A novel planner framework compatible with various end-effector constraints

Yahao Wang, Yanghong Li, Zhen Li and HaiYang He
Department of Precision Machinery and Precision Instrumentation, University of Science and Technology of China, Hefei, China
Sheng Chen
China Electric Power Research Institute, Beijing, China, and
Erbao Dong
Department of Precision Machinery and Precision Instrumentation, Key Laboratory of Precision and Intelligent Chemistry, University of Science and Technology of China, Hefei, China

Abstract

Purpose – Aiming at the problem of insufficient adaptability of robot motion planners under the diversity of end-effector constraints, this paper proposes Transformation Cross-sampling Framework (TC-Framework) that enables the planner to adapt to different end-effector constraints.

Design/methodology/approach – This work presents a standard constraint methodology for representing end-effector constraints as a collection of constraint primitives. The constraint primitives are merged sequentially into the planner, and a unified constraint input interface and constraint module are added to the standard sampling-based planner framework. This approach enables the realization of a generic planner framework that avoids the need to build separate planners for different end-effector constraints.

Findings – Simulation tests have demonstrated that the planner based on TC-framework can adapt to various end-effector constraints. Physical experiments have also confirmed that the framework can be used in real robotic systems to perform autonomous operational tasks. The framework's strong compatibility with constraints allows for generalization to other tasks without modifying the scheduler, significantly reducing the difficulty of robot deployment in task-diverse scenarios.

Originality/value – This paper proposes a unified constraint method based on constraint primitives to enhance the sampling-based planner. The planner can now adapt to different end effector constraints by opening up the input interface for constraints. A series of simulation tests were conducted to evaluate the TC-Framework-based planner, which demonstrated its ability to adapt to various end-effector constraints. Tests on a physical experimental system show that the framework allows the robot to perform various operational tasks without requiring modifications to the planner. This enhances the value of robots for applications in fields with diverse tasks.

Keywords Motion planning, Constrained planning, Rapidly-exploring random trees, Manifold constraints

Paper type Research paper

Introduction

Robots in fields such as disaster relief, spacecraft, logistics and home care are expected to perform a large number of operational tasks (opening doors, serving water, etc.). This raises the issue of the diversity of end-effector constraints. Imagine a service robot in a chemistry lab that needs to help a lab technician remove a test tube containing an experimental liquid from a thermostat (Burger *et al.*, 2020). As shown in Figure 1, to accomplish this task, the robot needs to complete a sequence of operations, starting with opening the lid of the thermostat, pulling out the drawer and removing the test tube. These actions, such as opening lids and pulling out drawers, introduce different end-effector constraints to limit the motion of the robot's end-effector. Instead of multiple specialized planners, we expect a general-purpose planner to produce the desired trajectories for the above tasks (Park *et al.*, 2022). This brings new challenges to robot motion planning.

Robot motion planning refers to the generation of collision-free paths to connect given start and goal configurations in response to a planning task. Many fruitful approaches already exist for the basic form motion planning problem (Qureshi *et al.*, 2022; Secil and Ozkan, 2023; Nakatsuru *et al.*, 2023; Wang *et al.*, 2024). However, these methods often fail when dealing with end-effector constraints because the motion process connecting the start configuration and the target configuration is also considered part of the task (Brock and Khatib, 2002).

The current issue and full text archive of this journal is available on Emerald Insight at: **https://www.emerald.com/insight/2754-6969.htm**

Robotic Intelligence and Automation
44/5 (2024) 746–759

[DOI 10.1108/RIA-02-2024-0043]

The authors gratefully acknowledge support from the National Key R&D Program of China (NO. 2018YFB1307400) and the Students "Innovation and Entrepreneurship Foundation of USTC".

Received 27 February 2024
Revised 6 May 2024
11 June 2024
Accepted 17 June 2024

Initially, researchers attempted several methods to solve the planning problem under constraints (Kingston *et al.*, 2018), but most of them have some limitations. Moreover, these methods lose some of their performance when generalized to high-dimensional robotic systems. Some researchers have proposed incorporating end-effector constraints by adding constraint modules to the planner, leveraging the modular nature of sampling-based planning algorithms (Kingston *et al.*, 2019).

Therefore, to adapt to scenarios with different constraints, all possible end-effector constraints must be pre-modelled to manually construct the corresponding constraint modules. However, this approach is uneconomical given the diversity of constraints. At the same time, with the application of large-scale models, robots are expected to be abstracted into individual functional modules or even individual functions. Thus, further improvement in the generalization capabilities of motion planners is also necessary for the development of robotics (Miao *et al.*, 2023). For these two reasons, we plan to design a robot motion planner that can handle the diversity of end-effector constraints.

The main contribution of this work is the proposal of a robot object-oriented planner framework called the Transformation Cross-sampling Framework (TC-Framework). This framework introduces constraint input interfaces, enabling planners based on it to adapt to various end-effector constraints by switching inputs. This offers a cost-effective solution for deploying robots in scenarios with multiple constraints.

The rest of the paper is organized as follows: Section 2 summarizes the development of motion planning methods under end-effector constraints. Section 3 presents the basic definitions related to motion planning and end-effector constraints. Section 4 describes how the TC-Framework works. Section 5 presents simulation experiments to verify the effectiveness of the proposed method. Section 6 shows the effect of the proposed method deployed in a physical experimental system. In Section 7, conclusions are given and an outlook is presented.

2. Related work

End-effector constraints first appeared in assembly tasks for industrial robots through industrial control to generate the desired motions for tasks such as grinding and welding (Khatib, 1987). Due to the suboptimal performance of robots, early research concentrated on motion planning problems in low-dimensional spaces (Mitchell *et al.*, 1987). As the demand for robots increased, robotic arms were used to perform tasks such as painting and assembling, introducing distinct end-effector constraints (Foumani *et al.*, 2015; Kolakowska *et al.*, 2014; Liu *et al.*, 2022). These constraints need to be re-conceived as a difficult addition to the motion planning problem as the robot gains further degrees of freedom and the planner is applied to more complex high-dimensional systems with more interesting operational tasks. Some research on accomplishing certain complex operational tasks via motion control has been conducted (Phoon *et al.*, 2022).

A common practice for end-effector constraints is to plan in the robot workspace to generate desired motions and map the planning results to the configuration space via inverse kinematics techniques. Rakita *et al.* (2020) proposed the RelaxedIK method to avoid the joint space discontinuity problem. However, the completeness of this method is difficult to guarantee because the workspace is not homeomorphous with the configuration space. Another class of methods uses optimization-based methods (Zucker *et al.*, 2013; Bonalli *et al.*, 2019) proposed a method for trajectory optimization on implicitly defined manifolds that avoids the shortcomings of optimizing in the workspace. However, how to avoid falling into local minima is a challenge that such methods must overcome. In addition, finding a suitable cost function for optimization is also a challenge (Bonalli *et al.*, 2019; Wang *et al.*, 2022).

Sampling-based motion planning algorithms provide a new solution idea. First of all its core idea is to sample the configuration space to avoid computing the free configuration space Q_{free}. With the introduction of end-effector constraints, the free configuration space is reduced to a zero-measure manifold, making it nearly impossible to randomly sample points in the free configuration space, thus causing sampling-based motion planning algorithms to fail (Kingston *et al.*, 2018).

The end-effector constraint can be regarded as a special kind of manifold constraint. Bonilla *et al.* (2017) relaxed the constraints that need to be strictly enforced by introducing a tolerance error and successfully applied it to the dual-arm operational problem. This mechanism is a relaxation-based approach that restores the effectiveness of sampling-based motion planning algorithms by relaxing the constraints and augmenting the low-dimensional constrained manifold into a full-dimensional boundary layer (Fusco *et al.*, 2019). However, the planner is still inefficient after converting the problem into a narrow channel problem. The idea of relaxing the constraints within a tolerance error is also retained in other methods.

Another approach involves attaching the random points obtained from each sampling to the constrained flow pattern using Newton's iteration of the Jacobi inverse matrix. This iterative process projects the sampling results onto configurations that satisfy the constraints (Yakey *et al.*, 2001). The projection-based approach has been widely adopted for solving general end-effector constraint problems by Yao and Gupta (2005). Building upon this, Lamiraux and Mirabel (2022) developed a Humanoid Path planner software platform for solving various operational planning problems. In recent research, cast shadows are generated using neural networks to further increase the computational speed (Qureshi *et al.*, 2022).

In another class of methods, a tangent space is used to constrain the streaming neighbourhood approximation. This differs from the projection method in that random points are eventually attached to the linear approximation (Stilman, 2010; Yakey *et al.*, 2001) have applied the method to the motion planning problem for an articulated robot with a closed kinematic chain. Jaillet and Porta (2013) further proposed a strategy to reuse the cut space instead of discarding it after computation to reduce the computational complexity. The elimination of the error problem posed by this linear

approximation is discussed in the work of Bordalba *et al.* (2023). Building upon this, Kingston *et al.* (2019) proposed a unified framework, implicit manifold configuration space to address end-effector constraints and integrate Projection and Atlas methods into it. However, the aforementioned approaches focus on solving single-constraint problems and do not discuss how to incorporate multiple end-effector constraints (Lamiraux and Mirabel, 2022). The need for planners in complex operational tasks to respond to switched end-effector constraints and generate operation sequences is also a practical issue that must be considered (Phoon *et al.*, 2022). In some studies, such planning tasks are considered as multimodal (Kingston and Kavraki, 2023; Palleschi *et al.*, 2022). These tasks can be solved by a combination of multiple planners targeting a single constraint. However, the fundamental problem remains that designing a planner specifically for a particular constraint is uneconomical.

3. Parameterization

3.1 Definition of motion planning under constraints

In this work, we investigate the motion planning problem under end-effector constraints, for which we provide the basic definition.

A robot is represented as a single point q in the Configuration Space. The Configuration Space (CS) is the space encompasses all possible configurations and is a metric space with dimension n equal to the degrees of freedom of the robot system.

To address the basic motion planning problem, the robot must navigate from the initial configuration q_{inital} to q_{goal} while avoiding obstacles. Then, we define CS_{free}, a subset of CS, comprising all configurations in the CS that do not result in collisions. Consequently, a collision-free motion can be defined as a continuous injective mapping $\tau[0,1] \rightarrow CS_{free}$.

Then, one can define the basic motion planning problem as finding such a collision-free trajectory given q_{inital}, q_{goal} and CS_{free}.

However, the situation changes for the planning problem under end-effector constraints to be solved in this work. A segment actuator constraint typically means that the robot's end-effector position is limited during the motion. We can represent the position of the end-effector by a $6D$ vector X, where $X \in se(3)$. Then, the constraint C can be defined as:

$$C : H(X) = 0 \tag{1}$$

Here, we only discuss the case of hard constraints, $H(X) = 0$. Furthermore, one can obtain $X = FK(q)$ from the positive robot kinematics. Then, one has $H\big(FK(q)\big) \Rightarrow F(q) : \mathbb{R}^n - \mathbb{R}^k$. Thus, the constrained manifold in the CS can be defined as:

$$MC = \{q \,|\, F(q) = 0\} \tag{2}$$

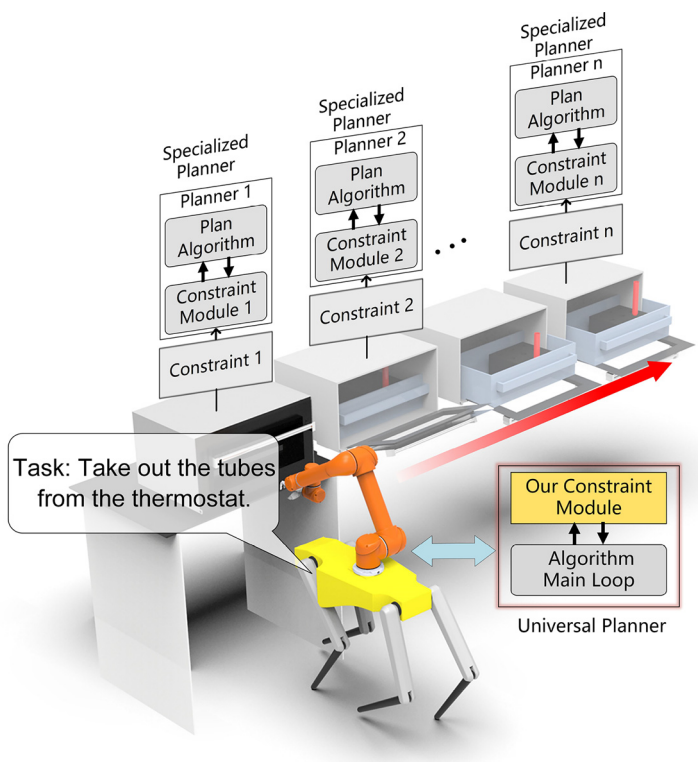
The original free configuration space under this constraint is compressed into MC_{free}, which is the intersection of CS_{free} and MC.

The motion planning problem is to find such a collision-free motion $\tau[0,1] \rightarrow MC_{free}$ given q_{inital}, q_{goal} and MC_{free}.

3.2 Motion planners under constraints

In this section, we describe how the planner operates. Given that most robotic systems faced are high-dimensional, we

Figure 1 Different methods generate the motion for the sequence of operations



Source: Authors'own

introduce sampling-based planners that efficiently perform in high-dimensional spaces. Sampling-based planning methods continuously explore the configuration space by generating nodes that satisfy constraints to find paths connecting the initial and goal configurations.

Firstly, in basic motion planning problems, the sampling strategy commonly involves randomly sampling in CS, discarding the nodes located on $CS_{obstacle}$ and retaining the nodes located on CS_{free}. This mechanism ensures that the planned motion satisfies the obstacle avoidance constraints.

However, when additional constraints need to be incorporated, the chance of obtaining the nodes located on the MC_{free} by random sampling is almost zero. Because CS_{free} is downscaled to MC_{free}. Thus, the rejection sampling approach fails.

Therefore, in previous research work, researchers have proposed many constraint methods to incorporate end-effector constraints. The goal is to project any node onto the constraint prevalence or an approximation of the constraint prevalence by constructing a constraint module $CON\big(F(q)\big)$.

We can define a *Planenr*:

$$Planenr = Build\{CS, CON\big(F(q)\big)\} \tag{3}$$

The constructed planner under the end-effector constraints needs to obtain the constraint function $F(q)$ in advance. The constraint information $INF(C)$ for a real robot system is always obtained through sensors, and its description space is the workspace. The process of transforming the system into the configuration space and further designing the abstraction function $F(q)$ are highly complicated. A powerful and unified method for designing $F(q)$ has not been obtained in the existing literature. Creating planners in real time is almost impossible. Furthermore, only fixed constraints can be solved for planners that have already been designed or a certain class of constraints can be solved by parameterization.

Therefore, a constraint module that no longer relies on *a priori* information about the constraints for its construction must be developed to improve the ability of the planner to generalize to different constraints.

4. Our method

The purpose of this work is to provide a planner framework for scenarios with diverse end-effector constraints.

4.1 Overall framework

Within the TC-Framework, the planner under end-effector constraints is divided into three components: constraint input interface, constraint module and planning algorithm.

In this framework, constraint information is obtained by encoding the positional information of the operating object bits obtained from the sensing system. To accommodate different end-effector constraints, we add a uniform constraint input interface to the framework. Then, based on this, a standard constraint method is designed to incorporate the input constraints into the planner. The constraint module of the integrated constraint method runs in each iteration of the planning algorithm and corrects the sampling points obtained from the iteration to make them satisfy the constraints.

Next, a brief description of how the framework takes effect is given. As shown in Figure 2, firstly, the position information of the target is obtained through the perception module. Then, the positional information of the job target is encoded into constraint information that can be input to the planner using the Encoder. Currently, the Encoder is defined based on different tasks. Upon receiving the constraint information, initial and target points, the planner commences operation. The initial points are used to initialize the random tree and random sampling in the CS guides the random tree to explore the CS. In each iteration of the planner, the expansion process of the random tree is corrected to satisfy the end-effector constraints by the constraint module, which receives the constraint information. The planner detects the corrected nodes through collision detection and discards invalid nodes to satisfy obstacle avoidance constraints. The iterations are repeated until a collision-free trajectory is found that connects the initial and target points. In *Algorithm* 1, we provide the pseudocode of the rapidly-exploring random trees (RRT) that uses this constraint module.

Algorithm 1: RRT

Input: $q_{initial}, q_{goal}, C$
Output: $Path$

$Tree \leftarrow$ InitialTree($q_{initial}$);
while Distance($Tree, q_{goal}$) $> d_0$ **do**
  $q_{rand} \leftarrow$ Random(Q);
  $Motion_{extend} \leftarrow$ TreeExtend($Tree, q_{rand}, d$);
  $q_{new} \leftarrow$ ConstraintModule($Motion_{extend}, q_{rand}$);
  if Collision(q_{new}) **then**
    $Node_{new} = q_{new}$;
    $Tree \leftarrow$ Update($Tree, Node_{new}$);
  end
end
$Path \leftarrow$ FindPath($Tree$);

4.2 Constrained input interface

Inspired by the work of Wang *et al.* (2021), constraints are decomposed into several constraint primitives. Then different types of constraints can be represented as a superposition of several primitives. With this design, it is possible to represent arbitrary types of constraints using a uniform form.

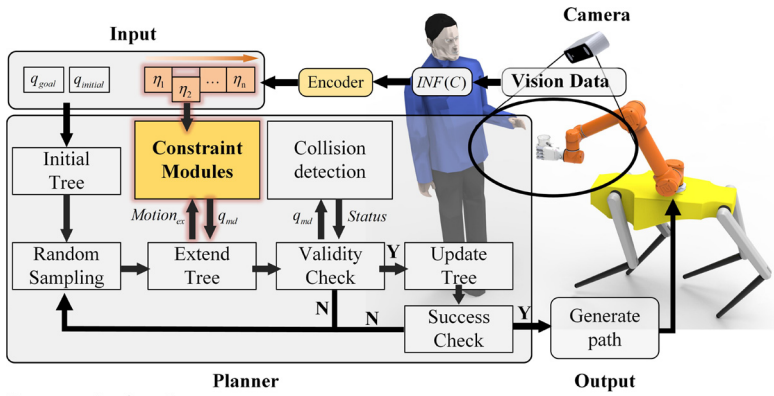
Firstly, the motion of an arbitrary rigid body in space can be described as a spiral motion. Usually, end-effector constraints originate from restrictions on the spiral motion imposed by properties of the operating object itself. As shown in Figure 3, the robot needs to slide a cup of water on a tabletop; then the robot end-effector is only allowed a combined motion of translation along the tabletop and rotation along the normal. Then, several orthogonal unit motions can be defined to describe all the allowed motions. Consider the unit motions as constraint primitives. Thus, a constraint can be represented as a matrix consisting of a number of constraint primitives $\boldsymbol{\eta}$.

$$C : \boldsymbol{\eta} = \{\eta_1, \eta_2 \cdots \eta_k\}. \tag{4}$$

where η_i is a 6 D vector. k denotes the number of orthogonal unit motions. As shown in Figure 3, it can encode the positional information into constrained primitives by means of empirical formulas $Fml_{()}$:

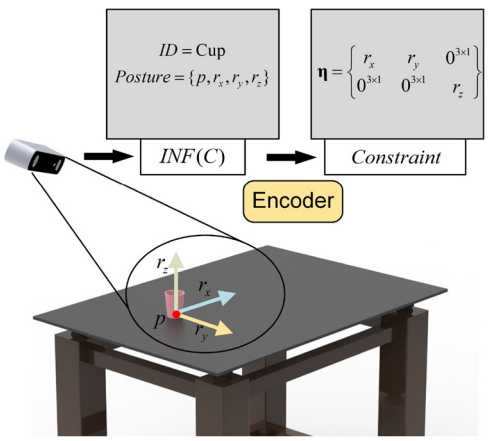
$$\begin{aligned} \boldsymbol{\eta} &= Fml(Posture) \\ Posture &= \begin{bmatrix} p & r_x & r_y & r_z \end{bmatrix}. \end{aligned} \tag{5}$$

Figure 2 Overall flowchart of the planner under end-effector constraints



Source: Authors'own

Figure 3 Simple example of how to obtain constraint representations based on constraint primitives from sensor information



Source: Authors'own

where $A_0 = A(X_0)$. The constraint can also be represented in the X_0 neighbourhood by $\boldsymbol{\eta}_{X_0}$.

The end-effector constraint is decomposed into k constraint primitives by using this modelling approach. Each constraint primitive represents the direction in which the end-effector can move. And k denotes the number of degrees of freedom that the constraint is open to the end-effector. Therefore, k can be used to indicate the stringency of the constraint, as it reflects the degree of restriction of the end-effector motion.

It is also interesting to note that the constraint primitives for many common constraints can be obtained from the sensor information without going through a complex encoding process.

4.3 Constraint method

To incorporate constraints through several constraint primitives into the planner, a generic constraint module is proposed, which is optimized for RRT and its variants of the algorithm".

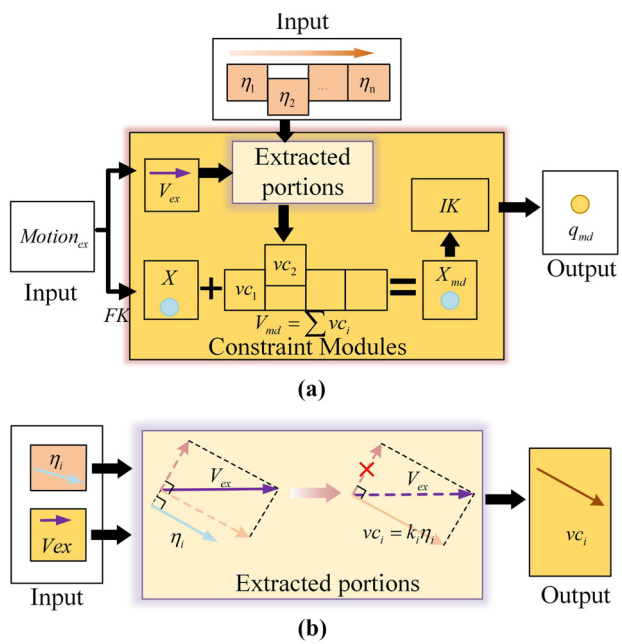
Each time the random tree is expanded, the planner finds the nearest node $Node_{near}$ in the random tree and expands it by one step d to the random point. Then, one expansion movement of the random tree $Motion_{extend}$ can be defined by $Node_{near}$ and d:

$$Motion_{extend} = \{Node_{near}, d\} \tag{6}$$

The constraint module gradually corrects $Motion_{extend}$ by sequentially incorporating the constraint primitives to ensure that the nodes obtained each time satisfy the constraints, as shown in Figure 4.

Next, we describe how the constraint modules operates during an expansion. Firstly, when d is small enough, $Motion_{extend}$ is denoted as a single differential motion of the robot from the occurrence. Then, $Motion_{extend}$ is projected into the workspace by robot kinematics to obtain X and V_{extend}. X denotes the position of the end-effector in the workspace when the robot is in $Node_{near}$ and V_{extend} is a $6D$ vector denoting the velocity of the end-effector at that time.

Figure 4 Schematic diagram of the computational process of the constraint module



Notes: (a) Calculation process of the constraint module; (b) extracted portions calculation process
Source: Authors' own

The components $k_i\eta_i$ that satisfy the constraints are gradually extracted from ve by *EXTRACTEDPORTIONS*. Finally, finally V_{new} is obtained. We can compute the corrected position X_{new} reached by the end-effector by using V_{new} and X and a potential new node q_{new} can be obtained by solving the inverse solution (*IK*) of robot kinematics. The pseudocode for the constraint module is presented in *Algorithm* 2.

During the constraint process, the constraint primitives are sequentially applied to *EXTRACTEDPORTIONS*, and the valid components are gradually extracted from V_{extend}. Taking an arbitrary constraint $\boldsymbol{\eta}$ as an example, the first constraint primitive that needs to be included is η_1. We need to extract the portion of V_{extend} that satisfies the constraint we need. Then, projecting V_{extend} on η_1 yields vc_1:

$$vc_1 = \frac{V_{extend} \cdot \eta_1}{|\eta_1|}\eta_1 = k_1\eta_1 \tag{7}$$

vc_1 is satisfied, so the process is considered as an extraction of the effective components. Then, we can obtain the remaining components $V_{nc} = V_{extend} - vc_1$. We need to extract the portion from V_{nc} that is linearly related to the next constrained primitive. After $i-1$, valid components $V_{nc},\ \eta_i$ are obtained:

$$V_{nc} = V_{extend} - k_1\eta_1 \cdots - k_{i-1}\eta_{i-1} \tag{8}$$

Given that each primitive in $\boldsymbol{\eta}$ is linearly independent of each other, then vc_i:

$$vc_i = \frac{V_{nc} \cdot \eta_i}{|\eta_i|}\eta_i = \frac{V_{extend} \cdot \eta_i}{|\eta_i|}\eta_i \tag{9}$$

Therefore, to extract the effective components step by step, project V_{extend} onto each constraint primitive successively. Then superposition of the effective components obtained each time can be obtained:

$$V_{new} = vc_1 + vc_2 \cdots + vc_n \tag{10}$$

Then, at this point:

$$V_{nc} = V_{extend} - vc_1 \cdots - vc_n \tag{11}$$

V_{nc} is orthogonal to each primitive element in η. Then, it is guaranteed that all the components in V_{extend} that satisfy the constraints have been extracted. Then, it is guaranteed that all constraint-satisfying components in V_{extend} have been extracted. This concludes the process of extracting constraint-satisfying components by introducing constraint primitives sequentially. It is not difficult to verify the uniqueness of V_{new}, which also means that there will not be multiple solutions to add extra computational overhead.

The neighbourhood linear approximation used to incorporate nonlinear constraints introduces constraint errors. To avoid the accumulation of constraint errors, the concept of tolerance error is introduced here. When the constraint error exceeds the threshold value, that sampling is considered a failure. This way, the error is limited to the desired range.

Algorithm 2: ConstraintModule

Input: $Motion_{extend}, \boldsymbol{\eta}$
Output: N_{new}
$[v, Node_{near}, d_0] \leftarrow Motion_{extend}$;
$V_{extend} = \text{Jacobi}(Node_{near}) * V$;
$X_{near} \leftarrow \text{FK}(Node_{near})$;
$m \leftarrow \text{Size}(\boldsymbol{\eta})$;
for $1 \rightarrow m$ **do**
  $vc_i \leftarrow \text{ExtrctedPosition}(V_{extend}, \eta_i)$;
  $V_{new} = \sum vc_i$;
end
$X_{new} \leftarrow \text{Sum}(X, V_{new})$;
$q_{new} \leftarrow \text{IK}(X_{new}, Node_{near})$;
if $\text{ValidTest}(q_{new}, V_{new})$ **then**
  return q_{new};
end
else
  return $q_{new} \leftarrow Null$;
end

5. Simulation

We conducted several simulation experiments to validate our approach. Our main objectives were: to test the efficiency of the method, to test the ability of the method to generalize to different end-effector constraints and to test the compatibility of the method with different planning algorithms.

The experimental robot system (DOG) consists of a six-jointed robotic arm mounted on a quadruped mobile platform. The test requires the system to accomplish a series of operational tasks corresponding to different end-effector constraints.

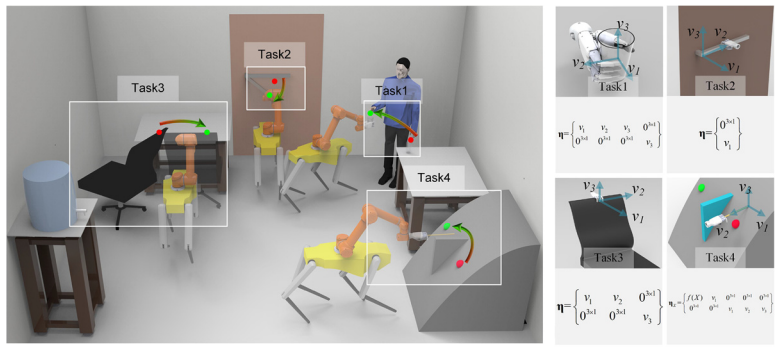
Here, the planner does not generate motion for the moving platform. The legged robot used in the simulation only demonstrates the promising applications of the framework. The quadruped platform has been positioned at the desired location by default to simplify the experiment. The planner generates the desired motions for the six-jointed robotic arm to accomplish the tasks. All planners were developed based on OMPL (Yakey *et al.*, 2001). All simulation experiments were conducted on the same workstation with an Intel processor and using identical planner parameters. The workstations were equipped with a CoreTM i7-8700K processor and 16 GB 2400 MHz DDR4 RAM.

5.1 Efficiency and adaptability test

To test the performance of the TC-Framework, we compare it with the constraint methods Atlas and Projection (TC-Planner, Atlas-Planner and Bi-Planner). Each of the three planners is designed specifically for Task1, and the generalization ability of the three planners is evaluated by observing their performance when applied to other tasks.

Firstly, we set up four operational tasks with a single constraint. The service robot needs to assist humans in completing each of the four operational tasks, as shown in Figure 5(a). The three end-effector constraints corresponding to these four tasks are shown in Figure 5(b). In Task 4, the constraint primitives need to be dynamically input due to the nonlinear constraints. Therefore,

Figure 5 Four types of single-constraint operational tasks and the corresponding four types of constraint expressions



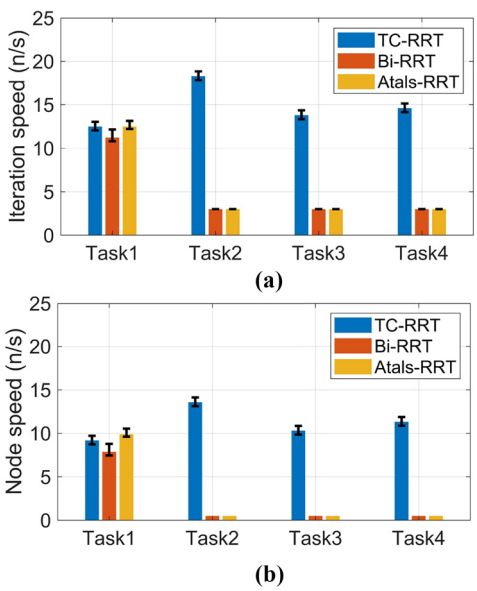
(a) **(b)**

Notes: (a) Schematic diagram of the four types of operational tasks. In Task 1, DOG needs to hand the water cup to its owner. In Task 2, DOG needs to push the chair to the target area while avoiding other obstacles. In Task 3, the DOG is asked to operate a door handle. In Task 4, DOG was asked to connect two points on a surface with a paintbrush; (b) description of the corresponding constraints for the four tasks
Source: Authors' own

we construct a constraint primitive generation module specifically for it in our experiments.

The three planners in the test are not modified in any way. By changing the input constraint primitives, the TC-RRT can adapt to different tasks. In contrast, the other planners lack the ability to change constraints. Each of the three planners was run independently 20 times for these single constraint tasks. The experimental data are shown in Figure 6. We recorded the iteration speed of each planner during the run. In Task 1, the iteration speed of TC-RRT based on TC-Framework outperforms the other two planners. However, when expanding to other tasks, the iteration speed of TC-RRT fluctuates significantly, indicating that the performance of the planner changes for different planning tasks. While the performance of Atlas-RRT and Bi-RRT sharply declines when expanding to other tasks, their iteration speed stabilizes at 3.6 times per second. This is because both methods are unable to find an efficient configuration in the face of other constraints, resulting in reaching the maximum number of iterations limit for each expansion. Figure 6(b) records the speed of acquiring valid nodes for each planner. In Task 1, the fastest planner to acquire valid nodes is TC-Planner, followed by Atlas-Planner. This indicates that the iteration speed of a planner affects its speed of acquiring valid nodes. This is because the planner that carries out higher frequency exploration has a high probability of obtaining valid nodes. However, when expanding to other tasks, the speed of obtaining valid nodes for Atlas-Planner and Bi-Planner is 0, and the planners are unable to obtain valid nodes when performing iterative computations. This indicates that these two planners cannot be expanded to other tasks.

Figure 6 Simulation results of three planners in different tasks



Notes: (a) Comparison of iteration speeds; (b) comparison of speed of obtaining valid nodes
Source: Authors' own

The planning success rate data recorded in Table 1 also reflects that only TC-RRT can be expanded to other tasks without

Table 1 Experimental result of efficiency and adaptability test

Task	Planer	Time	Success
Task 1	TC-RRT	92.251	*20/20*
	Bi-RRT	102.347	*20/20*
	Atlas-RRT	89.891	*20/20*
Task 2	TC-RRT	21.245	*20/20*
	Bi-RRT		*0/20*
	Atlas-RRT		*0/20*
Task 3	TC-RRT	21.168	*20/20*
	Bi-RRT		*0/20*
	Atlas-RRT		*0/20*
Task 4	TC-RRT	47.277	*20/20*
	Bi-RRT		*0/20*
	Atlas-RRT		*0/20*

Source: Authors' own

modification. This test indicates that TC-RRT based on TC-Framework can adapt to different end-effector constraints and also ensures certain advantages in performance.

5.2 Compatibility test of different planning algorithms

The selection of planning methodologies in the TC-Framework is independent, with three distinct planning methodologies, namely, RRT, LazyRRT and RRT-Connect, being established separately within the framework. The performance of the three planners in different tasks was observed to assess the compatibility of the framework with different planning approaches.

Robots frequently encounter tasks that impose mixed end-effector constraints. Accordingly, we set up a more complex task. In Figure 7(a), the robot needs to accomplish four subtasks sequentially to remove a test tube from a thermostat, and the planner needs to generate a corresponding sequence of operations. In Figure 7(b), the four subtasks correspond to different end-effector constraints.

The three planners are separately used to compute the desired sequence of actions for the task. Planning for the four subtasks was sequentially executed. Consequently, the failure of any sub-task results in the termination of that planning task. The experimental data for the 20 independent runs are presented in Figure 8 and Table 2. According to Figure 8(a), the time consumed for each switching task is 0.24*s*, indicating that the planner can quickly respond to changes in constraint conditions. Based on the stability of the planner, as shown in Figure 8(b), the speed of obtaining effective nodes remains stable in the four subtasks. However, the iteration speed of the planner varies even more than 10%, but its data are still concentrated between −5% and 5% overall.

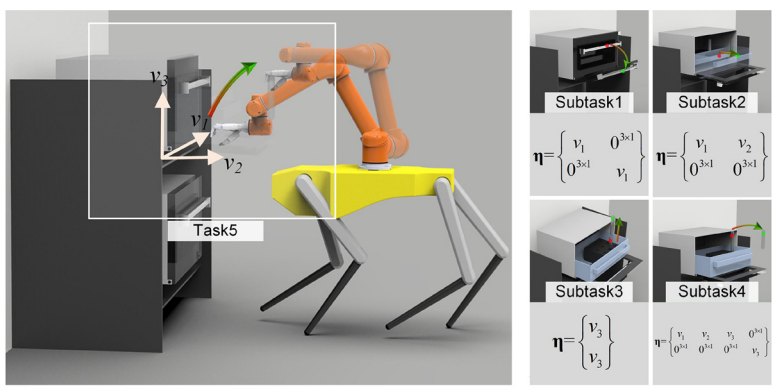
Observation of Table 2 reveals that Planner 2, with the lowest solution time, also exhibits the fastest overall iteration speed. However, when comparing the performance of the same planner across different tasks, it becomes apparent that in Subtask 4, which has the highest degree of constraint freedom, each planner instead exhibits the slowest iteration speed but achieves the fastest acquisition of effective nodes. This is because the iteration speed of the algorithm is influenced by various factors such as collision detection efficiency, and the state of the target and initial points. A higher degree of freedom in constraints ensures a greater likelihood of obtaining an effective node per iteration. Therefore, the probability of obtaining a valid node per iteration is further calculated, and the data in Table 2 corroborate this observation.

The TC-Framework-based planner underwent simulation testing, during which it was demonstrated that it is capable of adapting to various end-effector constraints. In contrast to planners that have been developed for specific constraints, the TC-Framework-based planner is capable of seamlessly handling different end-effector constraints by simply switching inputs. Moreover, planners based on different planning methods demonstrate consistent performance under diverse constraints, thereby illustrating the framework's compatibility with a range of planning algorithms. The framework's ability to decouple constraint modelling from planner design provides greater flexibility in the design of robotic systems, even when the inputs to the constraints are predefined in tests.

6. Experiments

This section outlines the deployment of a simplified autonomous operating system based on the TC-Framework on a physical experimental system. The practical value of the system is demonstrated through the operation of the robot under various end-effector constraint tasks. To detect errors in the execution of constraints, it is crucial to keep the base stationary during the manipulator moving to minimize sources of error. Hence, a wheeled robot, chosen for its higher

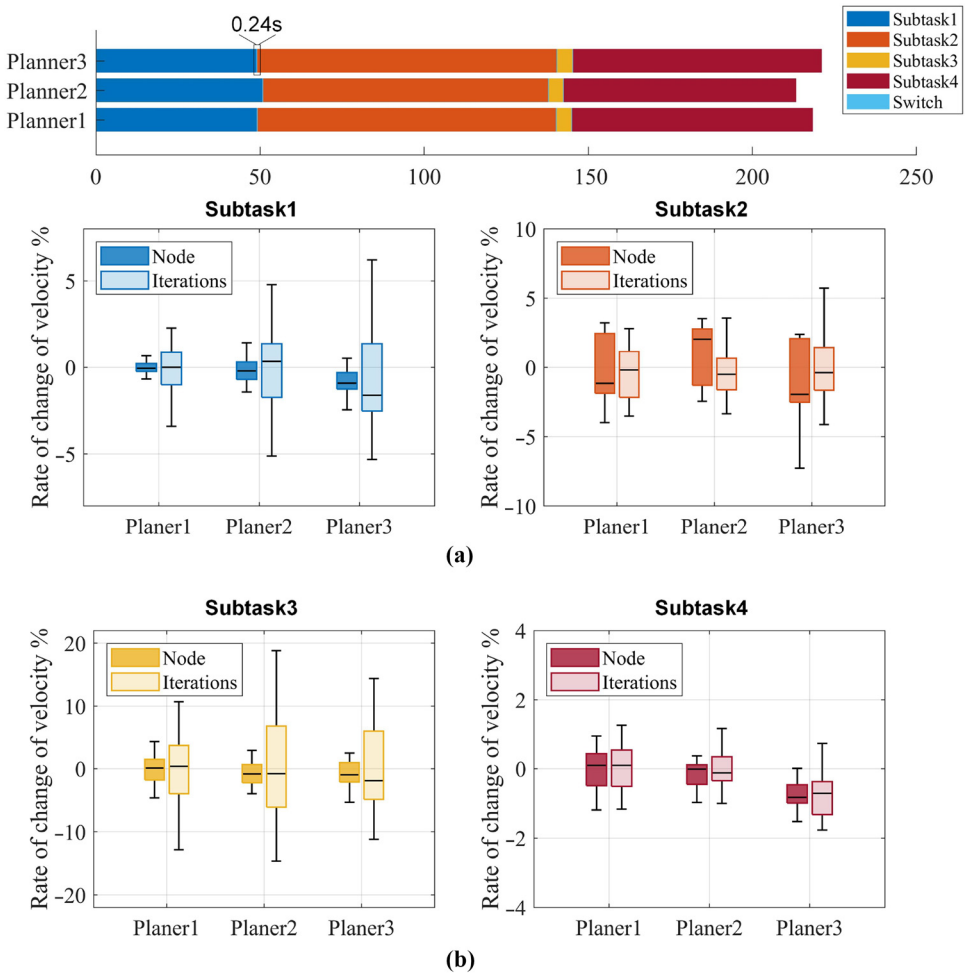
Figure 7 Operational tasks with mixed constraints and constraint representations corresponding to the four subtasks



(a) **(b)**

Notes: (a) The DOG needs to complete four subtasks in sequence to remove the test tube with reagents from the thermostat box. Subtask 1, open the lid of the thermostat along the hinge. Subtask 2, pull out the drawer containing the test tube cups. Subtask 3, remove the test tube upwards. Subtask 3, remove the test tube outward; (b) description of the corresponding constraints for the four subtasks
Source: Authors' own

Figure 8 Experimental results for the three planners in Task 5 (Planner 1 – RRT, Planner 2 – LazyRRT and Planner 3 – RRTConnect)



Notes: (a) Average planning time for the three planners. The time consumed to switch tasks is extremely short, reflecting the speed with which the constraint module can respond to constraints; (b) the magnitude of the fluctuations in the iteration speed and the speed of getting valid nodes are counted using the average value of Planner 1 in each subtask as a benchmark. The graph reflects whether the constraint module stably runs or not
Source: Authors' own

Table 2 Experimental result of compatibility test

Planner	Task	Iterations	Rate	Success
Planner 1	SubTask1	831 ± 334.0	0.650 ± 0.024	*20/20*
	SubTask2	1199.5 ± 125.5	0.831 ± 0.048	
	SubTask3	69.0 ± 10.0	0.742 ± 0.095	
	SubTask4	826 ± 114	0.986 ± 0.003	
Planner 2	SubTask1	838 ± 267	0.646 ± 0.040	*20/20*
	SubTask2	1214.5 ± 74.5	0.852 ± 0.025	
	SubTask3	70 ± 29.0	0.738 ± 0.144	
	SubTask4	835.5 ± 108.5	0.985 ± 0.004	
Planner 3	SubTask1	858 ± 182	0.655 ± 0.034	*20/20*
	SubTask2	994.5 ± 73.5	0.818 ± 0.062	
	SubTask3	51 ± 6	0.758 ± 0.056	
	SubTask4	812.5 ± 118.5	0.984 ± 0.009	

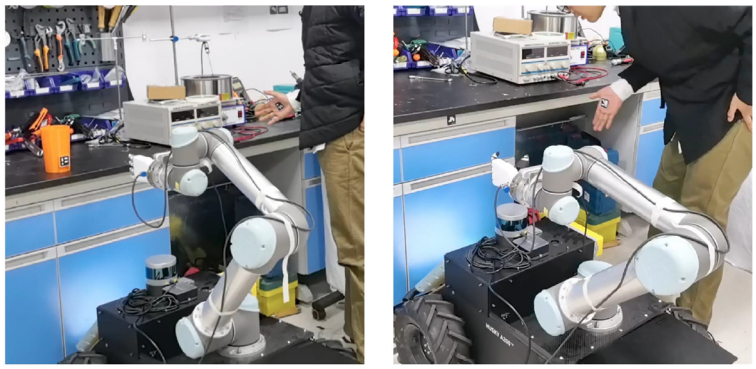
Source: Authors' own

stability compared to legged robots, was used in the physics experiments.

The two operational tasks presented in Figure 9 involve moving a beaker filled with chemicals to the experimenter and opening a drawer for the experimenter. The end-effector of the robot has specific trajectory constraints due to the avoidance of chemical spills and the properties of the drawer. To address these constraints, we implemented an autonomous operating system based on the TC-framework and deployed it on a real robot. As shown in Figure 10, the system comprises four main modules:

1 *Perception system*: obtains job task information through the vision system, including job target location, unprocessed constraint information and environmental information (Wang *et al.*, 2023). Here, this function is realized through AprilTag (Wang and Olson, 2016). The task to be performed is identified by the tag ID affixed to the operator's hand. The posture of the target

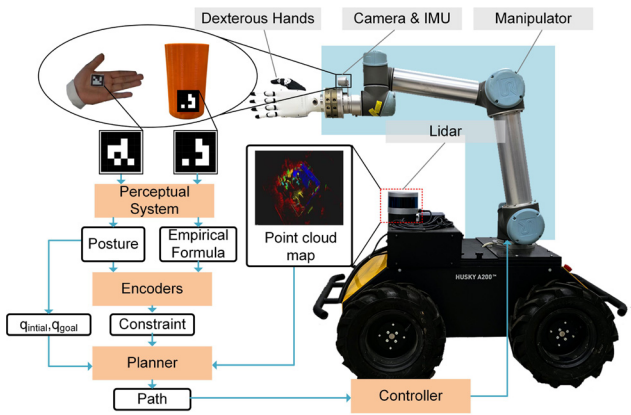
Figure 9 Two operational tasks



(a) **(b)**

Notes: (a) Moving beaker; (b) opening drawer
Source: Authors' own

Figure 10 Autonomous operational systems based on the TC-Framework



Source: Authors'own

object is identified by the tag affixed to the operation object.

2 *Encoder*: encode the constraint information into planner receivable importation. In the experiments, it is the encoding of the positional information of the target object into constraint primitives $\boldsymbol{\eta}$ by the empirical formula *Fml* (*Posture*) corresponding to the task. Here, empirical formulas are predetermined through *a priori* information. The appropriate empirical formula is selected by sensing the obtained task ID.

3 *Planner*: through the obtained task information, generate a collision-free trajectory for the job task that satisfies the constraints inputted by Encoder. In the experiments, the planner is based on the TC-Framework composition.

4 *Controller*: sends the generated collision-free trajectory to the robot for execution.

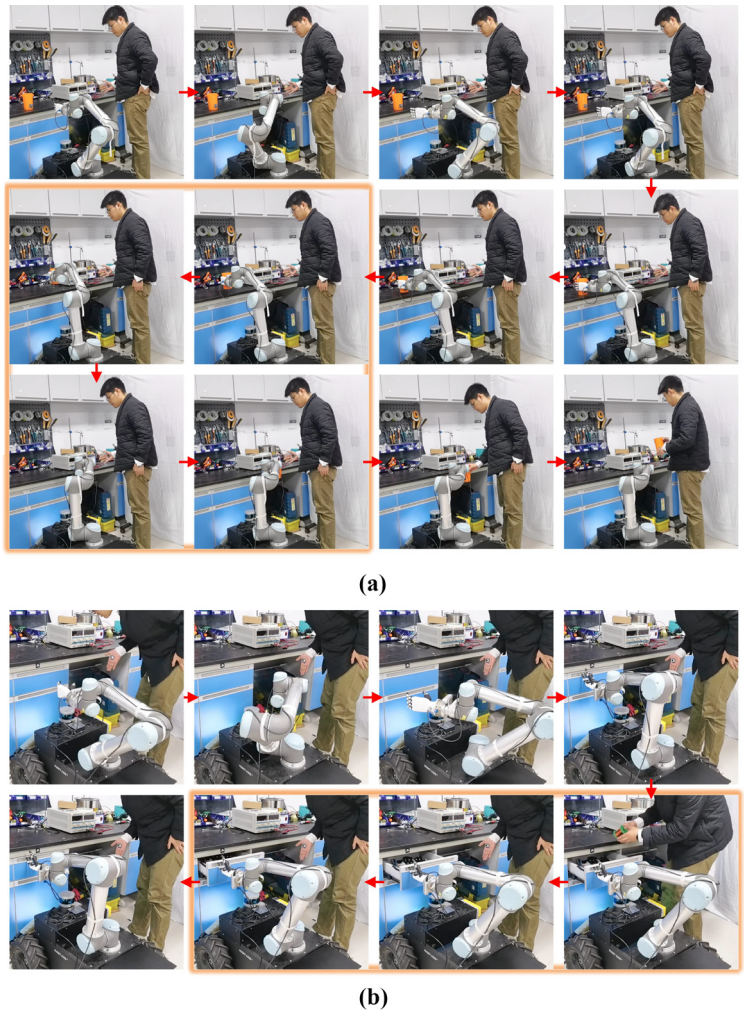
The robot moves randomly in the operation area and scans the environment for modelling. The position of the operating object is determined by the Perception system. Subsequently, this information is encoded into constraint primitives by the Encoder and incorporated into the planner through the constraint module. The planner generates the desired trajectory, connecting the initial and target points obtained by the Perception system. Subsequently, the trajectory is smoothed and transmitted to the controller for execution. Figure 11 shows the operation of the robot system for the two tasks. To ensure compatibility with both tasks, we assign the corresponding Ful to the constraints of each task and store them in AprilTag. The corresponding Ful can be obtained by identifying the AprilTag associated with each task. The position of the operating object and the Ful integrated into the Encoder for the two tasks are shown in Figure 12 and equation (12).

$$\boldsymbol{\eta} = Fml_{Task_1}(Posture) = \begin{bmatrix} r_x & r_y & r_z & 0^{3\times1} \\ 0^{3\times1} & 0^{3\times1} & 0^{3\times1} & r_z \end{bmatrix}$$

$$\boldsymbol{\eta} = Fml_{Task_2}(Posture) = \begin{bmatrix} r_x & r_y \\ 0^{3\times1} & 0^{3\times1} \end{bmatrix}. \quad (12)$$

The robot trajectories for the two tasks are displayed in Figure 13. Figure 13 demonstrates the changes in the

Figure 11 Process of running a robot that accomplishes the task of obtaining containers



Notes: (a) Task 1: moving beaker; (b) Task 2: opening drawer Marked through the orange highlights is the process of running the planning results
Source: Authors' own

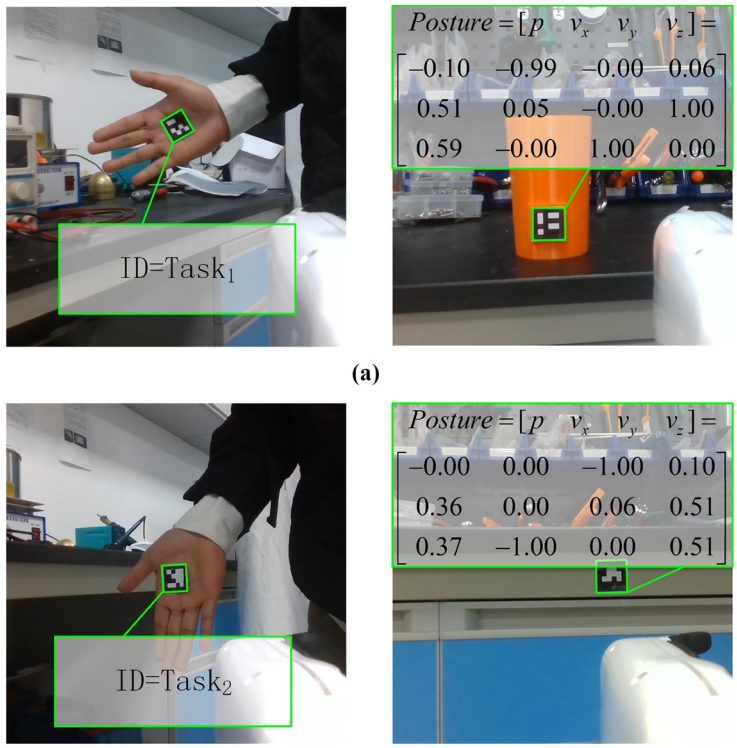
position and alpha value of the robot's end-effector as it picks up the beaker and moves it to the operator in Task 1. However, the gamma and beta values remained unchanged as we restricted any movement that could cause the beaker to tip. The autonomous system encoded the perception system data through encoding and incorporated the constraint into the planner. In Task 2, the rotation and vertical movements of the drawer were limited to facilitate pulling it. Figure 13 illustrates that the motion satisfies the constraint while pulling the drawer.

The experimental results demonstrate the feasibility of deploying an autonomous operating system based on the TC framework onto a field robot. This system effectively encodes object postures obtained from the Perception system into η inputs for the planner using an Encoder. Consequently, the robot can adapt to different end-effector constraints and autonomously perform various operational tasks efficiently. Planner pairs based on the TC-Framework do not require specialized modifications when extended to other tasks. Only the configuration of new labels and their corresponding constraint matrices is necessary. While designing the current constraint functions demands prior information about the tasks, creating the functional utility for the tasks entails much less effort than developing a dedicated planner for each task. This reduces the workload of deploying robots across diverse application domains. As AI comprehension capabilities advance, the Encoder is expected to integrate into the Perception system, generating η inputs directly for the planner.

7. Conclusion

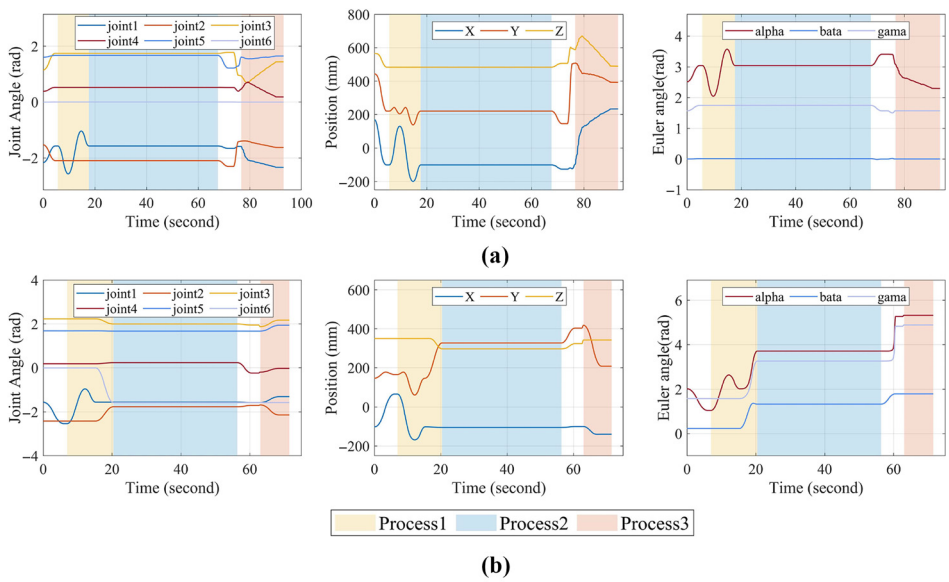
This paper proposes an object-oriented planner framework to address the adaptability limitations of robot motion planners under the diversity of end-effector constraints. The standard sampling-based planner framework is extended with a unified constraint input interface and constraint module. This provides a generic planner framework that allows the planner to respond

Figure 12 AprilTag that records task and constraint information and the position of the operational target



Notes: (a) Task1: moving beaker; (b) Task2: opening drawer
Source: Authors' own

Figure 13 Robot's joint angles, end-effector positions and Euler angle changes recorded during the completion of the two tasks



Notes: (a) Data recorded during Task 1; (b) data recorded in Task 2
Process1 is the process of scanning the environment and capturing the April tags; Process2 is the process of running the perception system and planner; Process3 is the process of the robot executing the planning result
Source: Authors' own

to different end-effector constraints imposed by different operational tasks.

The framework is tested under different end-effector constraints to verify its generalizability. The test results indicate that the planner, which is based on the TC planner framework, can accommodate various end effector constraints without requiring any adjustments. The adaptability of the TC framework to different planning methods is also tested. The planners based on different planning methods perform stably under different constraints, reflecting the compatibility of the framework with planning algorithms.

Tests on a physical experimental system show that the framework enables the robot to respond to a variety of operational tasks without the need to modify the planner. Although the Encoder (*FUL*) must be designed for various tasks, the process is decoupled from the design of the robot. Robotic systems can be designed modularly, which reduces the difficulty of deploying robots in the field. And, having identified the inputs of the image and the outputs of the constraint primitives, then future work is to implement an automated constraint acquisition module based on the AI model. This will lead to the creation of an end-to-end motion planning framework from the image to the path.

References

Bonalli, R., Bylard, A., Cauligi, A., Lew, T. and Pavone, M. (2019), "Trajectory optimization on manifolds: a theoretically-guaranteed embedded sequential convex programming approach", ArXiv, abs/1905.07654.

Bonilla, M., Pallottino, L. and Bicchi, A. (2017), "Noninteracting constrained motion planning and control for robot manipulators", *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4038-4043.

Bordalba, R., Schoels, T., Ros, L., Porta, J.M. and Diehl, M. (2023), "Direct collocation methods for trajectory optimization in constrained robotic systems", *IEEE Transactions on Robotics*, Vol. 39 No. 1, pp. 183-202.

Brock, O. and Khatib, O. (2002), "Elastic strips: a framework for motion generation in human environments", *The International Journal of Robotics Research*, Vol. 21 No. 12, pp. 1031-1052.

Burger, B., Maffettone, P.M., Gusev, V.V., Aitchison, C.M., Bai, Y., Wang, X., Li, X., Alston, B.M., Li, B., Clowes, R., Rankin, N., Harris, B., Sprick, R.S. and Cooper, A.I. (2020), "A mobile robotic chemist", *Nature*, Vol. 583 No. 7815, pp. 237-241.

Foumani, M., Gunawan, I., Smith-Miles, K. and Ibrahim, M. Y. (2015), "Notes on feasibility and optimality conditions of small-scale multifunction robotic cell scheduling problems with pickup restrictions", *IEEE Transactions on Industrial Informatics*, Vol. 11 No. 3, pp. 821-829.

Fusco, F., Kermorgant, O. and Martinet, P. (2019), "Improving relaxation-based constrained path planning via quadratic programming", in Strand, M., Dillmann, R., Menegatti, E. and Ghidoni, S. (Eds), *Intelligent Autonomous Systems 15*, pp. 15-26, Springer International Publishing, Cham.

Jaillet, L. and Porta, J.M. (2013), "Path planning under kinematic constraints by rapidly exploring manifolds", *IEEE Transactions on Robotics*, Vol. 29 No. 1, pp. 105-117.

Khatib, O. (1987), "A unified approach for motion and force control of robot manipulators: the operational space formulation", *IEEE Journal on Robotics and Automation*, Vol. 3 No. 1, pp. 43-53.

Kingston, Z. and Kavraki, L.E. (2023), "Scaling multimodal planning: using experience and informing discrete search", *IEEE Transactions on Robotics*, Vol. 39 No. 1, pp. 128-146.

Kingston, Z., Moll, M. and Kavraki, L.E. (2018), "Sampling-based methods for motion planning with constraints", *Annual Review of Control, Robotics, and Autonomous Systems*, Vol. 1 No. 1, pp. 159-185.

Kingston, Z., Moll, M. and Kavraki, L.E. (2019), "Exploring implicit spaces for constrained sampling-based planning", *The International Journal of Robotics Research*, Vol. 38 Nos 10/11, pp. 1151-1178.

Kolakowska, E., Smith, S.F. and Kristiansen, M. (2014), "Constraint optimization model of a scheduling problem for a robotic arm in automatic systems", *Robotics and Autonomous Systems*, Vol. 62 No. 2, pp. 267-280.

Lamiraux, F. and Mirabel, J. (2022), "Prehensile manipulation planning: modeling, algorithms and implementation", *IEEE Transactions on Robotics*, Vol. 38 No. 4, pp. 2370-2388.

Liu, Z., Liu, Q., Xu, W., Wang, L. and Zhou, Z. (2022), "Robot learning towards smart robotic manufacturing: a review", *Robotics and Computer-Integrated Manufacturing*, Vol. 77, p. 102360.

Miao, R., Jia, Q. and Sun, F. (2023), "Long-term robot manipulation task planning with scene graph and semantic knowledge", *Robotic Intelligence and Automation*, Vol. 43 No. 1, pp. 12-22.

Mitchell, J.S.B., Mount, D.M. and Papadimitriou, C.H. (1987), "The discrete geodesic problem", *SIAM Journal on Computing*, Vol. 16 No. 4, pp. 647-668.

Nakatsuru, K., Wan, W. and Harada, K. (2023), "Implicit contact-rich manipulation planning for a manipulator with insufficient payload", *Robotic Intelligence and Automation*, Vol. 43 No. 4, pp. 394-405.

Palleschi, A., Pollayil, G.J., Pollayil, M.J., Garabini, M. and Pallottino, L. (2022), "High-level planning for object manipulation with multi heterogeneous robots in shared environments", *IEEE Robotics and Automation Letters*, Vol. 7 No. 2, pp. 3138-3145.

Park, S., Lee, H., Kim, S.-G., Baek, J., Jang, K., Kim, H.C., Kim, M. and Park, J. (2022), "Robotic furniture assembly: task abstraction, motion planning, and control", *Intelligent Service Robotics*, Vol. 15 No. 4, pp. 441-457.

Phoon, M.S., Schmitt, P.S. and Wichert, G.V. (2022), "Constraint-based task specification and trajectory optimization for sequential manipulation", *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 197-202.

Qureshi, A.H., Dong, J., Baig, A. and Yip, M.C. (2022), "Constrained motion planning networks x", *IEEE Transactions on Robotics*, Vol. 38 No. 2, pp. 868-886.

Rakita, D., Mutlu, B. and Gleicher, M. (2020), "An analysis of RelaxedIK: an optimization-based framework for generating

accurate and feasible robot arm motions", *Autonomous Robots*, Vol. 44 No. 7, pp. 1341-1358.

Secil, S. and Ozkan, M. (2023), "A collision-free path planning method for industrial robot manipulators considering safe human–robot interaction", *Intelligent Service Robotics*, Vol. 16 No. 3, pp. 323-359.

Stilman, M. (2010), "Global manipulation planning in robot joint space with task constraints", *IEEE Transactions on Robotics*, Vol. 26 No. 3, pp. 576-584.

Wang, J. and Olson, E. (2016), "AprilTag 2: efficient and robust fiducial detection", *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4193-4198.

Wang, Y., Li, Z., Li, Y. and Dong, E. (2024), "An efficient constraint method for solving planning problems under end-effector constraints", *Industrial Robot: The International Journal of Robotics Research and Application*, Vol. 51 No. 3.

Wang, Z., Garrett, C.R., Kaelbling, L.P. and Lozano-Pérez, T. (2021), "Learning compositional models of robot skills for task and motion planning", *The International Journal of Robotics Research*, Vol. 40 Nos 6/7, pp. 866-894.

Wang, Z., Zhou, X., Xu, C. and Gao, F. (2022), "Geometrically constrained trajectory optimization for multicopters", *IEEE Transactions on Robotics*, Vol. 38 No. 5, pp. 3259-3278.

Wang, Z., Liu, Y., Duan, Y., Li, X., Zhang, X., Ji, J., Dong, E. and Zhang, Y. (2023), "USTC FLICAR: a sensors fusion dataset of lidar-inertial-camera for heavy-duty autonomous aerial work robots", *The International Journal of Robotics Research*, Vol. 42 No. 11, pp. 1015-1047.

Yakey, J., LaValle, S. and Kavraki, L. (2001), "Randomized path planning for linkages with closed kinematic chains", *IEEE Transactions on Robotics and Automation*, Vol. 17 No. 6, pp. 951-958.

Yao, Z. and Gupta, K. (2005), "Path planning with general end-effector constraints: using task space to guide configuration space search", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1875-1880.

Zucker, M., Ratliff, N., Dragan, A.D., Pivtoraiko, M., Klingensmith, M., Dellin, C.M., Bagnell, J.A. and Srinivasa, S.S. (2013), "Chomp: covariant Hamiltonian optimization for motion planning", *The International Journal of Robotics Research*, Vol. 32 Nos 9/10, pp. 1164-1193.

Corresponding author

Erbao Dong can be contacted at: ebdong@ustc.edu.cn