The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-021-07985-5

ORIGINAL ARTICLE

Benchmarking and optimization of robot motion planning with motion planning pipeline

Shuai Liu[1] · Pengcheng Liu[1]

Received: 21 March 2021 / Accepted: 30 August 2021

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.

Keywords Robot motion planning · Benchmarking · Optimization · Motion planning pipeline · Manipulation

1 Introduction

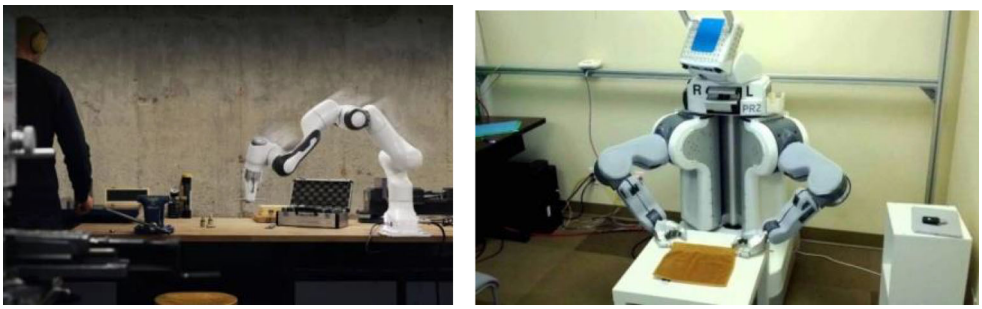
With the development and wide application of robot technology, robots reflect their importance and superiority in production and life. Robots have become an effective tool and experimental platform for studying complex intelligent behaviors and exploring human thinking patterns. The robotic arm is one 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 have 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 [3]. The basic task of motion planning can be described as the movement from the starting state to the target state [4–8]. 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.

For decades, as robots have become an important part of modern industry and daily life, motion planning algorithms have received a lot of attention. The earliest motion planning problem is to consider how to move a piano from one room to another room collision-free with any objects (the piano mover's problem). In this problem, the piano has six degrees of freedom (x, y, z, roll, pitch, yaw). In order to solve this problem, a data set containing the six parameters of the piano must be set and calculated, which is time parameterized, and each parameter should be continuous for the change of time,

✉ Pengcheng Liu
pengcheng.liu@york.ac.uk

[1] Department of Computer Science, University of York, York YO10 5GH, UK

Published online: 10 September 2021



(a) (b)

Fig. 1 The robotic arm assists people in their work. **a** Franka Emika Panda [1]. **b** PR2 Robot [2]

moving from one configuration space (starting configuration) to another configuration space (goal configuration), while meeting the internal and external constraints of obstacle avoidance. As early as 1978, Lozano Perez and Wesley introduced the conceptual construction planner of configuration space (C-Space), which was an epoch-making revolution for modern motion planning. In 1979, the motion planning of the piano porter problem proved to be a PSPACE-hard problem. Traditional motion planning includes subdivision algorithm, the potential field method, and roadmap algorithm. These algorithms have a common feature. When their parameters are set appropriately, these algorithms can ensure the integrity of the plan. And the upper limit of the time spent can also be guaranteed in a complex environment.

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 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 and 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 analyze 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.

This paper is organized as follows. Section 2 investigates the related works in motion planning and formulate the problem. Section 3 discusses the motion planning benchmarking framework and optimization procedure. In Section 4, the benchmarking and optimization results are analyzed and discussed. Finally, conclusions are outlined in Section 5.

2 Related works

Barraquand J and Latombe J C first proposed the RPP method of motion planning based on sampling [9]. It was this method that influenced the subsequent random potential field method. At the same year, another random sampling motion planning method called ZZ-method was proposed in [15]. This method has had a huge impact on the probabilistic roadmaps (PRM) algorithm. Then the first stochastic method in history to solve the problem of complex motion planning in high-dimensional space in a true sense, that is, the PRM method, was studied [16–18]. The PRM method has quickly become one of the most popular motion planning methods based on random sampling in recent years. It has shown good performance in many high-dimensional space planning applications [19, 20]. J. Kuffner and Steven M. LaValle proposed rapidly exploring random trees (RRT) [21]. This algorithm is another motion planning algorithm based on random sampling that has been widely developed and applied in the past 10 years. The RRT algorithm is suitable for solving the path planning problems of mufti-degree-of-freedom robots in complex and dynamic environments [22]. In 2000, LaValle and Kuffner jointly proposed the RRT-Connect algorithm [23]. This bidirectional RRT technology has good search characteristics. Compared with the original rapidly exploring random tree algorithm, the search speed and search efficiency have been significantly improved. The improved algorithm of PRM, Lazy PRM, was also proposed by R. Bohlin and L. E. Kavraki in 2000. This method reduces the preprocessing process of collision detection in the sampling phase and speeds up planning [24]. In 2010, Sertac and Emilio of MIT proved that in the sampling-based motion planning algorithm, as the sampling points of the RRT algorithm tend to infinity, the probability of converging to the optimal solution is 0. For this reason, they proposed the asymptotic optimality RRT* algorithm [25]. In recent years, there have been some further optimization studies on RRT and PRM. Liangjun Zhang and D. Manocha proposed a new optimization-based retraction algorithm and an enhanced version of RRT planner [26]. The optimization-based retraction algorithm can improve the performance of the RRT planner by shrinking the sample, making its sample more likely to be connected to the tree. D. Kim, Y. Kwon, and S. Yoon proposed an adaptive lazy collision checking named adaptive lazy PRM* in 2018 [27]. The planning algorithm is based on lazy PRM, and its convergence rate is 20 to 250% faster than previous methods.

Regarding optimization-based methods, CHOMP and STOMP are typical representatives of optimization algorithms. CHOMP was proposed in 2009. It is a new method that uses covariant gradient techniques to improve the quality of the sampled trajectory. It is a new trajectory optimization program based on covariant gradient descent [13, 28]. In 2011, Zucker M et al. proposed STOMP [14]. This is the first method of motion planning using a stochastic trajectory optimization framework to explore the space around the trajectory by generating noisy trajectories. This method is different from CHOMP in that it is optimized by a specific optimization algorithm and does not require gradient information. In recent years, CHOMP and STOMP have also received great attention and further research [29–31]. In this article, we use the planning request adapter in Moveit to modify the trajectory created by the sampling algorithm in OMPL. The initial motion plan generated by OMPL is optimized by CHOMP/STOMP to generate a new motion planning trajectory. In this process, we set STOMP/CHOMP as the post-process of an algorithm of OMPL and use the planning pipeline which chains a motion planner (OPML) with post-processing (STOMP/CHOMP) stages and benchmark the result.

With the continuous increase of a large number of different types of planning algorithms, the evaluation of different algorithms, the fitness of different scene, and the comparison of different algorithms are still one of the distressing problems for scholars who study this field. In the past two decades, a large number of motion planning algorithms have been developed. M. Elbanhawi and M. Simic conducted a general review of sampling-based planner and a comprehensive survey of the ever-growing work body in the sampling-based planner [32]. Although they have considered all aspects, they did not conduct actual benchmarking and did not propose related information about benchmarks. It is difficult for novices and experts to compare which type of problem these different algorithms are suitable for.

When most motion plans are developed, there are corresponding benchmarkings. In the early, Baltes J. researched a benchmark kit for mobile robots, which provided a quantitative measure of the mobile robot's ability to perform specific tasks [33]. And through benchmarking, the path and trajectory tracking control and accuracy of the mobile robot, static path planning, and dynamic path planning capabilities were tested. The benchmarking can also be used as simple games. Including them in robotic games will allow researchers to increase their chances of evaluating their work without having to purchase expensive or specialized equipment. At the same

time, in the path planning of humanoid robots, the well-known "narrow passage" problem [34] associated with stochastic programming is considered. A benchmark system was proposed in [35] for comparing different sample-based motion planners in OMPL. The framework of this benchmark is very easy to access and use, and the benchmarks in the system can be downloaded and run for new sample-based motion planning algorithms. But it does not contain any information about the optimized planner. The benchmark system is based on the open-source framework of Moveit! [36]. Based on previous work, M. Moll, I. A. Sucan, and L. E. Kavraki proposed a benchmark infrastructure in 2015 that is not aimed at any specific benchmark problems, measurement standards, or planning procedures [37]. The infrastructure is a universal and extensible benchmarking, which can help researchers easily analyze and visualize reproducible benchmarking results. But the basic structure of the benchmarking is integrated with OMPL, and no other planners are benchmarked, especially without any optimization-based motion planning. This benchmark is also the benchmark framework used in this article, which benchmarking OMPL, CHOMP, STOMP, and sample planners optimized by CHOMP/STOMP. Although the benchmarking is a broad and general software framework, other different motion planning algorithms and OMPL extensions algorithm are still to be studied. In this paper, we aim to fill this gap.

3 Motion planning benchmarking and optimization

In this paper, we propose a framework of planning adapters to combine sampling-based planning algorithms (PRM [18], RRT, LazyPRM [24], and RRTConnect [21] 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 (see Appendix).

3.1 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 RRT-Connect 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.

3.2 Motion planning algorithm based on trajectory optimization

3.2.1 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 scenes 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 [28]. 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.

3.2.2 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.

3.3 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.

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, AddTimeParameterization, CHOMPOptimizerAdapter, and STOMPSmoothing-Adapter. 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; and (3) combining the planner with the plan requests that the adapters are grouped together in a serialized manner.

3.4 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

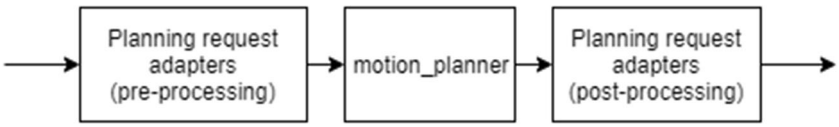


Fig. 2 Motion planning pipeline

algorithm is 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.

4 Experiment results

4.1 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.

4.2 Motion planning benchmarking

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

4.2.2 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 does not 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. 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 picks 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

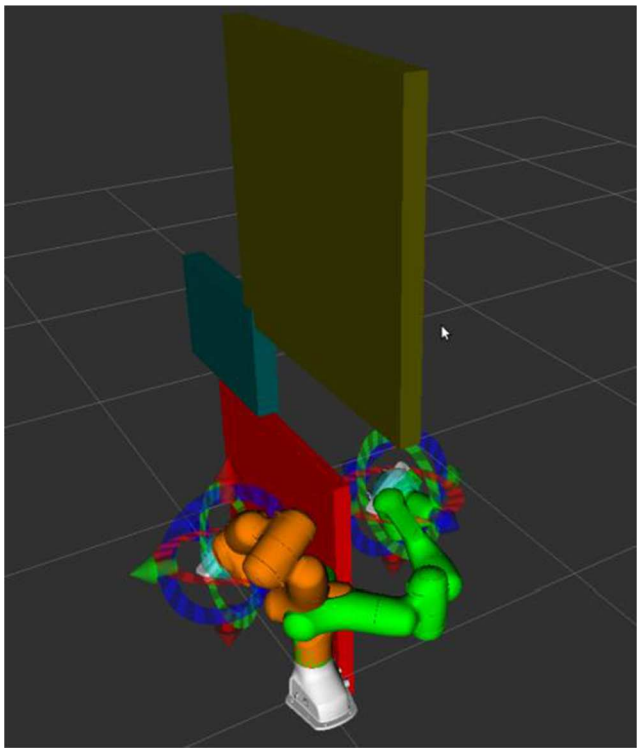
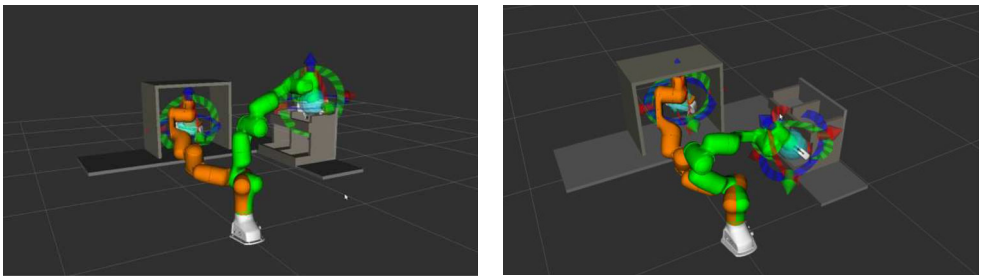


Fig. 3 Narrow tunnel



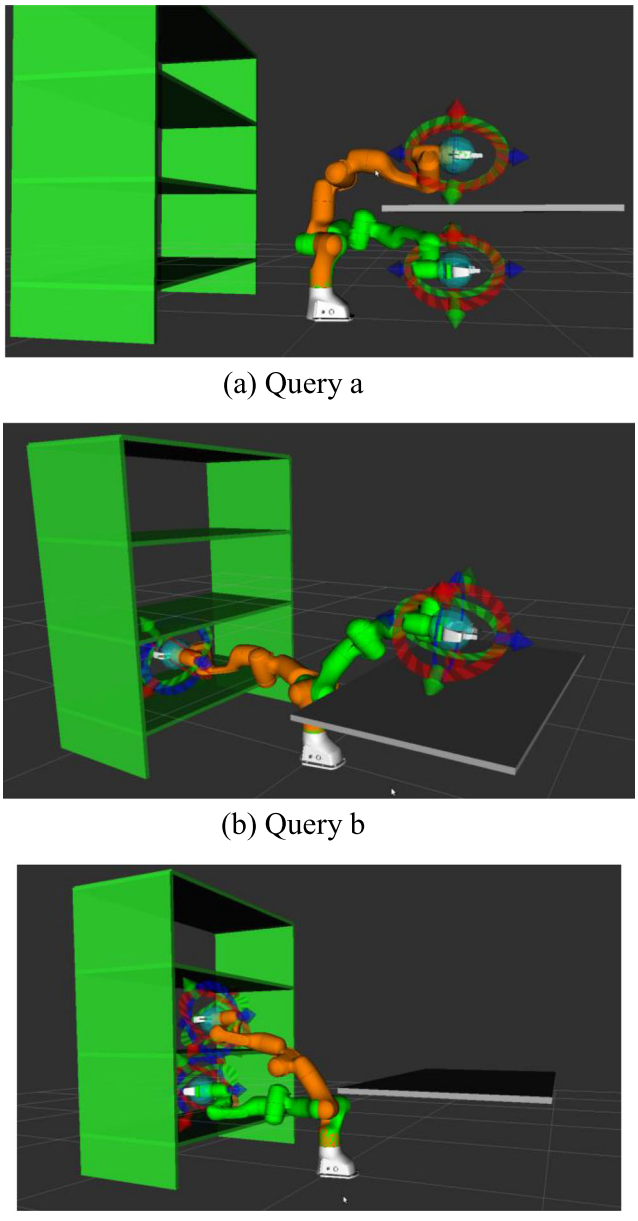
(a) Query a (b) Query b

Fig. 4 The queries of industrial workshop

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 shown in Fig. 5. 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. It is



(a) Query a

(b) Query b

(c) Query c

Fig. 5 The queries of library scene

noted that the main results/findings are based on motion planning which are prior grasping tasks, so including the grasping in the system will not affect the results.

4.2.3 Benchmarking metrics

This experiment measures the performance, reliability, and quality of the paths generated by all planners via the following 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, simplification time, and process time. The smaller the metric, the better the performance of the motion planner.

4.2.4 The process of benchmarking

The experiment contains scene design, motion planner selection, planning adapter setting, and benchmarking. In each scene, benchmarking the different queries is 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 path planning 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.

5 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 Appendix.

5.1 Narrow space (tunnel)

Table 1 shows that, in this scenario, the resolution rate of the original algorithm is almost 100%, and only the CHOMP

Table 1 Tunnel benchmarking results

Planners	Solve (%)	Path time (s)	Path smoothness	Path clearance	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

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.

5.1.1 Industrial

a. The first floor parts box to table query:

In this query, the original algorithm PRM, RRTConnect, and STOMP all have a good problem solving rate. As shown in Table 2, the RRT-Connect 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:

In this query, the problem solving rate of RRT, PRM, and STOMP in the original algorithm is still the highest. As shown in Table 3, 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.

Table 2 Industrial_FTT benchmarking results

Planner	Solve (%)	Path time (s)	Path smoothness	Path clearance	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

Table 3 Industrial_STT benchmarking results

Planner	Solve (%)	Path time (s)	Path smoothness	Path clearance	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

5.1.2 Library

a. Under table to on table query:

Under the query in the library scene, we can see from Table 4 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 time 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 4 Library_DTU benchmarking results

Planner	Solve (%)	Path time (s)	Path smoothness	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

Table 5 Library_TTB benchmarking results

Planner	Solve (%)	Path time (s)	Path smoothness	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, it can be seen from Table 5 that 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 RRT-Connect 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 the third floor to the second floor on the bookshelf query:

As shown in Table 6, the complexity of the query is very high, so that the original algorithm has the problem of planning failure. Only the PRM and RRT-Connect algorithms have a higher success rate. STOMP has also been greatly affected, and its success rate is only 32%, which also leads

Table 6 Library_ B3TB2 benchmarking results

Planner	Solve (%)	Path time (s)	Path smoothness	Path clearance	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

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.

6 Conclusions

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 and PRM-STOMP. 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.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s00170-021-07985-5.

Author contribution Pengcheng Liu contributes to the conception and design of the work and critical revisions of the article, Shuai Liu drafts the article and conducts the experiments and related analysis.

Data availability The data that support the findings of this study are available from the corresponding author, P Liu, upon reasonable request.

Code availability The code that supports the findings of this study is available from the corresponding author, P Liu, upon reasonable request.

Declarations

Competing interests The authors declare no competing interests.

References

1. Overview — Franka Control Interface (FCI) documentation. https://frankaemika.github.io/docs/overview.html. Accessed 19 Oct 2020
2. Robots/PR2 - ROS Wiki. http://wiki.ros.org/Robots/PR2. Accessed 24 Oct 2020
3. Shyam RA, Lightbody P, Das G, et al (2019) Improving local trajectory optimisation using probabilistic movement primitives. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, pp 2666–2671. https://doi.org/10.1109/IROS40897.2019.8967980
4. Liu P, Yu H, Cang S (2018) Geometric analysis-based trajectory planning and control for underactuated capsule systems with viscoelastic property. Trans Inst Meas Control 40:2416–2427
5. Huda MN, Liu P, Saha C, Yu H (2020) Modelling and motion analysis of a pill-sized hybrid capsule robot. J Intell Robot Syst. https://doi.org/10.1007/s10846-020-01167-3
6. Liu P, Yu H, Cang S (2018) Optimized adaptive tracking control for an underactuated vibro-driven capsule system. Nonlinear Dyn 94:1803–1817
7. Liu P, Yu H, Cang S (2018) Trajectory synthesis and optimization of an underactuated microrobotic system with dynamic constraints and couplings. Int J Control Autom Syst 16:2373–2383

8. Liu P, Yu H, Cang S (2019) Adaptive neural network tracking control for underactuated systems with matched and mismatched disturbances. Nonlinear Dyn 98:1447–1464. https://doi.org/10.1007/s11071-019-05170-8
9. Barraquand J, Latombe J-C (1990) A Monte-Carlo algorithm for path planning with many degrees of freedom. In: Proceedings., IEEE International Conference on Robotics and Automation. IEEE, vol. 3 pp 1712–1717. https://doi.org/10.1109/ROBOT.1990.126256
10. Kavraki L, Latombe J-C (1994) Randomized preprocessing of configuration for fast path planning. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation. IEEE, vol. 3 pp 2138–2145. https://doi.org/10.1109/ROBOT.1994.350966
11. LaValle SM (1998) Rapidly-exploring random trees: a new tool for path planning. Technical Report No. 98-11 (Iowa State Univ., 1998)
12. Sucan IA, Moll M, Kavraki LE (2012) The open motion planning library. IEEE Robot Autom Mag 19:72–82
13. Ratliff N, Zucker M, Bagnell JA, Srinivasa S (2009) CHOMP: Gradient optimization techniques for efficient motion planning. In: Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. IEEE, pp 489–494. https://doi.org/10.1109/ROBOT.2009.5152817
14. Kalakrishnan M, Chitta S, Theodorou E, et al (2011) STOMP: Stochastic trajectory optimization for motion planning. In: 2011 IEEE international conference on robotics and automation. IEEE, pp 4569–4574. https://doi.org/10.1109/ICRA.2011.5980280
15. Barraquand J, Latombe J-C (1991) Robot motion planning: a distributed representation approach. Int J Robot Res 10:628–649
16. Kavraki L, Latombe J-C (1994) Randomized preprocessing of configuration space for path planning: articulated robots. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'94). IEEE, vol. 3 pp 1764–1771. https://doi.org/10.1109/IROS.1994.407619
17. Amato NM, Wu Y (1996) A randomized roadmap method for path and manipulation planning. In: Proceedings of IEEE international conference on robotics and automation. IEEE, vol. 1 pp 113–120. https://doi.org/10.1109/ROBOT.1996.503582
18. Kavraki LE, Svestka P, Latombe J-C, Overmars MH (1996) Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Trans Robot Autom 12:566–580
19. Amato NM, Song G (2002) Using motion planning to study protein folding pathways. J Comput Biol 9:149–168
20. Fox D, Burgard W, Kruppa H, Thrun S (2000) A probabilistic approach to collaborative multi-robot localization. Auton Robot 8:325–344
21. Kuffner JJ, LaValle SM (2000) RRT-connect: an efficient approach to single-query path planning. In: IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00. pp 995–1001 vol.2. https://doi.org/10.1109/ROBOT.2000.844730
22. Ettlin A, Bleuler H (2006) Randomised rough-terrain robot motion planning. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp 5798–5803. https://doi.org/10.1109/IROS.2006.282390
23. LaValle SM, Kuffner JJ (2001) Rapidly-exploring random trees: progress and prospects. Algorithmic Comput Robot New Dir 5:293–308
24. Bohlin R, Kavraki LE (2000) Path planning using lazy PRM. In: Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065). IEEE, pp 521–528. https://doi.org/10.1109/ROBOT.2000.844107
25. Karaman S, Frazzoli E (2010) Incremental sampling-based algorithms for optimal motion planning. Robot Sci Syst VI 104(2)
26. Zhang L, Manocha D (2008) An efficient retraction-based RRT planner. In: 2008 IEEE International Conference on Robotics and Automation. IEEE, pp 3743–3750. https://doi.org/10.1109/ROBOT.2008.4543785
27. Kim D, Kwon Y, Yoon S (2018) Adaptive lazy collision checking for optimal sampling-based motion planning. In: 2018 15th International Conference on Ubiquitous Robots (UR). IEEE, pp 320–327. https://doi.org/10.1109/URAI.2018.8442203
28. Zucker M, Ratliff N, Dragan AD et al (2013) Chomp: covariant Hamiltonian optimization for motion planning. Int J Robot Res 32:1164–1193
29. Osa T (2020) Multimodal trajectory optimization for motion planning. Int J Robot Res 39:983–1001
30. He K, Martin E, Zucker M (2013) Multigrid CHOMP with local smoothing. In: 2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids). IEEE, pp 315–322. https://doi.org/10.1109/HUMANOIDS.2013.7029993
31. Magyar B, Tsiogkas N, Brito B et al (2019) Guided stochastic optimization for motion planning. Front Robot AI 6:105
32. Elbanhawi M, Simic M (2014) Sampling-based robot motion planning: a review. IEEE Access 2:56–77
33. Baltes J (2000) A benchmark suite for mobile robots. In: Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000)(Cat. No. 00CH37113). IEEE, vol. 2pp 1101–1106. https://doi.org/10.1109/IROS.2000.893166
34. Xia Z, Chen G, Xiong J, et al (2009) A random sampling-based approach to goal-directed footstep planning for humanoid robots. In: 2009 IEEE/ASME International Conference on Advanced Intelligent Mechatronics. IEEE, pp 168–173. https://doi.org/10.1109/AIM.2009.5230019
35. Cohen B, Şucan IA, Chitta S (2012) A generic infrastructure for benchmarking motion planners. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, pp 589–595. https://doi.org/10.1109/IROS.2012.6386228
36. Chitta S (2016) MoveIt!: an introduction. In: Robot operating system (ROS). Springer, Cham, pp 3–27
37. Moll M, Sucan IA, Kavraki LE (2015) Benchmarking motion planning algorithms: an extensible infrastructure for analysis and visualization. IEEE Robot Autom Mag 22:96–102