvish, A. Aspuru-Guzik, and A. Garg, "Replan: Robotic replanning with perception and language models," 2024. [Online]. Available: https://arxiv.org/abs/2401.04157

[4] C. Xie and D. Zou, "A human-like reasoning framework for multi-phases planning task with large language models," 2024. [Online]. Available: https://arxiv.org/abs/2405.18208

[5] V. Bhat, A. U. Kaypak, P. Krishnamurthy, R. Karri, and F. Khorrami, "Grounding llms for robot task planning using closed-loop state feedback," 2024. [Online]. Available: https://arxiv.org/abs/2402.08546

[6] A. A. Khan, M. Andrev, M. A. Murtaza, S. Aguilera, R. Zhang, J. Ding, S. Hutchinson, and A. Anwar, "Safety aware task planning via large language models in robotics," 2025. [Online]. Available: https://arxiv.org/abs/2503.15707

[7] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, and M. Aiello, "Delta: Decomposed efficient long-term robot task planning using large language models," 2025. [Online]. Available: https://arxiv.org/abs/2404.03275

[8] A. Melnik, B. Alt, G. Nguyen, A. Wilkowski, Q. Wu, S. Harms, H. Rhodin, M. Savva, M. Beetz *et al.*, "Digital twin generation from visual data: A survey," *arXiv preprint arXiv:2504.13159*, 2025.

[9] S. Yang, J. Liu, R. Zhang, M. Pan, Z. Guo, X. Li, Z. Chen, P. Gao, Y. Guo, and S. Zhang, "Lidar-llm: Exploring the potential of large language models for 3d lidar understanding," 2023. [Online]. Available: https://arxiv.org/abs/2312.14074

[10] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa, C. Gan, C. M. de Melo, J. B. Tenenbaum, A. Torralba, F. Shkurti, and L. Paull, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," 2023. [Online]. Available: https://arxiv.org/abs/2309.16650

[11] T. Chen, Y. Mu, Z. Liang, Z. Chen, S. Peng, Q. Chen, M. Xu, R. Hu, H. Zhang, X. Li, and P. Luo, "G3flow: Generative 3d semantic flow for pose-aware and generalizable object manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2411.18369

[12] F. Zeng, W. Gan, Y. Wang, N. Liu, and P. S. Yu, "Large language models for robotics: A survey," 2023. [Online]. Available: https://arxiv.org/abs/2311.07226

[13] A. Baratta, A. Cimino, L. Gazzaneo, L. Nicoletti, and V. Solina, "Conceptual modeling for a simulation-based digital twin in human-robot collaboration," *Procedia Computer Science*, vol. 253, pp. 3247–3256, 01 2025.

[14] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, Q. Vuong, T. Zhang, T.-W. E. Lee, K.-H. Lee, P. Xu, S. Kirmani, Y. Zhu, A. Zeng, K. Hausman, N. Heess, C. Finn, S. Levine, and B. Ichter, "Pivot: Iterative visual prompting elicits actionable knowledge for vlms," 2024. [Online]. Available: https://arxiv.org/abs/2402.07872

[15] A. Melnik, M. Büttner, L. Harz, L. Brown, G. C. Nandi, A. PS, G. K. Yadav, R. Kala, and R. Haschke, "Uniteam: Open vocabulary mobile manipulation challenge," *arXiv preprint arXiv:2312.08611*, 2023.

[16] Y. Mikami, A. Melnik, J. Miura, and V. Hautamäki, "Natural language as policies: Reasoning for coordinate-level embodied control with llms," *arXiv preprint arXiv:2403.13801*, 2024.

[17] A. PS, A. Melnik, and G. C. Nandi, "Splatr: Experience goal visual rearrangement with 3d gaussian splatting and dense feature matching," *arXiv preprint arXiv:2411.14322*, 2024.

[18] S. Yenamandra, A. Ramachandran, M. Khanna, K. Yadav, J. Vakil, A. Melnik, M. Büttner, L. Harz, L. Brown, G. C. Nandi *et al.*, "Towards open-world mobile manipulation in homes: Lessons from the neurips 2023 home-robot open vocabulary mobile manipulation challenge," *arXiv preprint arXiv:2407.06939*, 2024.

[19] T. Birr, C. Pohl, A. Younes, and T. Asfour, "Autogpt+p: Affordance-based task planning using large language models," in *Robotics: Science and Systems XX*, ser. RSS2024. Robotics: Science and Systems Foundation, Jul. 2024. [Online]. Available: http://dx.doi.org/10.15607/RSS.2024.XX.112

[20] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable robot task planning," 2023. [Online]. Available: https://arxiv.org/abs/2307.06135

[21] K. Rana, A. Melnik, and N. Sünderhauf, "Contrastive language, action, and state pre-training for robot learning," *arXiv preprint arXiv:2304.10782*, 2023.

[22] P. Arjun, A. Melnik, and G. C. Nandi, "Cognitive planning for object goal navigation using generative ai models," in *NeurIPS 2024 Workshop on Open-World Agents*.

[23] S. S. Kannan, V. L. N. Venkatesh, and B.-C. Min, "Smart-llm: Smart multi-agent robot task planning using large language models," 2024. [Online]. Available: https://arxiv.org/abs/2309.10062

[24] R. Abe, A. Ito, K. Takayasu, and S. Kurihara, "Llm-mediated dynamic plan generation with a multi-agent approach," 2025. [Online]. Available: https://arxiv.org/abs/2504.01637

[25] Z. Liu, A. Bahety, and S. Song, "Reflect: Summarizing robot experiences for failure explanation and correction," 2023. [Online]. Available: https://arxiv.org/abs/2306.15724

[26] D. Driess, F. Xia, M. S. M. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, W. Huang, Y. Chebotar, P. Sermanet, D. Duckworth, S. Levine, V. Vanhoucke, K. Hausman, M. Toussaint, K. Greff, A. Zeng, I. Mordatch, and P. Florence, "Palm-e: An embodied multimodal language model," 2023. [Online]. Available: https://arxiv.org/abs/2303.03378

[27] J.-W. Choi, Y. Yoon, H. Ong, J. Kim, and M. Jang, "Lota-bench: Benchmarking language-oriented task planners for embodied agents," 2024. [Online]. Available: https://arxiv.org/abs/2402.08178

[28] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," 2020. [Online]. Available: https://arxiv.org/abs/1912.01734

[29] "AI2-THOR: An Interactive 3D Environment for Visual AI," *ArXiv*, vol. abs/1712.05474, 2017.

APPENDIX

Generate a list of action triplets planner for executing the following Task:Put a clean mug under the coffee maker.
Only use the actions listed here: ['PickupObject', 'PutObject', 'OpenObject', 'CloseObject', 'SliceObject', 'ToggleObjectOn', 'ToggleObjectOff', 'EmptyLiquidFromObject']]
Select appropriate object w.r.t to action from the Object_List by analyzing their Object_actionable_properties and Object_rules
---
Where as:
For cleaning,rinse or washing, put object in the SinkBasin first and then toggled the Faucet.
[Object_List]['CoffeeMachine', 'Mug', 'ButterKnife', 'Knife', 'CounterTop', 'Microwave', 'Fridge']

[Object_actionable_properties]{#OBJECTS_ACTIONABLE_PROP#}

[Object_rules]{#OBJECTS_PROP#}

[Actions_rules]Basic Instructions about actions selection w.r.t to objects actionable properties
PickupObject: allows the agent to pick up an interactable object specified by its objectId. Compatible objects have "Pickupable" as actionable property
PutObject: attempts to put an object the agent is holding onto or into the target receptacle. Valid target receptacle objects have "Receptacle" as actionable property.
OpenObject: attempts to open an object having "Openable" as actionable property.
CloseObject: attempts to close an object having "Openable" as actionable property.
SliceObject: attempts to slice an object having "Sliceable" as actionable property. If an object, such as an Apple, is successfully sliced, there may include several new AppleSliced objects in the scene. So Apple Changed to AppleSliced after SliceObject.
ToggleObjectOn: toggles an object on if object is "Toggleable"
ToggleObjectOff: toggles an object off if object is "Toggleable"
EmptyLiquidFromObject: attempts to empty the liquid from an object which has "Fillable" as actionable property and fill with liquid previously.

Example:
Task1:[#TASK_DESCRIPTION#]
Output1:
[#TRIPLETS#]
Task2:[#TASK_DESCRIPTION#]
Output2:
[#TRIPLETS#]
Return a Python-style list containing only the list of triplets e.g. [['Action', 'Object1', 'Object2']]. Do not include any explanations.

Fig. 5. Example of Triplet Prompt

[Query] ---
Response should be like this Reason(string),ActionPair(list)
Example#Reason:The agent needs to go to the Fridge.,ActionPair['GotoLocation', 'Fridge|-00.32|00.00|+03.60']

Agent Fails to execute Action Pair ('PickupObject', 'Knife|-03.17|+00.92|+05.20') and
got Error: Agent hand has something in it already! Can't pick up anything else..
Where as Knife|-03.17|+00.92|+05.20 is not visible, is in/on('CounterTop|-04.26|+00.96|+04.94',)
and agent is near to Drawer|-04.09|+00.78|+04.66,Fork|-03.20|+00.92|+05.46,Knife|-03.17|+00.92|+05.20,
Spatula|-03.91|+00.92|+05.17,Spoon|-03.17|+00.91|+05.37, and holding Potato|-04.08|+00.95|+02.68.

Please select replanner action pair from given list : [['SliceObject', 'Potato|-04.08|+00.95|+02.68'],
['PickupObject', 'Knife|-03.17|+00.92|+05.20'], ['Crouch', 'Knife|-03.17|+00.92|+05.20'],
['Stand', 'Knife|-03.17|+00.92|+05.20'], ['PutObject', 'CounterTop|-04.26|+00.96|+04.94'],
['GotoLocation', 'CounterTop|-04.26|+00.96|+04.94']]

[Feedback]Already tried action pairs are : [['SliceObject', 'Potato|-04.08|+00.95|+02.68']]

[Response]

Reason: The agent needs to free its hand by putting down the potato before attempting another pickup.
ActionPair: ['PutObject', 'CounterTop|-04.26|+00.96|+04.94']

Fig. 6. Example of Failure Query