
White Rose Research Online

eprints@whiterose.ac.uk
https://eprints.whiterose.ac.uk
Universities of Leeds, Sheffield and York

Deposited via The University of York.

White Rose Research Online URL for this paper:
https://eprints.whiterose.ac.uk/id/eprint/234509/

Version: Accepted Version

Proceedings Paper:
Liu, Shuai and Liu, Pengcheng (2021) Robot motion planning benchmarking and optimization through motion planning pipeline. In: The 17th IEEE International Conference on Automation Science and Engineering (CASE 2021). 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), 23-27 Aug 2021 IEEE International Conference on Automation Science and Engineering (CASE). IEEE, FRA, pp. 633-638.

https://doi.org/10.1109/CASE49439.2021.9551646

Reuse
Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown
If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.



Robot motion planning benchmarking and optimization through motion planning pipeline

Shuai Liu, Pengcheng Liu, *Member, IEEE*

***Abstract*—Algorithms have been designed for robot motion planning with various adaptability to different problems. However, how to choose the most suitable planner in a scene has always been a problem worthy of research. This paper aims to find the most suitable motion planner for each query under three different scenes and six different queries. The work lies in optimization of sampling-based motion planning algorithms through Motion Planning Pipeline and Planning Request Adapter. The idea is to use the pre-processing of the planning request adapter, to run OMPL as a pre-processer for the optimized CHOMP or STOMP algorithm, and connect through the motion planning pipeline, to realize the optimization of the motion trajectory. The optimized trajectories are compared with original trajectories through benchmarking. The benchmarking determines the most suitable motion planning algorithm for different scenarios and different queries. Experimental results show that after optimization, the planning time of the algorithm is longer, but the efficiency is significantly improved. In the low-complexity scenes, STOMP optimizes the sampling algorithm very well, improves the trajectory quality greatly, and has a higher success rate. CHOMP also has a good optimization of the sampling algorithm, but it reduces the success rate of the original algorithm. However, in more complex scenes, optimization performance of the two optimization methods may not be as good as the original algorithm. In future work, we need to find better algorithms and better optimization algorithms to tackle with complex scenes.**

I. Introduction

With the development and wide application of robot technology, robots reflect their importance and superiority in production and life. Robots have become an effective tool and experimental platform for studying complex intelligent behaviors and exploring human thinking patterns. The robotic arm is one of the earliest robots used in real life and social production. It is composed of many links (metal rods) and joints (electric axles). For example, a 7-DOF Panda arm is shown in Fig. 1. The robotic arm realizes any three-dimensional position and orientation within its range of motion according to the linkage and joint. The robotic arms has been assisting or streamlining operations in certain harsh or harmful environments. In practical, the working environment of the robotic arm is very complicated, and there may be many obstacles exist, so the motion planning of the robotic arm is extremely important [1]. The basic task of motion planning can be described as: the movement from the starting state to the target state [2]–[6]. It must satisfy the external constraints of avoiding obstacles in the configuration space and the internal constraints of the robot's speed and acceleration in terms of machinery, sensing. In simple terms, the use of computer and software algorithm technology to determine the optimal path of the robot arm from the starting state to the target state, while ensuring that it avoids obstacles and meets its own motion performance.

(a) (b)

Fig. 1. The robotic arm assists people in their work: (a) Franka Emika Panda [7]; (b) PR2 Robot [8].

The sampling-based motion planning algorithms are proposed in recent decades and have attracted great attention. In a nutshell, they generally connect a series of randomly sampled points from an unobstructed space, trying to establish a path from the initial state to the target state. These algorithms are originated with the RPP (Randomized Potential Planner) algorithm proposed by Barraquand and Latombe [9]. In 1994, the PRM (Probabilistic Road Maps) algorithm [10] and the RRT (Rapidly-exploring Random Tree) algorithm [11], setting off a new wave of research on robot motion planning. Currently, PRM and RRT belong to the Open Motion Planning Library (OMPL) [12]. OMPL is a C++ open-source library based on sampling/random motion planning algorithms. It contains many prevailing algorithms for motion planning, of which the most famous are PRM and RRT. Although optimization motion planning is mentioned in OMPL, OMPL is still a sampling planning algorithm library.

Optimization-based methods recently become popular for motion planning. The most famous of these are Covariant Hamiltonian optimization for motion planning (CHOMP) [13] and Stochastic Trajectory Optimization for Motion Planning (STOMP) [14]. CHOMP uses gradient-based optimization, while STOMP uses stochastic gradient-free optimization. CHOMP is a novel gradient-based trajectory optimization program and used for motion planning, which makes many daily motion planning problems simple and trainable [13]. Although most high-dimensional motion planners divide trajectory generation into different planning and optimization stages, this algorithm uses covariant gradient and functional gradient methods to perform the optimization stage to design a motion planning algorithm based entirely on trajectory optimization. STOMP is a motion planner based on probability optimization [14]. STOMP produces a collision-free and

S. Liu and P. Liu are with the Department of Computer Science, University of York, York YO10 5GH, United Kingdom (e-mail: pengcheng.liu@ york.ac.uk).

smooth trajectory that satisfies the constraints within a certain period of time. The algorithm does not require gradients and can also produce a better trajectory. STOMP can solve motion planning queries very well and quickly when the scene is not overly complicated.

The sampling-based planning algorithm is the most common probabilistic complete algorithm and is widely used in robotic platforms with many degrees of freedom. Among such algorithms, many variant algorithms have been proposed in recent years, but there is no clear method to prove which type of problem is solved by which algorithm is the most reasonable. For different motion planners, we usually judge the performance of different motion planners through benchmarking. In this paper, we aim to optimize the sampling algorithm and benchmarking, compare the optimized trajectory metrics with the original trajectory metrics through benchmarking, in order to find a relatively good algorithm for different scenes. Future researchers can get some help through this article. We will use the optimization-based algorithm CHOMP or STOMP to modify the initial trajectory formed by the motion planning algorithm in OMPL through the concept of Planning Adapter and Planning Pipeline. And use their different queries on different scenes to perform benchmarking respectively, and then research and analysis on the obtained data. In this way, we want to optimize some of the existing algorithms and analyze their different metrics in different problem scenes through benchmarking, so as to find a relatively good algorithm for different planning problems. We speculate that the trajectory generated by the optimization algorithm is the shortest path, and its quality should be better than the initial trajectory, and the optimization does not affect the success rate. In different scenarios, it has a good optimization performance and stability. Our contribution is linking STOMP/CHOMP and OMPL by using the motion planning pipeline and using STOMP/CHOMP as the post-processing of OMPL through the planning request adapter. We provide relevant information for benchmarking of the implemented optimization algorithm to facilitate users to choose a specific planner.

II. Methodology

In this paper, we propose a framework of planning adapters to combine sampling-based planning algorithms (PRM [15], RRT, LazyPRM [16] and RRTConnect [17] in OMPL) with optimization-based algorithms (CHOMP and STOMP), and these algorithms can be used in a pipeline to produce robust motion plans. For example, the pipeline of a motion plan may be an initial plan established by OMPL, and then optimized through STOMP to generate a better path. In this section, we will introduce the methods used in this article, different motion planners and their corresponding libraries.

A. Sampling-based motion planning algorithm library-OPML

Among the sampling-based motion planning algorithms, the RRT algorithm, PRM algorithm, LazyPRM algorithm and RRT-Connect algorithm in OMPL are chosen. Both RRT and PRM are the most classic and basic algorithms in motion planning, so optimizing them may have obvious effects, which is beneficial to the experiment. Since it is uncertain whether the optimization algorithm is better for the basic algorithm or the improved algorithm, so we have also analyzed and researched the two algorithms of LazyPRM and RRTConnect. RRTConnect is one of the most efficiently improved algorithms. From above, it is noted that the RRTConnect algorithm compensates for the low growth efficiency of the single tree of the RRT species and the problem of tree node redundancy through the use of bidirectional-tree opposing growth strategies and Connect operations. However, some problems in RRT still exist, and RRTConnect still has these shortcomings in algorithm performance: (1) The problem of lack of directionality in tree expansion still exists; (2) As tree nodes increase, its search area is gradually reduced. The probability that the location point is selected as the sampling point also gradually decreases, that is, the guiding effect of the search area on the tree is weakened, and it is easy to cause the tree expansion to stagnate. Therefore, the RRT-Connect algorithm has much room for improvement.

Besides, RRT algorithms still have many shortcomings in the path search: (1) It is only a single tree growing from the initial point to the target point, and the algorithm efficiency is low; (2) Using uniform random sampling, although avoiding the algorithm from falling into the local optimum, the path obtained is not optimized enough due to the lack of directionality in the expansion of the random tree; (3) All points that meet the collision constraint are added to the tree as q_new, but some of the q_new may not be far apart, which is not very helpful to generate a new path, resulting in redundancy of nodes on the random number and making the tree too large and too much nodes, and cause the efficiency of the SelectNearestNeighbor function to drop significantly; (4) Although the algorithm can find a path to the target node, it takes a long time for collision constraint detection and node connection caused by tree node redundancy, and the algorithm has extremely poor performance in the face of real-time planning problems.

A. Motion planning algorithm based on trajectory optimization

1) CHOMP

CHOMP is an algorithm to iteratively improve the quality of the initial trajectory through functional gradient technology and optimize the smoothness and obstacle avoidance. It makes many motion planning problems simple and trainable. This approach can be used in high-dimensional space motion planning, while avoiding obstacles and generating smoother motion planning. It can solve motion planning queries in different scene and locally optimize feasible trajectories to improve trajectory quality, which is a simple variational strategy for achieving good trajectories.

The algorithm is mainly based on two cores: (1) gradient information is often available and can be computed inexpensively; (2) trajectory optimization should be invariant to parametrization [18]. On the basis of these two principles, a motion planning algorithm is designed to generate high-quality trajectories for complex robot systems with multiple degrees of freedom, which can smoothly eliminate unnecessary motions and avoid collisions. While most high-dimensional motion planners divide trajectory generation into different planning and optimization stages, the algorithm uses the covariant gradient and functional gradient methods in the

optimization stage to design a motion planning algorithm based entirely on trajectory optimization. CHOMP can quickly react to the surrounding environment and pull out the infeasible trajectory from the collision, while optimizing dynamics such as joint speed and acceleration. It can quickly converge to a smooth collision-free trajectory and can be executed efficiently on the robot.

Although CHOMP can avoid obstacles in most cases. But if the trajectory is incorrectly guessed in the initial state, the algorithm will fall into a local minimum. Due to its gradient-based nature, CHOMP usually fails to find a solution or returns to a sub-optimal solution after falling into a local minimum, resulting in planning failure. OMPL can effectively alleviate this problem. In the experimental part, we will try to use the trajectory generated by OMPL as the initial trajectory of CHOMP and implement CHOMP optimization on this trajectory through the Motion Planning Pipeline.

2) STOMP

STOMP uses a stochastic trajectory optimization framework, which is an optimization-based motion planner. It will generate smooth and conflict-free motion planning trajectories in a short time. This method relies on noise trajectories to explore the space around a possible but not necessarily feasible initial trajectory, and then combines these trajectories into a lower cost updated trajectory. The cost function includes general cost, control cost, obstacle cost and smoothing cost. STOMP mainly optimizes the combination of obstacle cost and smoothing cost. In each iteration of random trajectory optimization, a series of randomly optimized trajectories with noise trajectories will be generated. The newly generated trajectory is used for simulation to determine its cost, and the candidate solution is updated by its cost. In the whole process, this method does not need to use gradient information, and it can get a certain optimization through general constraints and additional non-smooth costs. This method can directly generate high-quality trajectories or optimize the formed trajectories.

This method also uses a cost function similar to CHOMP, but compared with CHOMP, its optimization method can handle general cost functions without gradients. As mentioned above, due to gradient-based reasons, CHOMP may fall into a local minimum, and STOMP can overcome this problem through its randomness and get better optimization. OMPL is very suitable as the initial trajectory of CHOMP for optimization, STOMP and CHOMP have a similar cost function, so we reasonably speculate that OMPL as the initial trajectory of STOMP will also have a good optimization effect which also has been approved in our experiment.

B. Motion Planning Pipeline

In Moveit!, use the motion planner to plan the unknown environment with obstacles. In the experimental part, we will introduce in detail the detailed process of Moveit! for robot motion planning. In this part, we will talk about Motion Planning Pipeline of Moveit!. In some scenarios, some per-processing motion plan requests or post-processing motion plan requests are needed to process the motion planning. In this case, it can help us connect per-processing, motion planner, and post-processing in pipeline. The complete motion planning pipeline includes Motion Planner and Plan Request Adapters, as shown in Fig. 2.

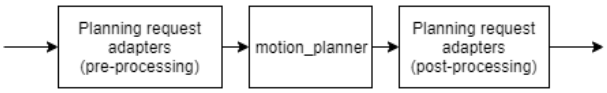

Fig. 2. Motion planning pipeline.

As shown in Fig. 2, motion Planner contains the OPML, CHOMP, STOMP and other motion planning algorithms mentioned earlier. Planning request adapters (pre-processing) include: FixStartState-Bounds, FixWorkspaceBounds, FixStartStateCollision, Fix-StartStatePathConstraints, AddTimeParameter-ization, CHOMPOptimizerAdapter and STOMPSmoothing-Adapter, etc. Their functions are: FixStartStateBounds is used to repair the initial limit of joint; FixWorkspaceBounds sets the default size of the workspace; FixStartStateCollision is responsible for repairing the collision configuration file; FixStartSta-tePathConstraints is responsible for finding the posture that satisfies the constraints as the initial state of the robot; Add-TimeParameterization can perform speed and acceleration constraints for the space trajectory, and add speed, acceleration, time and other parameters to each trajectory point; CHOMPOptimizerAdapter: Use CHOMP algorithm for trajectory optimization; STOMPSmoothingAdapter: Use STOMP algorithm for random trajectory optimization. The planning pipeline implements the following three functions: 1) Automatically instantiate a planner plug-in; 2) Automatically instantiate a set of planning request adapters, allowing per-processing of planning requests or post-processing of planning; 3) Combining the planner with the plan requests that the adapters are grouped together in a serialized manner.

C. Motion planning Benchmarking

The benchmarking package provided by Moveit! is used to perform benchmarking on motion planning algorithms and aggregate/draw statistics in combination with OMPL Planner Arena. The metrics of the current motion planning algorithm used by the robotic arm through the box plot. The benchmarking database are uploaded to OMPL Planner Arena to display interactive results. These results can help us compare different motion planners in the same environment, and also help us compare the same algorithms in different environments, so as to determine the most suitable motion planning algorithm for specific motion planning problem.

III. Experiments

A. Experimental Environment

All the experiments in this paper are on VMware virtual machines with 16G RAM and quad-core Intel i7-9750H CPU@ 2.60GHz. This virtual machine has 6G RAM, 40G storage space and the system Ubuntu 16.04. Because ROS and Moveit! involved in this experiment need to be used based on Ubuntu, and currently on Ubuntu 16.04, ROS and Moveit! support more complete versions.

A. Motion Planning Benchmarking

1) Robot for Benchmarking

In this experiment, the robot used is Franka Panda. It is a collaborative robot arm, developed by FRANKA EMIKA. Panda has 7 DOF with torque sensors in all seven axes, the arm can delicately manipulate objects, and accomplish complex tasks. And each joint of Panda is equipped with strain gauges,

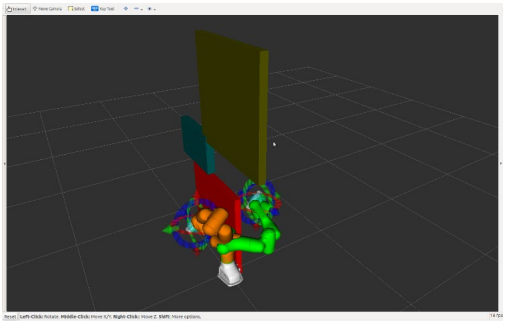

Fig. 3. Narrow tunnel.

which can detect minor collisions to avoid injury to people, and is suitable for high-precision 3C industries, such as assembly and inspection.

2) Benchmarking Scenarios

In this experiment, we mainly designed 3 different benchmarking scenarios:

Narrow Tunnel Scene: The first experimental environment is a narrow tunnel as shown in Fig. 3. It is a classic scene in the motion planning benchmarking. Due to the narrow passage and limited space, many classic motion planning algorithms are not necessarily applicable. In this experiment, we implemented pipelines on the traditional sampling-based motion planning algorithm and the optimization-based algorithm. The narrow tunnel is a very good obstacle avoidance environment for testing the optimized algorithm. This scene has only one query, that is, the start state (green) and goal state (orange) of the robotic arm are at both ends of the channel. There are two ways for the robot arm to complete the motion planning, one is to reach the other side of the board through a narrow tunnel; Second, it doesn't pass the tunnel, it takes a long path time for the arm to bypass the board. In this experimental scene, it is possible to judge whether the algorithm used passes through a narrow tunnel and completes the motion planning by running time, thereby judging the motion planning algorithm whether found the shortest way.

Industrial Workshop Scenes: The second experimental scene is an industrial workshop as shown in Fig. 4 and Fig. 5. In industrial workshops, workers need to assemble many different parts every day. They pick the parts they need from the box containing the parts and place them on the workbench for assembly. If the process of selecting and searching can be completed by a robotic arm, it will save time and the human operator can concentrate on assembling, reducing the workload and improving work efficiency. In this scene, two different queries are mainly involved: (1) the robotic arm pick up parts from the box on the second floor of the parts table and brings them to the workbench. The complexity of the benchmarking scene is low. As long as the robot arm does not touch the obstacles near the workbench, the motion planning can successfully make the robot arm from the second layer of the parts table to the workbench. (2) The robotic arm selects parts from the first layer of the parts table and brings the parts to the workbench. Although the query looks very similar to the first query, it is a little complex than the first, because it is necessary to avoid not only obstacles on the workbench but also obstacles on the parts table.

Library Scene: The last scene is the library environment as

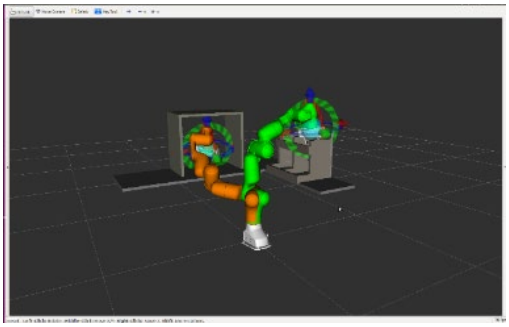

Fig. 4. The query a of industrial workshop.

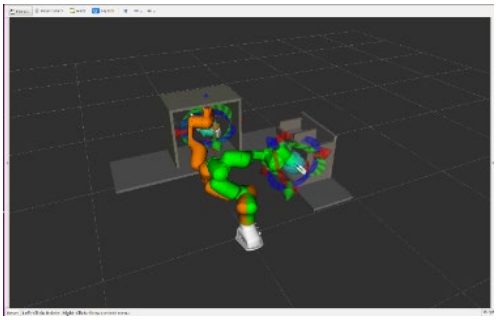

Fig. 5. The query b of industrial workshop.

shown in Figs. 6-8. The workload of the library is very large, and the staff needs to organize the books every day. This includes sorting out the books returned by readers, sorting them, and putting them back on the shelves. This is a heavy workload, especially in libraries with a lot of people. If we use robotic arms to classify and place different types of books. Then it will reduce people's workload to a large extent and improve work efficiency. In this scene, three different queries are set: (1) The robotic arm moves the arm from under the table to the top of the table. This query is mainly designed for when the book is accidentally dropped on the ground, and how putting the books back on the desktop after they dropped on the ground when they are collected. The motion planning must avoid collisions between the robotic arm and the desk and the bookshelf. The scene is relatively low in complexity, with fewer obstacles and not dense. (2) The robotic arm moves from the table to the bookshelf. This query is mainly for collecting and sorting the books and putting them back to the corresponding position on the bookshelf. The scene is highly complex, and it is necessary to avoid collisions with bookshelf spacers and tables. (3) The robotic arm moves from the third level of the bookshelf to the second level of the bookshelf. This query is mainly reflected in the operations needed to organize the bookshelf. The scene is highly complex and needs to move from two narrow spaces. The above three experimental scenes are good for a reasonable evaluation of the motion planning algorithm.

Multiple sets of data against different planning metrics are obtained. We need to analyze these data and find a better motion planning algorithm in the scenario. Besides, the above three scenarios only include the motion planning of the robot arm without the gripper, so the pick and place of the items are not implemented. In future work, it can be integrated.

1) Benchmarking Metrics

This experiment measures the performance, reliability and quality of the paths generated by all planners via the following

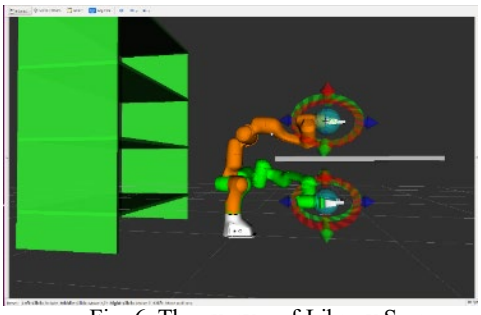

Fig. 6. The query a of Library Scene.

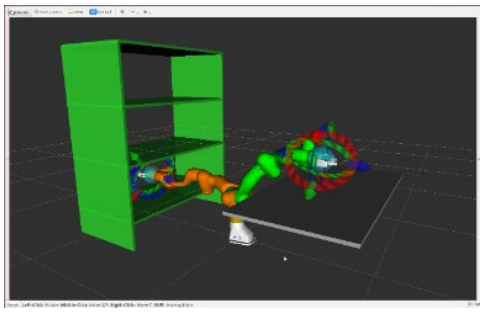

Fig. 7. The query b of Library Scene.

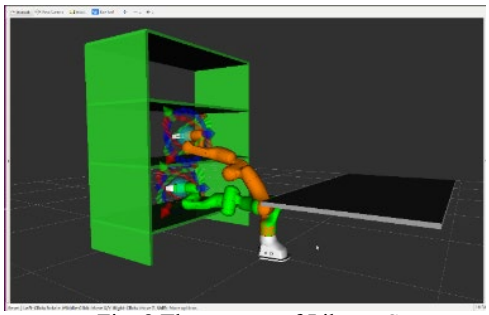

Fig. 8 The query c of Library Scene.

metrics: (1) Solve (%): it indicates the percentage of motion planners finding solutions to the current query in the current scene. The higher the value, the higher the probability that the planner finds a solution, that is, the better the performance of the motion planner. (2) Path plan time (s): It is the path running time planned by the motion planner. This value is another way to test the length of the path. The shorter the running time, the shorter the path planned by the motion planner, and the better the quality of the path generated by the motion planner. (3) Path smoothness: It represents whether the trajectory generated by the motion planner is smooth. If the trajectory is smooth, the smaller the metric, the better the performance and reliability of the motion planner. This value is calculated from three consecutive path points and the angle formed between them. The closer the value is to 0, the more the trajectory tends to be a straight line. (4) Path clearance (m): It represents the average value from the path point on the trajectory generated by the motion planner to the nearest obstacle or invalid state value. If the value is larger, it proves that the motion planning path is far away from obstacles and invalid states, and it also indicates that the quality and reliability of the path generated by the motion planner are higher. (5) Total time(s): It refers to the time it takes for the entire motion planning algorithm to find a feasible solution. It includes plan time, interpolation time, simplify time and process time. The smaller the metric, the better the performance of the motion planner.

2) The Process of Benchmarking

The experiment contains scene design, motion planner selection, planning adapter setting and benchmarking. In each scene, benchmarking the different queries are required, and each query is described and analyzed. The benchmarking performed 50 times planning for each motion planning algorithm and collected its data. If the single running time exceeds 10.0s, the motion planning will be stopped, and the planning failure will be marked. First, we configure the motion planner parameters and benchmarking parameters, and then run the benchmarking to benchmark the currently configured scene and motion planner. Get the corresponding database. Then we uploaded the database through the OMPL Planner Arena, the visual data results shown by box plots are obtained.

B. Results

There may be outliers in the obtained data set, so we have no way to evaluate this set of data in the form of an average. The median better interprets the average level of the data set with outliers, so the median is used in this experiment to better describe the data information of each data set. All the original results of the experiment are submitted in supplementary files.

1) Narrow Space (Tunnel)

TABLE I. TUNNEL BENCHMARKING RESULT

Planner	Solve (%)	Path time (s)	Path smooth	Path clear-ance	Total time(s)
PRM	100	0.0600	0.228	0.0412	0.061
RRT	100	0.0438	0.198	0.0351	0.045
LazyPRM	100	0.0465	0.201	0.0412	0.048
RRTConnect	100	0.0372	0.208	0.0362	0.041
CHOMP	0	-	-	-	-
STOMP	100	0.2063	0.013	0.0343	0.211
PRM-CHOMP	84	0.1462	0.075	0.0365	0.380
RRT-CHOMP	74	0.1300	0.055	0.0360	0.389
LazyPRM-CHOMP	78	0.1320	0.083	0.0395	0.377
RRTConnect-CHOMP	78	0.1310	0.048	0.0340	0.372
PRM-STOMP	100	0.1410	0.036	0.0410	0.168
RRT-STOMP	100	0.1210	0.028	0.0371	0.144
LazyPRM-STOMP	100	0.1250	0.030	0.0374	0.150
RRTConnect-STOMP	100	0.1120	0.037	0.0388	0.140

Table I shows that, in this scenario, the resolution rate of the original algorithm is almost 100%, and only the CHOMP algorithm fails. In the original algorithm, the path time of RRT, LazyPRM and RRTConnect is relatively short. After optimizing through the CHOMP algorithm, the success rate of the algorithm is reduced. Our reasonable guess may be that the CHOMP algorithm is trapped in a local minimum so the motion planning fails. Trajectory time can generally represent the length of the planned path. It can be seen from the trajectory time that motion planning algorithms based on

sampling can find a shorter path through the tunnel, while the STOMP algorithm cannot find a shorter path through the tunnel, so the trajectory time is longer. Secondly, it can be seen that the time of the trajectory generated based on the optimized sampling algorithm is greater than the original time.

Regarding the smoothness, the optimized sampling algorithm has been significantly improved. Although there is a gap with the smoothness of STOMP, it has provided a fairly good motion planning trajectory. Since this scene is tested for the narrow space, the clearance of this set of data is basically unchanged. In terms of total time, the sampling algorithm based on optimization takes longer than the original sampling algorithm, but it is worth noting that the sampling algorithm based on optimization takes less time than the STOMP algorithm.

It can be seen from the above that in a narrow space scene, after CHOMP optimization, the success rate of the algorithm will decrease, but the trajectory quality is better than the original trajectory quality. The sampling-based algorithm optimized by STOMP has a high success rate, and the quality of the trajectory is greatly improved compared with the trajectory generated by the initial algorithm, and it also improves that the STOMP algorithm cannot find a shorter planned path through the tunnel. In the tunnel scene, we recommend to use any sampling algorithm and optimize the STOMP trajectory.

2) Industrial

a. The first floor parts box to table query:

TABLE II. INDUSTRIAL_FTT BENCHMARKING RESULT

Planner	Solve (%)	Path time(s)	Path smooth	Path clear-ance	Total time(s)
PRM	100	0.87	0.29	0.046	0.89
RRT	26	0.98	0.32	0.051	10.0
LazyPRM	46	0.60	0.30	0.070	10.0
RRTConnect	100	0.064	0.34	0.044	0.068
CHOMP	0	-	-	-	-
STOMP	94	0.17	0.051	0.163	0.19
PRM-CHOMP	64	0.99	0.050	0.046	1.70
RRT-CHOMP	16	0.26	0.070	0.053	10.0
LazyPRM-CHOMP	32	0.54	0.050	0.049	10.0
RRTConnect-CHOMP	64	0.175	0.070	0.045	0.47
PRM-STOMP	92	0.74	0.083	0.158	0.83
RRT-STOMP	28	0.70	0.077	0.128	10.0
LazyPRM-STOMP	42	0.98	0.072	0.140	10.0
RRTConnect-STOMP	96	0.12	0.075	0.160	0.16

In this query, the original algorithm PRM, RRTConnect and STOMP all have a good problem solving rate. The RRTConnect algorithm takes extremely short time and can generate the shortest trajectory, but its trajectory quality is not very good. After the optimization of the CHOMP algorithm, the success rate of the sampling algorithm is greatly reduced, and the smoothness of the trajectory is greatly improved. After the basic sampling algorithm is optimized by the STOMP algorithm, the success rate is almost unchanged, but the quality of the generated trajectory has been significantly improved. Under the query of this scenario, the optimal solution algorithm is the RRTConnect-STOMP algorithm, which can generate a shorter trajectory with better quality in a relatively short time.

b. The second floor parts box to table query:

TABLE III. INDUSTRIAL_STT BENCHMARKING RESULT

Planner	Solve (%)	Path time(s)	Path smooth	Path clear-ance	Total time(s)
PRM	100	0.48	0.28	0.067	0.49
RRT	18	1.48	0.29	0.080	10.0
LazyPRM	68	0.11	0.27	0.072	1.0
RRTConnect	100	0.041	0.26	0.062	0.046
CHOMP	0	-	-	-	-
STOMP	98	0.06	0.032	0.182	0.067
PRM-CHOMP	44	0.36	0.076	0.080	1.15
RRT-CHOMP	16	0.21	0.122	0.076	10.0
LazyPRM-CHOMP	36	0.17	0.148	0.079	1.14
RRTConnect-CHOMP	58	0.146	0.080	0.077	0.42
PRM-STOMP	90	0.45	0.080	0.164	0.63
RRT-STOMP	36	0.95	0.071	0.153	10.0
LazyPRM-STOMP	64	0.25	0.076	0.162	0.62
RRTConnect-STOMP	96	0.11	0.066	0.161	0.14

In this query, the problem solving rate of RRT, PRM and STOMP in the original algorithm is still the highest. The success rate of the algorithm obtained after CHOMP optimization has been reduced a lot, resulting in unstable algorithm. However, after the optimization of the STOMP algorithm, although its success rate is slightly reduced, it is always within an acceptable range. The trajectory generated by the STOMP algorithm is the best trajectory among all benchmarking algorithms. Not only is the generated trajectory the shortest path, but it also has a fairly good quality. Although the sampling-based algorithm also has a significant improvement after optimization, the STOMP algorithm solution is more excellent under the query of this scene, so STOMP is the best solution algorithm under the query of this scene.

3) Library

a. Under table to on table query:

TABLE IV. LIBRARY_DTU BENCHMARKING RESULT

Planner	Solve (%)	Path time(s)	Path smooth	Path clearance	Total time(s)
PRM	100	0.043	0.48	0.093	0.046
RRT	100	0.029	0.44	0.064	0.032
LazyPRM	100	0.026	0.36	0.071	0.028
RRTConnect	100	0.019	0.41	0.081	0.022
CHOMP	0	-	-	-	-
STOMP	100	0.115	0.05	0.186	0.119
PRM-CHOMP	80	0.129	0.18	0.088	0.45
RRT-CHOMP	62	0.125	0.12	0.080	0.47

LazyPRM-CHOMP	70	0.121	0.14	0.088	0.42
RRTConnect-CHOMP	80	0.115	0.16	0.075	0.41
PRM-STOMP	100	0.078	0.058	0.143	0.082
RRT-STOMP	100	0.065	0.063	0.153	0.07
LazyPRM-STOMP	100	0.069	0.052	0.139	0.075
RRTConnect-STOMP	100	0.053	0.072	0.149	0.058

Under the query in the library scene, we can see from the table that all the original algorithms except CHOMP can complete the task satisfactorily. This is because the complexity of the query is relatively low. The quality of the trajectory produced based on the sampling algorithm is not as good as the STOMP algorithm. However, the length of the trajectory generated by the STOMP algorithm is longer, which is much longer than the trajectory generated by the sampling algorithm. After the optimization of the CHOMP algorithm based on the sampling algorithm, its success rate has decreased, requiring a long path time, and the generated trajectory need a long process time, that is not the shortest path, but its trajectory quality has been improved. After the sampling algorithm is optimized by the STOMP algorithm, its success rate is still 100%. The path time of the generated path is shorter than that of the original STOMP algorithm, indicating that the path is shorter. The path time of the generated path is shorter than that of the original STOMP algorithm, indicating that the path is shorter, and the quality of the path has also been greatly improved. Therefore, in this query, we recommend using a sampling algorithm optimized by the STOMP algorithm.

b. Table to bookshelf query:

TABLE V. LIBRARY_TTB BENCHMARKING RESULT

Planner	Solve (%)	Path time(s)	Path smooth	Path clearance	Total time(s)
PRM	100	0.71	0.098	0.094	0.85
RRT	26	0.32	0.076	0.118	10.0
LazyPRM	58	0.50	0.098	0.101	2.51
RRTConnect	100	0.044	0.158	0.073	0.049
CHOMP	0	-	-	-	-
STOMP	54	0.25	0.043	0.171	0.71
PRM-CHOMP	70	0.78	0.047	0.098	1.25
RRT-CHOMP	10	0.29	0.049	0.127	10.0
LazyPRM-CHOMP	44	0.48	0.046	0.104	0.65
RRTConnect-CHOMP	68	0.14	0.082	0.071	0.45
PRM-STOMP	78	1.25	0.037	0.149	1.67
RRT-STOMP	16	0.28	0.021	0.134	10.0
LazyPRM-STOMP	50	0.32	0.028	0.140	3.0
RRTConnect-STOMP	68	0.33	0.050	0.130	0.65

In this query, RRTConnect has the best success rate, the shortest trajectory, and the most time-saving among the basic algorithms, but its trajectory quality is very poor and is not recommended. If the shortest path is considered, it is recommended to use the original RRTConnect algorithm, and the trajectory produced is the shortest path. If the path quality is considered, PRM-STOMP is most recommended in this query. Its success rate is only 78% and the trajectory quality is high, but its long trajectory is not the shortest path. If considering the overall situation, it is recommended to use the RRT-STOMP algorithm, all metrics of which are normal and acceptable values, and the resulting trajectory is also excellent. In the data of the original algorithm, it can be seen that the usually stable STOMP algorithm has also been affected by the complexity of the scene, and the success rate is only 54%. This also leads to a decrease in the optimization performance of the STOMP algorithm, which reduces the success rate of the sampling algorithm after STOMP optimization.

c. Move book from Third floor to Second floor on the bookshelf query:

TABLE VI. LIBRARY_B3TB2 BENCHMARKING RESULT

Planner	Solve (%)	Path time(s)	Path smooth	Path clear-ance	Total time(s)
PRM	92	1.2	0.32	0.38	1.24
RRT	66	0.1	0.30	0.36	1.3
LazyPRM	60	0.75	0.40	0.41	5.0
RRTConnect	100	0.038	0.31	0.47	0.041
CHOMP	0	-	-	-	-
STOMP	32	0.68	0.11	0.178	1.6
PRM-CHOMP	50	0.77	0.19	0.042	1.35
RRT-CHOMP	48	0.61	0.135	0.036	1.35
LazyPRM-CHOMP	16	0.2	0.275	0.132	10.0
RRTConnect-CHOMP	46	0.132	0.125	0.037	0.98
PRM-STOMP	40	2.0	0.07	0.168	3.80
RRT-STOMP	30	1.4	0.08	0.145	3.81
LazyPRM-STOMP	30	1.2	0.07	0.151	10.0
RRTConnect-STOMP	38	0.75	0.07	0.137	2.60

The complexity of the query is very high, so that the original algorithm has the problem of planning failure. Only the PRM and RRTConnect algorithms have a higher success rate. STOMP has also been greatly affected, and its success rate is only 32%, which also leads to a greatly reduced success rate of the optimized algorithm. Through the comparison of the data in the table, the most suitable motion planning algorithm used in this scenario is the RRTConnect.

IV. CONCLUSION

In this paper, we have studied the robot motion planning benchmarking and optimization based on motion planning pipeline. Six queries in three different scenes have been explored and the most suitable algorithm has been proposed for the query. The planning algorithms studied include PRM, RRT, LazyPRM, RRTConnect in OMPL, CHOMP, STOMP, and the new algorithm obtained by OMPL with CHOMP and STOMP in pipeline, respectively. Through experimental results, it is found that in most scenes, the quality of the trajectory generated by the sampling algorithm after STOMP optimization is better than the quality of the trajectory generated after CHOMP optimization, and the success rate is higher, the time to find the better path is short and it is easy to find the better path. When executing queries with lower complexity in many scenes, sampling-based algorithms optimized by STOMP are the best solutions, such as

RRTConnect-STOMP, PRM-STOMP, etc. But in more complex scene, STOMP algorithm has received a serious impact. This situation leads to a decrease in the success rate of its trajectory generation, and the optimization effect is greatly reduced or even worse than the original sampling algorithm. In the high-complexity scene, only the best sampling-based algorithm RRTConnect can successfully generate a shorter motion trajectory.

The trajectory generated by the sampling algorithm is also optimized through the planning adapter and use the planning pipeline which chains one motion planner of OMPL with STOMP/CHOMP. Through experiments, we found that in general scenes, both STOMP and CHOMP are very effective in optimizing the algorithm, and STOMP has a better effect. But in more complex scenes, the optimization effect is not as good as the original algorithm.

In future work, we will consider optimizing the parameters based on the sampling planner in OMPL to study whether the optimized trajectory effect has some improvements. This experiment only uses some of the more common sampling-based optimization, we have not studied all sampling-based algorithms. Also, attempts based on sampling algorithms can be broadened, and more sampling-based planning algorithms and even improved sampling-based motion planning algorithms can be studied and benchmarking. The same is true based on optimization planning algorithms. Since only CHOMP and STOMP were tested, other optimization algorithms were not tested. Besides, we will try other prevailing optimization algorithms to optimize the sampling-based planning algorithm. CHOMP and STOMP are good optimization algorithms, they can also form better planning trajectories independently. In the future, we can also try to use optimization-based algorithms as pre-processing for other algorithms. Regarding motion planning pipeline, there is only one pre-processing or one post-processing in this paper. In the future, multiple planning adapters before and after the motion planning algorithm can be approached, using a variety of different motion planning algorithms to optimize the existing sub-trajectories. For example, after the RRT algorithm, perform STOMP optimization and CHOMP optimization. In this experiment, we only analyze and compare the basic indicators in motion planning, while different indicators can be applied in the future work. For more cluttered scenarios, the applicability and adaptability of the motion planning pipeline and optimization will need to be explored further.

REFERENCES

[1] R. B. Shyam, P. Lightbody, G. Das, P. Liu, S. Gomez-Gonzalez, and G. Neumann, "Improving Local Trajectory Optimisation using Probabilistic Movement Primitives," 2019.

[2] P. Liu, H. Yu, and S. Cang, "Geometric analysis-based trajectory planning and control for underactuated capsule systems with viscoelastic property," *Trans. Inst. Meas. Control*, vol. 40, no. 7, pp. 2416–2427, 2018.

[3] M. N. Huda, P. Liu, C. Saha, and H. Yu, "Modelling and Motion Analysis of a Pill-Sized Hybrid Capsule Robot," *J. Intell. Robot. Syst.*, Mar. 2020, doi: 10.1007/s10846-020-01167-3.

[4] P. Liu, H. Yu, and S. Cang, "Optimized adaptive tracking control for an underactuated vibro-driven capsule system," *Nonlinear Dyn.*, vol. 94, no. 3, pp. 1803–1817, 2018.

[5] P. Liu, H. Yu, and S. Cang, "Trajectory synthesis and optimization of an underactuated microrobotic system with dynamic constraints and couplings," *Int. J. Control Autom. Syst.*, vol. 16, no. 5, pp. 2373–2383, 2018.

[6] P. Liu, H. Yu, and S. Cang, "Adaptive neural network tracking control for underactuated systems with matched and mismatched disturbances," *Nonlinear Dyn.*, vol. 98, no. 2, pp. 1447–1464, Oct. 2019, doi: 10.1007/s11071-019-05170-8.

[7] "Overview — Franka Control Interface (FCI) documentation." https://frankaemika.github.io/docs/overview.html (accessed Oct. 19, 2020).

[8] "Robots/PR2 - ROS Wiki." http://wiki.ros.org/Robots/PR2 (accessed Oct. 24, 2020).

[9] J. Barraquand and J.-C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proceedings., IEEE International Conference on Robotics and Automation*, 1990, pp. 1712–1717.

[10] L. Kavraki and J.-C. Latombe, "Randomized preprocessing of configuration for fast path planning," in *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, 1994, pp. 2138–2145.

[11] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[12] I. A. Sucan, M. Moll, and L. E. Kavraki, "The open motion planning library," *IEEE Robot. Autom. Mag.*, vol. 19, no. 4, pp. 72–82, 2012.

[13] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, 2009, pp. 489–494.

[14] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*, 2011, pp. 4569–4574.

[15] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[16] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, 2000, vol. 1, pp. 521–528.

[17] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, 2000, vol. 2, pp. 995–1001 vol.2, doi: 10.1109/ROBOT.2000.844730.

[18] M. Zucker *et al.*, "Chomp: Covariant hamiltonian optimization for motion planning," *Int. J. Robot. Res.*, vol. 32, no. 9–10, pp. 1164–1193, 2013.