AdaTAMP: Adaptive Task and Motion Planning with LLM-based Embodied Agents

Karan Baijal* Zhiwen (Calvin) Qiu* Jennifer Sun

***Abstract*—We introduce AdaTAMP, an Adaptive Task and Motion Planning (TAMP) framework leveraging Large Language Models (LLMs) for embodied agents in dynamic environments. AdaTAMP integrates symbolic task planning with continuous motion planning through a real-time self-feedback loop, enabling efficient error correction and seamless multi-agent collaboration. Evaluations conducted in the VirtualHome simulator demonstrate that AdaTAMP significantly outperforms baseline methods in success rate, planning efficiency, and adaptability, particularly for complex, long-horizon, multi-agent scenarios.**

I. Introduction

Successful execution of long-horizon tasks by embodied agents demand reasoning over sequences of dependent actions, ensuring feasibility given environmental and physical constraints [11], [12]. Task and Motion Planning (TAMP) addresses this by hierarchically dividing the planning problem into symbolic (high-level) planning and continuous motion (low-level) planning. Classical approaches, such as those using Planning Domain Definition Language (PDDL) [17], often require extensive manual domain specification, lack adaptability to real-time changes, and struggle with complex, long-term planning tasks.

Recent advancements in Large Language Models (LLMs), like GPT-4, have shown promise in decomposing natural language instructions into executable subtasks. While effective at symbolic task planning, LLM-based approaches typically face difficulties translating high-level instructions into precise motion commands and adapting to dynamic constraints. Frameworks such as AutoTAMP [6] and CoELA [27] have attempted to address these gaps; however, they often neglect *continuous* motion integration and lack a robust *real-time feedback* mechanism, making it difficult to generalize to complex multi-agent coordination in dynamic, multi-room environments.

To address these limitations, we present AdaTAMP, an LLM-based adaptive TAMP framework that combines continuous motion planning with a self-feedback loop for real-time correction to enhance the integration of symbolic task planning and motion execution for embodied agents in household environments. Our contribution can be summarized as follows: (1) We present AdaTAMP framework, which integrates symbolic task sequences with continuous motion planning, addressing both navigation and task execution for embodied agents in a multi-agent setting; (2) We incorporate a self-feedback loop that allows for real-time corrections based on motion feedback to adapt to failures and dynamic environment changes; (3) We perform a preliminary analysis of the framework's effectiveness in navigating and handling dynamic scenarios in household environments, demonstrating its scalability and practicality for multi-agent interactions across different metrics.

*Equal contribution. All authors are with Cornell University, Ithaca, NY, 14850

II. Method

Problem Description. We aim to convert natural-language instructions into executable motion plans for multiple embodied agents, respecting spatial, temporal, and physical constraints. The environment is represented as a graph of objects (with positions, orientations, etc.), and we employ a closed-loop self-feedback mechanism: whenever an action fails, the plan is updated and the agent's trajectory τ is adjusted based on environmental feedback f_t via

$$\tau_{t+1} = \tau_t + \Delta(\tau_t, f_t),$$

to recover and continue toward the goal state.

Multi-Agent Task Planning with LLM. We employ GPT-4 [3] to generate high-level task sequences (e.g., clean the room) from natural language instructions. To enable the LLM to understand the environment information symbolically, we leverage grounding information from the simulated environment. This includes translating the graph representations of nodes (i.e., objects) and edges (i.e., relationships between objects) into natural language descriptions that encapsulate object properties, locations, and spatial relationships (e.g., mug is *on* the kitchen table).

Continuous Motion Planning. For each symbolic step, we run A* [24], [28] on the simulator's NavMesh to compute collision-free, shortest-path trajectories. The planner continuously rechecks and adjusts paths to avoid inter-agent conflicts and dynamic obstacles, allowing concurrent execution.

Synthesizing Environment Feedback. When execution fails (e.g., due to collisions or unreachable targets), we generate semantically meaningful feedback at both agent and task levels. This feedback refines individual motion plans without restarting the entire task, maintaining team coordination and progress.

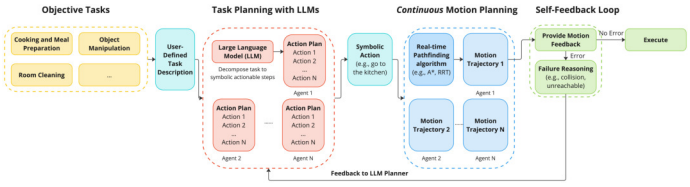

Fig. 1. AdaTAMP framework for continuous task and motion planning. It consists of three steps: high-level task planning with LLMs for multi-agents, motion planning for low-level actions, and self-feedback loop for failure correction.

III. Experiments

Experiment Setup. We evaluate on 20 VirtualHome tasks (6 easy, 7 medium, 7 hard), from simple pick-and-place (e.g., "pick up an apple") to long-horizon multi-step tasks (e.g., "get the water glass...put it on the coffee table"). Agents run in single and multi agent modes (multiple agents only on medium and hard tasks) with randomized start positions and number of agents in object-rich regions. For each task, GPT-4 is prompted (e.g., "set up the table") to generate a plan, which AdaTAMP executes using A* motion planning plus closed-loop feedback. We compare against a one-shot LLM plan baseline without feedback or dynamic replanning.

Evaluation Metrics. We measure: Final Success Rate (task goal achieved), Subgoal Success Rate (fraction of intermediate goals met), Completion Time (total duration), Action Count Efficiency (actions vs. optimal), Levenshtein Distance [29] (edit distance to ground-truth plan), and Coordination Score (percentage of overlapping agent steps).

Results. In the single-agent scenario, AdaTAMP demonstrated superior performance compared to the baseline across varying task difficulties, as can be seen in Figure 2. This emphasizes the critical role of our feedback loop and TAMP strategy for better task and motion planning. AdaTAMP exhibited better adaptability compared to baseline, as can be seen from its lower Levenshtein Distance scores. These results underscored the framework's capability to execute sequential tasks with greater precision and adaptability in dynamic environments for single agent.

In the multi-agent scenario, AdaTAMP showed notable performance improvements over the baseline. As no ground truth exists for direct comparison, Levenshtein Distance could not be evaluated in this context. Action count efficiency was generally comparable between AdaTAMP and the baseline; however, AdaTAMP outperformed the baseline specifically in hard tasks and demonstrated better performance in multi-agent settings. This improvement aligns with expectations, as the integration of environmental feedback enables AdaTAMP to select more optimal actions, a capability that becomes particularly critical for handling the complexity of hard tasks.

AdaTAMP most often failed due to inadequate understanding of the environment from the LLM. For example, given a task "pick up the cup," the LLM may give a subtask to pick up the water cup from a place where the cup doesn't exist. Another place of failure, albeit rare, is due to incorrect sequence planning (e.g., taking out fridge item before opening fridge) by the LLM.

When transitioning from single-agent to two-agent scenarios, AdaTAMP maintained consistent performance for medium tasks. However, the two-agent and multi-agent setting saw a notable improvement in final success rates (85.7%) compared to the single-agent scenario (42.8%) in hard tasks, demonstrating its ability to effectively handle long-horizon planning and inter-agent dependencies. Metrics of coordiation and final success rate indicate that similar to humans, splitting and executing tasks to solve a common goal becomes more efficient for AdaTAMP with multiple agents.

Task Level		Final Success Rate	Subgoal Success Rate	Action Count Efficiency	Completion Time	Levenshtein Distance
Easy	AdaTAMP	**100%**	**91.7%**	89%	-20.4%	**0.67**
	baseline	66.7%	83.3%	94.5%	**-15.2%**	0.83
Medium	AdaTAMP	**71.4%**	**70.3%**	84.3%	-35.3%	**4.7**
	baseline	57.1%	64.7%	85%	**-40.2%**	5.1
Hard	AdaTAMP	**42.8%**	**55.6%**	**78.1%**	**-49.2%**	**4.7**
	baseline	14.3%	52%	75.6%	-45.2%	4.9
Overall	AdaTAMP	**70%**	**72.5%**	83.6%	-33%	**3.5**
	baseline	45%	65.5%	**84.5%**	**-45%**	3.8

Fig. 2. Results comparing baseline planning methods and AdaTAMP in single-agent scenario.

Task Level		Final Success Rate	Subgoal Success Rate	Action Count Efficiency	Completion Time	Coordination (parallelization)
Medium	AdaTAMP	57.1%	75.3%	**87.4%**	24%	75%
	baseline	57.1%	75.3%	86.4%	24.7%	75%
Hard	AdaTAMP	**85.7%**	88.6%	**117.57%**	39.1%	**69.1%**
	baseline	71.4%	93%	110.7%	42.4%	65.6%
Overall	AdaTAMP	**76.9%**	88.2%	**107.3%**	33.9%	**0.776**
	baseline	64.3%	**90.6%**	106.2%	**36.1%**	0.756

Fig. 3. Results comparing baseline planning methods and AdaTAMP in two-agent scenario.

Task Level		Final Success Rate	Subgoal Success Rate	Action Count Efficiency	Completion Time	Coordination (parallelization)	Average Number of Agents
Hard	AdaTAMP	**100%**	**100%**	**140.7%**	52.8%	**66.7%**	2.8
	baseline	85.7%	90.8%	136.7%	44.2%	65.2%	2.8

Fig. 4. Results comparing baseline planning methods and AdaTAMP in multi-agent scenario.

IV. Conclusion

We presented AdaTAMP, an LLM-based task and motion planning framework enhanced with a self-feedback loop for real-time correction. Our results show that AdaTAMP performs well in dynamic environments and significantly improves performance in multi-agent settings, especially for complex, long-horizon tasks. We are working on validating AdaTAMP on real-world robots for both robot-robot and human-robot collaboration remains an exciting direction.

References

[1] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.

[2] J. E. Laird, A. Newell, and P. S. Rosenbloom, "Soar: An architecture for general intelligence," *Artificial Intelligence*, vol. 33, no. 1, pp. 1–64, 1987.

[3] J. Achiam, S. Adler, S. Agarwal, *et al.*, "GPT-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[4] H. Touvron, T. Lavril, G. Izacard, *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "LLM-planner: Few-shot grounded planning for embodied agents with large language models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 2998–3009.

[6] Y. Chen, J. Arkin, C. Dawson, Y. Zhang, N. Roy, and C. Fan, "AutoTAMP: Autoregressive task and motion planning with LLMs as translators and checkers," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pp. 6695–6702, 2024.

[7] Y. Wu, Y. Fan, P. P. Liang, A. Azaria, Y. Li, and T. M. Mitchell, "Read and reap the rewards: Learning to play Atari with the help of instruction manuals," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[8] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[9] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, *et al.*, "Self-refine: Iterative refinement with self-feedback," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[10] G. Kim, P. Baldi, and S. McAleer, "Language models can solve computer tasks," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[11] V. N. Hartmann, A. Orthey, D. Driess, Ö. S. Oguz, and M. Toussaint, "Long-horizon multi-robot rearrangement planning for construction assembly," *IEEE Trans. Robotics*, vol. 39, no. 1, pp. 239–252, 2022.

[12] Z. Zhou, J. Song, K. Yao, Z. Shu, and L. Ma, "ISR-LLM: Iterative self-refined large language model for long-horizon sequential task planning," in *2024 IEEE Int. Conf. Robotics and Automation (ICRA)*, 2024, pp. 2081–2088.

[13] Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan, "Embodied task planning with large language models," *arXiv preprint arXiv:2307.01848*, 2023.

[14] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *Proc. Int. Conf. Machine Learning*, 2022, pp. 9118–9147.

[15] S. Wang, M. Han, Z. Jiao, Z. Zhang, Y. N. Wu, S.-C. Zhu, and H. Liu, "LLM³: Large language model-based task and motion planning with motion failure reasoning," *arXiv preprint arXiv:2403.11552*, 2024.

[16] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez, "Integrated task and motion planning," *Annu. Rev. Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 265–293, 2021.

[17] M. Fox and D. Long, "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *J. Artif. Intell. Res.*, vol. 20, pp. 61–124, 2003.

[18] E. A. Emerson, "Temporal and modal logic," in *Formal Models and Semantics*, 1990, pp. 995–1072.

[19] Z. Jiao, Y. Niu, Z. Zhang, S.-C. Zhu, Y. Zhu, and H. Liu, "Sequential manipulation planning on scene graph," in *2022 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2022, pp. 8203–8210.

[20] W. Huang, F. Xia, D. Shah, D. Driess, A. Zeng, Y. Lu, P. Florence, I. Mordatch, S. Levine, K. Hausman, *et al.*, "Grounded decoding: Guiding text generation with grounded models for embodied agents," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[21] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, and M. Aiello, "Delta: Decomposed efficient long-term robot task planning using large language models," *arXiv preprint arXiv:2404.03275*, 2024.

[22] H. Sun, Y. Zhuang, L. Kong, B. Dai, and C. Zhang, "AdaPlanner: Adaptive planning from feedback with language models," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[23] J.-W. Choi, Y. Yoon, H. Ong, J. Kim, and M. Jang, "Lota-bench: Benchmarking language-oriented task planners for embodied agents," *arXiv preprint arXiv:2402.08178*, 2024.

[24] E. Latif, "3P-LLM: Probabilistic path planning using large language model for autonomous robot navigation," *arXiv preprint arXiv:2403.18778*, 2024.

[25] X. Puig, K. Ra, M. Boben, J. Li, T. Wang, S. Fidler, and A. Torralba, "VirtualHome: Simulating household activities via programs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognition*, 2018, pp. 8494–8502.

[26] M. Li, S. Zhao, Q. Wang, K. Wang, Y. Zhou, S. Srivastava, C. Gokmen, T. Lee, L. E. Li, R. Zhang, *et al.*, "Embodied agent interface: Benchmarking LLMs for embodied decision making," *arXiv preprint arXiv:2410.07166*, 2024.

[27] H. Zhang, W. Du, J. Shan, Q. Zhou, Y. Du, J. B. Tenenbaum, T. Shu, and C. Gan, "Building cooperative embodied agents modularly with large language models," *arXiv preprint arXiv:2307.02485*, 2023.

[28] A. V. Goldberg, H. Kaplan, and R. F. Werneck, "Reach for A*: Efficient point-to-point shortest path algorithms," in *2006 Proc. Eighth Workshop Algorithm Engineering and Experiments (ALENEX)*, 2006, pp. 129–143.

[29] L. Yujian and B. Liu, "A normalized Levenshtein distance metric," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1091–1095, 2007.