Trajectories caching for path planning in variable environments

Vitaliy Yakovyna[†] and Andrii Medvid[*,†]

Lviv Polytechnic National University, Stepan Bandera Street 12 79013 Lviv, Ukraine

Abstract

Trajectory planning for high-degree-of-freedom manipulators in dynamic and partially predictable environments is often time-consuming and unsuitable for real-time robotics applications. To address this limitation, we propose a trajectory caching-based technique that accelerates the planning process by storing and reusing entire previously computed trajectories when similar motion queries are encountered. Unlike traditional approaches that rely on reusing individual states or initializing optimization-based planners with prior paths, our technique retrieves full feasible trajectories based on a simple yet effective geometric similarity metric applied to the start and goal configurations. The retrieved trajectories undergo an adaptation phase, including endpoint substitution and optional re-anchoring of the mobile base when required. To ensure robustness under stochastic execution conditions and environmental variability, all reused trajectories are subject to high-resolution tracing and collision checking before execution. This validation step guarantees safety in real-world deployments. The method is integrated with a fallback planner, enabling graceful degradation and completeness in cases where cached trajectories are not applicable. We evaluated our approach using a simulated mobile cleaning robot operating in public restrooms, where tasks such as opening stall doors, manipulating toilet seats, and spraying fluids require fast and repeated trajectory planning under varying environmental conditions. In addition to simulation, we validated the benefits of pre-training the trajectory cache under multiple environment configurations and tested the method on a real robot. Experimental results show that trajectory caching significantly reduces planning time – by more than a factor of four in some scenarios – without compromising safety or execution quality. These findings demonstrate that trajectory caching is a practical, efficient solution for repetitive service robotics tasks in partially structured and variable environments.

Keywords

trajectory planning, motion reuse, robot manipulator, collision avoidance, cache-based planning

1. Introduction

Planning collision-free trajectories for high-degree-of-freedom manipulators in dynamic and constrained environments remains a computationally expensive task, particularly when real-time performance is required. This problem becomes critical in service robotics scenarios, where delays in planning can directly affect the responsiveness and reliability of robotic behavior.

In this work, we consider a cleaning robot operating in public bathrooms, executing tasks such as opening stall doors, raising and lowering toilet seats, spraying chemicals and water, and vacuuming liquids. Some of these tasks require the robot's manipulator to find collision-free trajectories under the time constraints. In our experiments, we observed that trajectory planning may take several seconds in challenging environments, during which the robot becomes partially unresponsive. Reducing this delay is crucial for maintaining fluid task execution.

A technique for trajectory reuse through caching the entire trajectories is proposed. The key insight is that many planned trajectories exhibit structural similarity, especially when similar tasks are repeated in similar contexts. By caching previously computed trajectories and retrieving them

[*]*AISSLE-2025: International Workshop on Applied Intelligent Security Systems in Law Enforcement, October, 30–31, 2025, Vinnytsia, Ukraine*

[1*] Corresponding author.

[†] These authors contributed equally.

vitaliy.s.yakovyna@lpnu.ua (V. Yakovyna); andrii.y.medvid@lpnu.ua (A. Medvid)

0000-0003-0133-8591 (V. Yakovyna); 0000-0001-9044-6505 (A. Medvid)

based on proximity to the current request, we aim to reduce planning time without compromising safety.

This approach was implemented and evaluated in a simulated environment developed by Somatic Holdings Ltd., designed for benchmarking a bathroom-cleaning robot. Experiments show that our method reduces overall trajectory planning time by more than three times in repeated runs in simulation. Also using cache precalculating with simulation was tested on the real robot.

The contributions are as follows:

- A cache-based trajectory reuse mechanism that supports online selection of previously successful plans was designed.
- Was defined a proximity-based metric to evaluate trajectory relevance and describe an adaptation procedure to replace endpoints and, if necessary, re-anchor the base.
- The method was validated in a realistic simulation and on a real robot and detailed experimental results were presented.
- Shown the ability of using a cache that has been pre-trained in simulation to speed up path planning by a robot in real life.

2. Related works

A range of methods has been proposed to accelerate trajectory planning through the reuse of previously computed plans. These approaches can be categorized by the structure and complexity of their reuse mechanisms.

2.1. Roadmaps and experience reuse

Early approaches in experience-based motion planning focused on storing large graphs to accelerate future queries. One of the foundational methods is Probabilistic Roadmaps (PRMs), which constructs a graph of collision-free configurations in the robot's state space for rapid path finding [1]. Building on this idea, Experience Graphs were developed to store entire trajectories rather than individual configurations, allowing for more effective reuse of past solutions [2]. Systems like OMPL Lightning [3] use full motion plans as initial guesses for optimization rather than adapting them directly. Recent research continues to advance this concept; for instance, the work in [4] proposes a self-optimizing replay mechanism that stores and retrieves successful trajectories from an "experience pool" for manipulator motion planning in dynamic environments. These methods provide generalization across diverse queries but incur significant overhead in maintaining and searching large graph structures.

2.2. Template-based and local adaptation methods

Another group of methods aims to identify common motion patterns across tasks and store a set of trajectory templates. In approaches like Experience-Based Planning with Sparse Roadmap Spanners [10], the system maintains a sparse roadmap containing key trajectories. When a new query arrives, the system selects and adapts a suitable template to the new conditions. These approaches reduce memory usage compared to full experience graphs and enable local adaptation. However, their effectiveness relies on careful clustering or segmentation of the task space to create a representative set of templates.

Machine Learning for Trajectory Selection. Recent work investigates the application of machine learning to determine which past trajectories are relevant to new queries. Some approaches use neural embeddings to predict the similarity between the current task and stored trajectories or train classifiers to indicate reusability [5]. These approaches can generalize well from experience but require large datasets for training and add computational costs at the inference stage.

2.3. Direct trajectory caching and retrieval

The concept of caching, storing data for fast access, also finds broad application in related domains. For example, caching is used to optimize UAV trajectories for content delivery [6], to jointly optimize computation and path planning in mobile edge computing networks [7], and to preserve the privacy of vehicle trajectories [8].

Inspired by this general idea, a recent approach known as RT-Cache [9] demonstrates the effectiveness of direct trajectory caching for real-time manipulation tasks. This training-free system retrieves and stitches trajectory segments from a pre-computed cache to generate new motions, showing high performance.

2.4. Simplified caching as an alternative

Compared to the methods reviewed above, proposed approach intentionally focuses on a lightweight mechanism: cache full trajectories and retrieve them using a simple geometric proximity metric. While complex structures like experience graphs [2, 4] and machine learning models [5] offer powerful generalization, and modern systems like RT-Cache [9] demonstrate the effectiveness of direct retrieval, we hypothesize that in high-dimensional task spaces (e.g., 10-DOF robot + environment) with sparse and varied queries, the computational and implementation overhead of complex structures may outweigh their benefits. At the same time this hypothesis merits further experimental investigation.

3. Proposed technique

The proposed technique for path planning in variable environment centers on caching previously successful trajectories and reusing them when similar planning requests arise. Because the bathroom cleaning robot performs very similar tasks from day to day, but the trajectories performed are still not identical, using simple caching of entire trajectories with relatively small adjustments can be effective for this type of robot. This section describes the trajectory cache, the similarity metric used for retrieving candidates, and the adaptation and validation steps.

Trajectory cache structure: the cache is implemented as a collection of trajectories, each consisting of a sequence of key robot states (positions of the mobile base and joint angles of the manipulator). Between key points, the robot expected to move in a straight line in the robot's state space. For each new planning request, the cache is queried for existing trajectories whose start and goal states are similar to those of the current request.

Similarity metric: To identify relevant cached trajectories, we compute the following distances:

- The base distance, denoted as d_{base} is calculated as the sum of the Euclidean distance between the mobile base positions (p_1, p_2) and the absolute angular difference (in radians) between their orientations (θ_1, θ_2), as expressed in the following equation:

$$d_{base} = \|p_2 - p_1\| + |\theta_2 - \theta_1| \quad (1)$$

where $\|\cdot\|$ represents the Euclidean norm;

- The manipulator distance, denoted as d_{arm} is calculated as maximum joint angle difference between corresponding joints of the manipulator In radians) as expressed in following equation:

$$d_{arm} = max\left(|J_2^i - J_1^i|\right) \quad (2)$$

where J_k^i is an angle of rotation of i-th joint of a manipulator in k-th state.

A cached trajectory is considered relevant if both distances are below a fixed threshold (0.2 in our experiments).

There may be the following reasons why the cached trajectory failed the collision check:

- The cached trajectory is slightly offset from the input initial and final states and during trajectory adaptation it received a collision.
- The state of the environment has changed (for example, the angle of the adjacent doors has changed), so the trajectory that previously did not contain collisions now contains them.

Therefore, several different trajectories may be required to navigate between the same pairs of robot states. When multiple cached trajectories match the similarity metrics, they are sorted by the sum of the two distances. Up to three closest trajectories are selected.

Trajectory adaptation: each selected trajectory undergoes the following adaptation:

- The initial and goal states are replaced with the current request's start and goal states.
- If the base is fixed (non-moving) in the current request, all intermediate states in the trajectory are adjusted to match the base pose of the current request.

Trajectory validation: to ensure safety, each adapted trajectory is traced with fine resolution:

- Linear interpolation is performed with a step size of 0.04m (position), 0.04 radians (orientation), and 0.035 radians (joint angles).
- Each interpolated state is checked for collisions.

Only collision-free trajectories are reused. Otherwise, a fallback planning procedure (in this study the RRIS [12] path planning method was used as a fallback planner) is invoked to compute a new trajectory. If a collision-free trajectory is found by fallback planning procedure, it should be added to the cache for future use.

4. Experimental Setup

The experiments were conducted in a simulator developed by Somatic Holdings Ltd. that models a mobile cleaning robot operating in public bathroom environments. An image of the robot in the simulation can be seen in Figure 1.

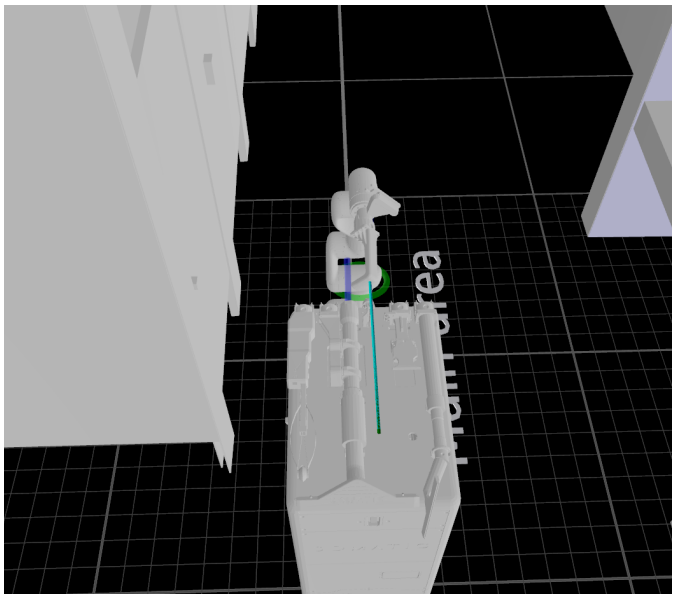

Figure 1: Bathroom cleaning robot in simulated working environment

The main parts of a robot are a movable base with three degrees of freedom and a 7-DOF robotic arm UFACTORY xArm7 [11], mounted on the base. The robot can also take various tools for various tasks.

The robot is tasked with a variety of sub-tasks that require precise manipulation under spatial constraints. These tasks include:

- Opening doors with handles.
- Opening the stall doors.
- Raising and lowering toilet seats.
- Spraying chemicals.
- Spraying water.
- Drying the surfaces.
- Vacuuming liquids.
- Other routine cleaning actions.

Many of these sub-tasks require the planning of arm trajectories that avoid collisions with the environment, which includes doors, walls, stalls, and fixtures. Since some of the actions must be executed in a real time during plan execution, delays in planning can lead to visible pauses in the robot's operation.

The simulation and a real life plan executing introduces variability in both the robot's behavior and the environment state. This non-determinism originates from the stochastic physics engine and the inherently variable execution context. As a result, the robot may encounter slightly different configurations on each execution, even when following the same high-level plan.

The goal was to evaluate whether caching and reusing previously computed trajectories could reduce planning delays in such a dynamic setting. The robot's control system was instrumented to measure the duration of all trajectory planning calls and logged whether the trajectory was retrieved from cache or planned from scratch.

In addition to the simulated cleaning environment, further evaluation using both simulation and a physical robot were conducted. To simulate environmental variability, multiple environment states were generated by changing the angles of stall doors between 0° and 90° in 10° increments. For each angle, the full cleaning plan was run in simulation and planning times were measured with and without caching.

Physical testing was also conducted, and four configurations were evaluated:

- No caching (all paths planned online).
- Caching enabled but initially empty (paths cached during execution).
- Caching with pre-trained trajectories from simulation.
- Second execution with accumulated real-robot cache.

Experiments in a simulation were conducted on a system with a 12th Gen Intel(R) Core(TM) i7-12700H, 32GB of RAM, and a GeForce RTX 3050 Mobile GPU. The exact parameters of the machine installed on the real robot cannot be disclosed due to a non-disclosure agreement, but in general the robot has a worse processor, but a better GPU. And because the trajectory planning was done on the CPU, the average runtime on the robot is slightly longer than in the simulation.

5. Results

The caching system evaluated over multiple simulation runs of the same high-level plan. The baseline (without caching) involved 271 trajectory planning calls, totaling 18.319 seconds of planning time.

In the first run with the caching mechanism, many trajectory requests had no cached equivalent, but some were still reused due to the fact that some trajectories planned within the same plan are close to each other. In the second run, the cache was more populated, resulting in significantly more reuse. The results of the caching system test in simulation can be seen in Table 1.

Table 1
Caching system results in simulation

Simulation run	Number of searches in cache	Number of found and validated trajectories in cache	Number of trajectories planned by back-up planner	Number of trajectories in cache after plan finished	Time of search and validating trajectories in cache, s	Time of planning trajectories by back-up planner, s	Total time of trajectories building, s	Speed-up factor
Without caching	271	-	271	-	-	18.319	18.319	1.0x
First run (with empty cache in the beginning)	271	137	134	134	0.698	14.319	15.619	1.10x
Second run (with filled cache after first run)	272	253	19	143	2.233	3.301	5.534	3.31x

The results presented in Table 1 show a substantial reduction in planning time across repeated executions of similar plans. The second run in simulation demonstrates that over 93% of trajectory requests were fulfilled by the cache. This highlights the system's ability to exploit task repetition and structural similarities in the robot's environment.

In the second test, collision safe trajectory planning with a caching system was tested on a real robot. Firstly a trajectory cache was precalculated in a simulation with different environmental states. Then precalculated cache usage compared to empty cache usage and to planning without a caching system. The results of the experiment can be seen in Table 2.

Table 2
Creating precalculated cache in simulation and testing cache system on real robot

Testing environment	Number of searches in cache	Number of found and validated trajectories in cache	Number of trajectories planned by back-up planner	Number of trajectories in cache after plan finished	Time of search and validating trajectories in cache, s	Time of planning trajectories by back-up planner, s	Total time of trajectories building, s	Speed-up factor**

Simulation, 0° doors	271	137	134	134	0.751	14.031	14.782	1.24x
Simulation, 10° doors	272	228	44	178	2.208	6.246	8.454	2.17x
Simulation, 20° doors	272	238	34	212	2.510	5.674	8.184	2.24x
Simulation, 30° doors	273	249	24	236	2.615	3.891	6.506	2.82x
Simulation, 40° doors	273	255	18	254	2.707	2.950	5.657	3.24x
Simulation, 50° doors	272	257	15	269	2.918	2.718	5.636	3.25x
Simulation, 60° doors	274	262	12	281	3.001	2.225	5.226	3.51x
Simulation, 70° doors	274	264	10	291	3.012	1.786	4.798	3.82x
Simulation, 80° doors	278	268	10	301	3.098	1.802	4.900	3.74x
Simulation, 90° doors	264	262	2	303	2.997	0.545	3.542	5.17x
Real robot, no caching	278	-	278	-	-	27.396*	27.396*	1.0x
Real robot, with empty cache at plan start	279	138	141	141	1.020*	19.875*	20.895*	1.31x
Real robot with cache precalculated in simulation	277	261	16	319	4.115*	3.581*	7.696*	3.56x

Real robot, second run with precalculated cache	278	269	9	328	4.202*	2.090*	6.292*	4.35x

* the time of path planning and a cached trajectory search is a little higher on a real robot than in simulation, because the computer installed on a real robot has a little worse computational abilities as described in section 4.

** the speed-up factors for all simulation runs were calculated relative to the simulation run without caching (results presented in the first row of Table 1), and the speed-up factors for all real robot runs were calculated relative to the real robot run without caching (presented in the fourth row from the bottom of Table 2).

It is worth noting that even during the robot's first simulation run, without a pre-populated cache but with the caching mechanism enabled, the robot reused almost half (134 out of 271) of the trajectories that had to be planned (as can be seen in Table 1 and Table 2). This can be explained by the fact that some of the robot's movements are highly repetitive. Therefore, after the robot planned a trajectory for the first time while executing a task, it was able to reuse it on subsequent occasions, if the new configuration of the surrounding environment allowed it.

It was observed that as the cache filled up over simulation runs in different environments, the number of invocations of the backup planner progressively decreased, eventually reaching a low of two. Correspondingly, the total time for planning and cache lookups fell by a factor of approximately 4.2, from 14.872 seconds during the initial run to 3.542 seconds in the final run. When compared to the run without caching, the overall speed increased by a factor of 5.17. In contrast, the physical robot required the backup planner more frequently, even though its cache was pre-populated with all trajectories generated in the simulation. This discrepancy can be attributed to the simulation being a significantly more simplified and deterministic environment compared to the real world.

On the real robot, pre-training the cache cut planning time from ~27 seconds to ~7.7 seconds. On a second run, the system reused nearly all trajectories, reducing time to 6.3 seconds. These findings demonstrate that trajectory caching generalizes across environment configurations and can accelerate execution even on maps not previously visited by the physical robot.

Caching also contributed to the robot's responsiveness by minimizing pauses due to planning delays. Importantly, the system remained robust to variability in execution, as each reused trajectory was validated for safety before being accepted.

6. Discussion

The results of the study support the practicality of caching entire trajectories for real-time robotic planning. The main advantage lies in the simplicity and speed of this approach, which makes it suitable for highly repetitive and partially structured tasks like restroom cleaning. In contrast, roadmap or ML-based reuse mechanisms introduce additional system complexity, memory usage, and require learning infrastructure.

We emphasize that this is not a criticism of such approaches, they may offer greater generalization in other settings. However, our hypothesis is that in high-dimensional state spaces with sparse plan reuse (e.g., 10-DOF robot in semi-structured map), the cost-benefit ratio may favor simpler caching schemes. This should be validated across domains.

Additionally, the impact of using cached trajectories on their overall quality warrants future evaluation. Since some planners optimize paths based on metrics such as clearance from the nearest obstacle, a reused trajectory may be of significantly lower quality than one specifically

computed by the main planner. After all, a cached trajectory was likely generated for a different environmental state and may, therefore, not be optimal in the new configuration.

To build upon these findings and systematically investigate this trade-off, we have identified several key areas for future work. Among them are the following:

- Adding environment-dependent indexing to the cache (e.g., door states, seat states).
- Combining caching with trajectory repair or smoothing algorithms.
- Incorporating fallback planner failure cases into the reuse logic (e.g., recording failed plans).
- Comparing proposed method with roadmap-based methods on a prepared set of tests.
- Integrating machine learning methods to predict the relevance of cached trajectories depending on the current environment state (e.g., map configuration, obstacle layout).
- Using machine learning to better match cached trajectories even when start and goal configurations are farther from those of the stored ones, potentially expanding the reuse range of the cache.

7. Conclusions

This work presents an effective trajectory caching strategy for robotic manipulators operating in variable but structured environments. By reusing entire trajectories and adapting them to new requests, planning delays were significantly reduced while maintaining safety. Experiments in simulation and on a real robot confirm that pre-training the cache enables the robot to execute plans with minimal latency. Already on the first run of the plan on real robot, the use of caching allowed to reduce the execution time by 3.5 times, and on the second run, the planning time was further reduced, reaching a total acceleration of 4.3 times. Such performance gains are particularly valuable in applications where minimizing setup time and maximizing operational throughput are critical. It is hypothesized that, compared to roadmap-based methods, the proposed technique is more lightweight and better suited for tightly scoped but high-dimensional robotic tasks. This hypothesis requires experimental validation.

Beyond the specific cleaning scenario, the proposed trajectory caching approach may also be applicable to other domains involving repetitive motion planning, such as warehouse logistics, industrial manipulation, or assistive service robotics. These areas share similar challenges of frequent re-planning under partially predictable conditions, where caching entire trajectories could provide comparable benefits in speed and reliability.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

The authors would like to express their deep gratitude to Somatic Holdings LTD, whose codebase, the simulation environment and the infrastructure greatly facilitated the development of the mechanism presented in this paper.

Declaration on Generative AI

During the preparation of this work generative AI tools (Gemini 2.5 Pro, ChatGPT 5) have been used for grammar checks, text polishing. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] Kavraki, L. E., Svestka, P., Latombe, J.-C., Overmars, M. H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation, 12(4), 566–580, 1996. doi: 10.1109/70.508439.

[2] Phillips, M. Experience Graphs: Leveraging Experience in Planning. PhD Dissertation, 2015.

[3] Sucan, I. A., Moll, M., Kavraki, L. E. The Open Motion Planning Library. IEEE Robotics & Automation Magazine, 19(4), 72–82, 2012. doi: 10.1109/MRA.2012.2205651.

[4] Xu, P., Di, C., Lv, J., Zhao, P., Chen, C., Wang, R. Robotic Arm Trajectory Planning in Dynamic Environments Based on Self-Optimizing Replay Mechanism. Sensors, 25, 4681, 2025. doi: 10.3390/s25154681.

[5] Khan, A., Kumar, V., Ribeiro, A. Learning Sample-Efficient Target Reaching for Mobile Robots. arXiv:1803.01846, 2018. doi: 10.48550/ARXIV.1803.01846.

[6] Wang, W., Xu, X., Bilal, M., Khan, M., Xing, Y. UAV-Assisted Content Caching for Human-Centric Consumer Applications in IoV. IEEE Transactions on Consumer Electronics, 70(1), 927–938, 2024. doi: 10.1109/TCE.2023.3349079.

[7] Sun, H., Zhou, Y., Zhang, H., Ale, L., Dai, H., Zhang, N. Joint Optimization of Caching, Computing, and Trajectory Planning in Aerial Mobile Edge Computing Networks: An MADDPG Approach. IEEE Internet of Things Journal, 11(24), 40996–41007, 2024. doi: 10.1109/JIOT.2024.3456846.

[8] Huang, Q., Xu, X., Chen, H., Xie, L. A Vehicle Trajectory Privacy Preservation Method Based on Caching and Dummy Locations in the Internet of Vehicles. Sensors, 22, 4423, 2022. doi: 10.3390/s22124423.

[9] Kwon, O., George, A., Bartsch, A., Farimani, A. B. RT-Cache: Training-Free Retrieval for Real-Time Manipulation. arXiv:2505.09040, 2025. doi: 10.48550/ARXIV.2505.09040.

[10] Coleman, D., Sucan, I. A., Moll, M., Okada, K., Correll, N. Experience-Based Planning with Sparse Roadmap Spanners. arXiv:1410.1950, 2014. doi: 10.48550/ARXIV.1410.1950.

[11] The difference between UFACTORY xArm5, UFACTORY xArm6 and UFACTORY xArm7. UFACTORY. URL: http://help.ufactory.cc/en/articles/4491842-the-difference-between-ufactory-xarm5-ufactory-xarm6-and-ufactory-xarm7

[12] Medvid, A. Y., Yakovyna, V. S. Redundant Robotic Arm Path Planning Using Recursive Random Intermediate State Algorithm. Radio Electronics, Computer Science, Control, 3, 173–181, 2025. doi: 10.15588/1607-3274-2025-3-16.