IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 August 2023, accepted 2 September 2023, date of publication 8 September 2023, date of current version 13 September 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3313173

RESEARCH ARTICLE

LAC-RRT: Constrained Rapidly-Exploring Random Tree With Configuration Transfer Models for Motion Planning

CHI-KAI HO AND CHUNG-TA KING
Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan
Corresponding author: Chi-Kai Ho (s104062580@m104.nthu.edu.tw)

ABSTRACT Motion planning is challenging for robotic manipulators. Achieving high efficiency and generalization while considering various constraints simultaneously can be difficult. One common solution is to use iterative projection techniques to acquire feasible manipulator configurations and then search for a valid path to reach the goal state based on those configurations. However, such iterative techniques tend to be computationally expensive and time-consuming. The problem becomes more serious if equality constraints are involved, due to their narrower solution space. In this paper, we propose the Learning-Assisted Constrained Rapidly-Exploring Random Tree (LAC-RRT) algorithm, which employs self-supervised learning to train a model that can directly convert any sampled configuration to a new and valid configuration using feature values constrained by the imposed equality constraints, avoiding the need for iterative optimizations. Unlike other learning-based motion planning techniques, which typically solve the problem by building the constraint manifold based on a fixed set of constraints, LAC-RRT permits better generalization by allowing the equality constraints to be specified at run time. The experimental results show that the proposed LAC-RRT surpasses other approaches in most cases. Specifically, LAC-RRT can significantly reduce computation time by 80-90% for acquiring valid configurations and performing motion planning.

INDEX TERMS Constrained motion planning, self-supervised learning, constraint manifold.

I. INTRODUCTION

Motion planning is crucial in controlling the motion of a robotic manipulator, which involves finding a sequence of actions that enables the manipulator to transition from its current state to a desired state. The resultant motion plan typically needs to satisfy requirements of the given task and the restrictions imposed by the environment, such as avoiding obstacles, maintaining end effector orientation, or staying in contact with an object. The problem is normally formulated as an optimization problem that searches for a feasible and/or optimal path in the *configuration space* that satisfies the imposed constraints. Configuration space refers to the space of joint states in the robotic manipulator and the subspace that satisfies the constraints is called the *constraint manifold* [15].

The associate editor coordinating the review of this manuscript and approving it for publication was Guangcun Shan.

For robotic manipulators with a high degree of freedom (DoF), the computation complexity in finding a feasible solution in the configuration space is very high. Sampling-based motion planning, e.g., Rapidly-exploring Random Tree (RRT), is often the preferred solution due to its efficiency in high dimensional space [1], [2], [8], [17], [20].

Constraints in motion planning can be categorized as inequality constraints, e.g., obstacle avoidance, and equality constraints, e.g., maintaining a given orientation. Inequality constraints generally have a large constraint manifold because of the amount of configurations that can satisfy the constraints. Hence, to handle inequality constraints, the most common approach is to use *rejection sampling* [5], which involves generating a large number of samples and discarding those that fail to meet the constraints. However, rejection sampling is not suitable for solving problems with equality constraints, because it is unlikely to obtain samples on

low-dimensional constraint manifolds, such as a line, through repeated sampling.

To handle equality constraints, projection strategies, such as gradient descent projection and pseudo-inverse Jacobian projection, have been studied extensively [1], [2], [10], [21]. However, these methods may be inefficient because of the large number of iterations required to project the sampled configuration onto the constraint manifold [16]. Alternatively, direct sampling strategies may be exploited, which learn the constraint manifold offline and then sample from the manifold during online planning. For example, Şucan and Chitta [17] utilized the approximation graph to store the precomputed valid configurations. Recently, deep neural networks are employed to learn the constraint manifold for better generalization. Sutanto et al. [18] trained constraint-manifold networks with different loss functions to describe motion constraints, while Lembono et al. [11] used generative adversarial networks to learn the constraint manifold.

Although direct sampling strategies accelerate the planning process by precomputing the constraint manifold, they typically require ground-truth data for training, which may be generated from existing planners. This limits their performance to the quality of the planners. Furthermore, changes in the environment or constraints require the training data recollected and the model retrained. They cannot handle new or different constraints during online inference. From the above discussions, we can see that an ideal strategy for robotic motion planning should efficiently handle constraints while generalize well to new constraints at run time.

To meet the above requirement and address equality constraints, this paper introduces a novel motion planning approach, called *Learning-Assisted Constrained Rapidly-Exploring Random Tree* (LAC-RRT). It is based on the key insight that equality constraints of motion planning typically impose restrictions on ''features'' of the manipulator, such as end-effector orientation, wrist position, the angle between the plane of the elbow joint and the vertical plane, etc. Thus, handling equality constraints in the feature space may be more straightforward and efficient than in the configuration space. What is needed is to transform a sampled configuration during the random tree exploration into the corresponding feature embedding, adjust the feature embedding according to the equality constraints, and then map it back to get a valid configuration. To achieve these, this paper proposes a self-supervised deep learning network, the *Configuration Transfer Model* (CTM).

CTM employs an encoder-decoder structure where the encoder maps a given configuration into the feature embedding and the decoder transforms it back. What is novel in CTM is a third module, called the *Custom Feature Composer* (CFC), which during training indicates the explicit features of the manipulator that will be constrained by the imposed equality constraints and what the values of these features are for this specific input configuration. CFC is used to force the decoder to learn not just an arbitrary transformation but a transformation that knows how to map an embedding with specific feature values back to a configuration. It follows that during inference, i.e., online path planning, we only need to provide the feature values imposed by the equality constraints to the decoder and it can generate a valid configuration from the embedding of a sampled configuration produced by the encoder.

By working in the feature space, the proposed LAC-RRT can directly transform any random configuration to a constraint-satisfying, valid configuration. This significantly accelerates the planning process compared to iterative projection methods. Unlike learning-based direct sampling, LAC-RRT provides the flexibility of evaluating equality constraints that constrain using different feature values during online planning. There is no need to recollect data or retrain the model.

Our experiments show that LAC-RRT requires only 1.4 - 3.1% of the computation time to generate a feasible configuration compared to conventional projection methods. In comparison to least squares optimization, our method requires only 4.3% and 14.8% of computation time to meet position and orientation constraints, respectively. Moreover, LAC-RRT needs only 20% time to meet orientation constraints compared to L-BFGS-B. When considering obstacle avoidance, it only needs about 10% computation time to plan a motion to meet orientation constraints.

The remainder of the paper is organized as follows. Section II surveys related works, and Section III presents the proposed method. Section IV shows our experiments and discusses the results. Conclusions are drawn in Section V.

II. RELATED WORK

The LAC-RRT algorithm is motivated by CBiRRT [1], [2], which employs iterative projection techniques to obtain suitable configurations on the constraint manifold for constrained motion planning. Other studies also embrace similar ideas, including Lavalle et al. [10], which uses randomized gradient descent to handle closed-chain kinematic constraints, and ATACE [21], which iteratively calculates projected nodes based on random configurations by randomized gradient descent. In a more general constrained motion planning context, Stilman [16] compares various projection techniques and finds that the pseudo-inverse projection method is more efficient in calculating valid configurations. Although these approaches offer promising solutions for constrained motion planning, the iterative process tends to be computationally expensive and time-consuming.

To address the efficiency problem, direct sampling strategies are proposed. These strategies model the constraint manifold offline and bypass the iterative process by sampling directly from the model during online planning. Şucan and Chitta [17] approximate the constraint manifold by computing valid configurations beforehand and storing them in a data structure for online sampling. To improve generalization,

recent studies have turned to deep learning techniques. Sutanto et al. [18] propose ECoMaNN to learn the manifold for equality constraints, while Lembono et al. [11] use generative adversarial networks (GANs) to learn the distribution of the constraint manifold and directly utilize the generated samples to avoid computations during planning. The problem with direct sampling strategies is that they require data recollection and model retraining if there are any changes to the environment or constraints. They cannot handle new or different constraints during online inference.

It can be seen that direct sampling is similar to iterative projection in that they both work in the configuration space. The difference is that direct sampling pre-evaluates the constraints and establishes the valid configurations offline to avoid online iterative computations, while iterative projection evaluates the constraints at run time and approximates its way to a valid configuration gradually. As a result, iterative projection can flexibly handle new constraints at run time, while direct sampling must stick to the constraints that it is trained for. In contrast, instead of learning the constraint manifold, LAC-RRT learns how to transform between configuration space and feature space. During the process, it changes the features of the input configuration to conform to the imposed equality constraints and then produces a valid configuration. It thus allows the direct transformation of any random configuration to an equality-constraint-satisfying configuration, while inequality constraints can be simply handled through rejection sampling. This significantly accelerates the planning process compared to iterative projection methods and provides the flexibility of evaluating new constraints at run time when compared to direct sampling.

In addition to sampling-based approaches, recent studies have considered reinforcement learning (RL) as a solution to the motion planning problem [3], [13]. RL involves training a policy network that takes observations as input and outputs the desired velocity of the actuators, enabling it to control robots in real-time. Unlike sampling-based approaches, RL efficiently handles dynamic obstacles without the need to check for collisions at each sampled joint configuration, making it more suitable for real-time applications. However, RL's training process can be time-consuming, and it may encounter convergence issues when dealing with complex problems. In this paper, our focus is specifically on solving problems involving static obstacles.

III. APPROACH

LAC-RRT is an extension of RRT-based algorithms [6], [7], [9], designed for constrained motion planning. By utilizing CTMs, LAC-RRT can efficiently generate configurations on the constrained manifold directly without a lengthy iterative process to search for valid configurations. The difference between prior approaches and LAC-RRT is illustrated in Fig. 1. The same CTM can handle the same type of constraints, which constrain on the same set of features, e.g., moving on a horizontal plane but varying in height. New constraints may be imposed at run time as long as they belong to the same type. For different types of equality constraints, different CTMs may need to be trained. It is also important to note that CTM is not specialized to any particular RRT approach. In this paper, we use the bi-directional RRT (BiRRT) as an example to illustrate the proposed method.

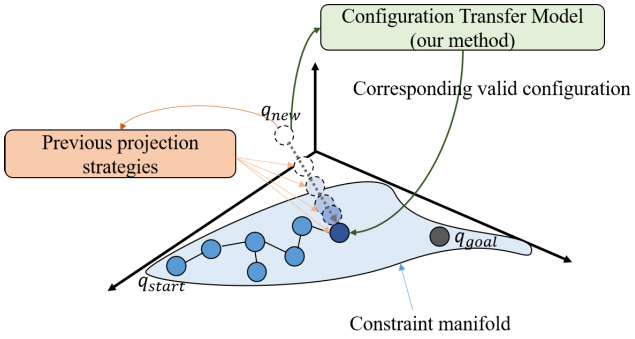

FIGURE 1. Constrained sampling-based methods aim to construct a path connecting the start configuration and the goal configuration on the constraint manifold by building a tree. To locate a node that satisfies the constraints for expanding the tree, prior methods utilize projection techniques, which usually involve iterative computation to move the projected configuration towards the constraint manifold. This paper introduces CTM, which directly transforms a sampled configuration to a valid configuration on the manifold based on the given constraints, thereby accelerating the planning process significantly.

A. PROBLEM DEFINITION

Consider a robotic manipulator with N-DoF, whose configuration space $\mathcal{C}$ is a N-dimensional joint space. Given a start configuration q_s and a goal configuration q_g, the goal of motion planning is to find a path on $\mathcal{C}$ connecting these two configurations. When considering constraints imposed by the environment and the task, such as obstacle avoidance and maintaining orientation, the valid configuration space is reduced to a low-dimensional constraint manifold. Obtaining valid configurations on this manifold is challenging, as resampling can be time-consuming and modeling the manifold may require recollection and retraining with every change in constraints. Our objective is to efficiently find a set of valid configurations on the constraint manifold that connects the start and goal configurations.

B. CONFIGURATION TRANSFER MODEL

The proposed CTM is a self-supervised learning model capable of directly generating a valid joint configuration satisfying specified constraints without the need for an iterative process. To convert a sampled configuration into a valid one, CTM separates the joint configuration into an implicit embedding part and an explicit vector part. Explicit features are the ones that need to be restricted, e.g., the orientation of the end-effector. A decoder then converts the long vector (obtained by concatenating the embedding and the vector) back to the configuration. Distinguishing partial explicit features allows us to adjust the values of the features based on application requirements, generating a valid corresponding configuration with desired feature values. As depicted in Fig. 2, CTM

comprises of three modules: an encoder, a decoder, and the Custom Feature Composer (CFC). The encoder converts the input configuration into a posture embedding, which captures the posture feature of the manipulator under the given configuration, whereas CFC calculates partial features of the manipulator using off-the-shelf methods, such as forward kinematics.

As mentioned above, CFC is a **user-defined problem-specific** module, which identifies the explicit features that will be subjected to equality constraints in online path planning, providing specific values for these features based on the input configuration. For instance, if the online path planning necessitates maintaining the orientation of the end-effector as an equality constraint, the explicit feature would be the end-effector orientation. In this case, a simple forward kinematics function can be employed within CFC to determine the end-effector orientation from the input configuration. Subsequently, the explicit features are concatenated with the embedding obtained from the encoder to train the decoder. In the end, the decoder learns not just an arbitrary transformation, but a transformation that can map an embedding with specific feature values back to a configuration. Note that since CFC does not require training, one may alternatively calculate the explicit features offline for better training efficiency.

During inference, i.e., online path planning, the encoder receives a sampled configuration and produces the corresponding feature embedding. The decoder then takes the embedding together with the desired values of the explicit features, e.g., a specific orientation of the end-effector, that are imposed by the equality constraints to produce a new configuration. The new configuration will be a valid configuration that satisfies the equality constraints. Finally, the decoder utilizes the posture embedding and the explicit features to produce a valid configuration.

CTM is more efficient than traditional projection methods in dealing with equality constraints, because it infers a valid configuration without iterations. However, CTM is limited to holonomic constraints, as it is designed as a transfer model that utilizes individual configurations and does not take into account velocities or accelerations.

C. LEARNING-ASSISTED CONSTRAINED RAPIDLY-EXPLORING RANDOM TREE

This section introduces the LAC-RRT algorithm, which achieves constrained motion planning using CTM. Although the pseudocode in Algorithm 1 is based on BiRRT, other RRT-based approaches may also be utilized. To begin, LAC-RRT initializes two trees (T_a and T_b) with the start and goal configurations. The algorithm samples a random configuration from the configuration space, and the NearestNeighborNode function identifies the closest node to the sampled configuration in T_a. The function ExtendCTM extends the tree toward the sampled configuration as closely as possible, while taking into account the desired constraints. This function returns the last achieved configuration, which has been added to the tree, while approaching the target configuration.

Algorithm 1 Constrained RRT With CTMs (BiRRT version)

Input: start configuration: q_s; goal configuration: q_g;
Output: a feasible path P on the manifold

1: T_a.Init(q_s); T_b.Init(q_g);
2: **while** iteration < MAX_ITERATION **do**
3: $q_{rand} \leftarrow$ RandomConfig();
4: $q^a_{near} \leftarrow$ NearestNeighborNode(T_a, q_{rand});
5: $q^a_{reached} \leftarrow$ **ExtendCTM**(T_a, q^a_{near}, q_{rand});
6: $q^b_{near} \leftarrow$ NearestNeighborNode(T_b, $q^a_{reached}$);
7: $q^b_{reached} \leftarrow$ **ExtendCTM**(T_b, q^b_{near}, $q^a_{reached}$);
8: **if** isConnected(T_a, T_b) **then**
9: $P \leftarrow$ ExtractPath(T_a, T_b);
10: **return** SmoothPath(P);
11: **else**
12: Swap(T_a, T_b)
13: **end if**
14: **end while**
15: **return** *None*

The extension process is terminated when the configuration violates any constraint, such as collisions with obstacles, and the last configuration is the configuration before the invalid one. The other tree undergoes a similar process, but the target is set to the configuration returned from the first tree, rather than the sampled configuration. Once the two trees are linked, a feasible path connecting the start and goal configurations can be found. Otherwise, the trees are exchanged and the whole process is repeated. To achieve a shorter path, a shortcut smoothing algorithm may be employed. A straightforward approach is to randomly select two nodes from the path and attempt to insert a short cut between them.

Algorithm 2 illustrates the ExtendCTM function, which incorporates a series of configuration nodes into the tree, starting from the closest node and moving towards the target direction. The DiscretedPath function breaks down the line between the closest and target configurations into multiple waypoints using a predetermined length $\Delta step$. As these waypoints may not lie on the constraint manifold because of the arbitrary nature of the sampled configuration, CTM is employed to transfer each waypoint to the corresponding configuration on the manifold.

These transferred configurations are stored (line 4-7) but are not directly inserted into the tree, because they may violate other (inequality) constraints such as joint limitations and obstacle avoidance. To filter out invalid configurations, the transferred configurations are passed through the takewhile* functions. These functions iteratively examine each element in the list, keeping the element if the statement is true until an invalid one is encountered. The takewhileValidConfig function retains the element if it satisfies the joint limitations and desired constraints, and does not exceed $\Delta step$ in each joint's moving distance. Similarly, the takewhileCollisionFree function retains the first few elements that do not collide with obstacles. After this, only the valid configurations remain

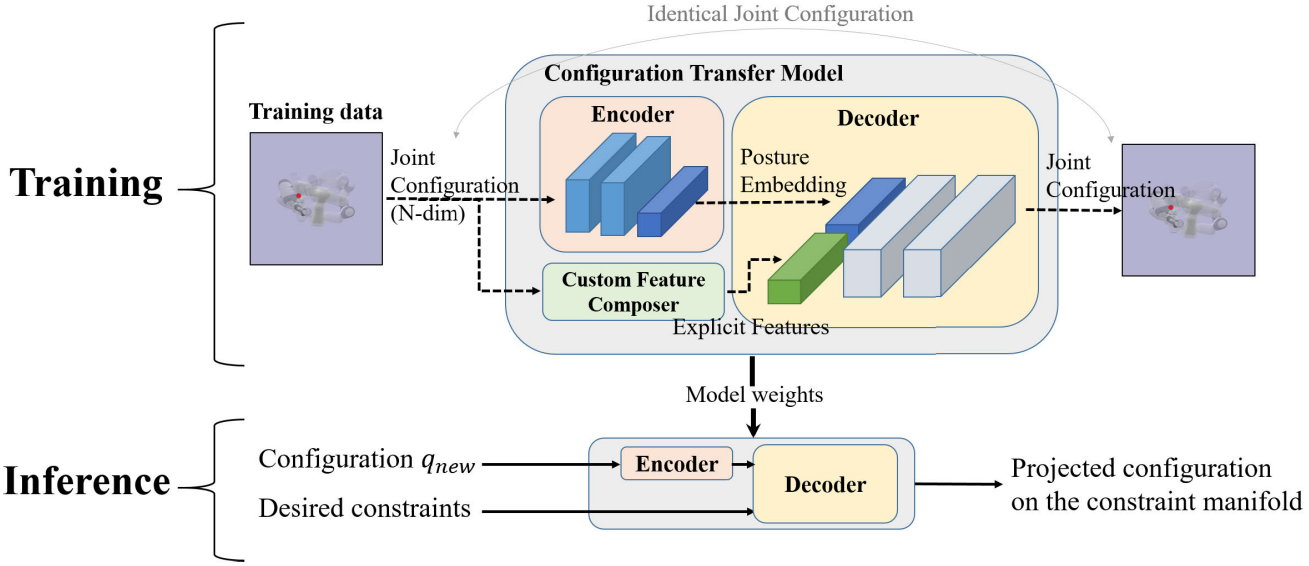

FIGURE 2. Architecture of CTM, which comprises of three modules: an encoder, a decoder, and a custom feature composer. The encoder compresses the joint configuration into an implicit feature embedding. The custom feature composer is a user-defined, problem-specific module that during training identifies the explicit features from the input configuration. These explicit features will be constrained by the equality constraints of the application at online motion planning. Finally, the decoder learns how to regenerate the input configuration knowing the values of the explicit features. Hence, during online motion planning, the decoder can produce a valid new configuration given a sampled configuration and a set of equality constraints on the explicit features.

Algorithm 2 ExtendCTM(T, q_{near}, q_{target})

Input: configuration tree: T; the nearest configuration: q_{near}; target configuration: q_{target};
Output: the last reached configuration $q_{reached}$

$\Delta step \leftarrow 0.05$ ▷ 0.05 degrees for revolute joints
$q_{reached} \leftarrow q_{near}$
$q^{1:K} \leftarrow$ DiscretedPath(q_{near}, q_{target}, $\Delta step$);
for each q^i in $q^{1:K}$ **do**
  $q^i_{onMfd} \leftarrow$ **CTM**(q^i, *EqualityConstraint*);
  q_{onMfd}.append(q^i_{onMfd});
end for
$q^{1:k}_{onMfd} \leftarrow$ takewhileValidConfig($q^{1:K}_{onMfd}$, $\Delta step$);
$q^{1:n}_{safe} \leftarrow$ takewhileCollisionFree($q^{1:k}_{onMfd}$);
for each q_{node} in $q^{1:n}_{safe}$ **do**
  T.Add(q_{node})
  $q_{reached} \leftarrow q_{node}$
end for
return $q_{reached}$

and can be inserted into the tree. It is important to note that additional takewhile* functions may be necessary if the application has more constraints, and multiple CTMs may need to be trained for different types of equality constraints.

IV. EXPERIMENTS

In Sec. IV-A, we provide a detailed description of the experimental setups. Sec. IV-B focuses solely on evaluating the performance of CTM in comparison with other techniques. CTM is the primary core of LAC-RRT, responsible for converting a sampled configuration to a valid configuration that satisfies given equality constraints. By evaluating the CTM module, we can gain a better understanding of LAC-RRT. Sec. IV-C presents a comparison of different CTMs that can convert configurations to the same constraint manifold to determine which design is superior in cases where constraints can be reduced. In Sec. IV-D, we compare the performance of motion planning in different scenarios across various approaches.

A. EXPERIMENTAL SETUP

For the evaluation of the proposed method, we utilized the Bullet Real-Time Physics Simulator [4] and the 7-DoF robotic manipulator, Franka Emika Panda.[1] All experiments were conducted on a computer with an Intel i7-8700 CPU and 64 GB of memory. In the experiments, we imposed equality constraints on either the end-effector position or orientation, as these are commonly used in manipulator planning. We adopted Variational Auto-encoder [14] as the encoder-decoder structure in CTMs. Most CTMs shared a similar architecture, with both the encoder and decoder consisting of two fully connected layers, each layer containing 512 neurons, and a posture embedding length of 4. As additional constraints, we considered the orientation of the end-effector and its position. The orientation of the end-effector is represented in 4 dimensions, while the position is represented in 3 dimensions, resulting in an approximate 50-50 ratio between the explicit feature and posture embedding. For implementing the CFC module, utilizing

[1] https://www.franka.de/

TABLE 1. Comparisons of the compliance rate, distance error, and average computation time using the gradient descent projection, the pseudoinverse projection, and the proposed method. The distance error was measured in meters. A converted configuration was considered compliant if its constrained features were within 0.05 meters of the constraints. Each constraint was tested 1000 times. Please note that CR stands for compliance rate.

Type of constraints	Value of constraints	Gradient Descent Projection			PseudoInverse Projection			CTM (Ours)		
		CR (%)	Error	Time (sec)	CR (%)	Error	Time (sec)	CR (%)	Error	Time (sec)
EE Position	(_, _, 0.4)	62.8	0.012	0.188	68.7	<0.001	0.358	61.2	0.017	**0.004**
	(_, -0.3, _)	50.4	0.019	0.201	65.0	<0.001	0.359	55.0	0.016	**0.004**
	(0.5, _, _)	27.0	0.028	0.208	62.4	0.001	0.357	50.8	0.016	**0.004**
	(0.2, -0.2, _)	20.7	0.025	0.213	44.5	<0.001	0.357	34.3	0.023	**0.004**

forward kinematics is sufficient because it allows us to calculate both constrained features. To accelerate the training process, we stored configurations at 30-degree intervals for each motor, and used forward kinematics to compute the corresponding end-effector pose beforehand.

The proposed method is compared with several algorithms, including the Jacobian-pseudo inverse projection method [1], [2], the gradient descent projection method, and optimization algorithms such as least squares optimization and limited-memory bound-constrained Broyden–Fletcher–Goldfarb–Shanno (L-BFGS-B) optimization [12], [19], [22]. The gradient descent projection method used in this study is the conventional approach that necessitates gradient computation instead of the randomized gradient descent method [10], [21]. To provide more experimental results, optimization algorithms that are commonly used to meet the objective with an initial state are also included. We implemented the CTMs in Pytorch and measured the computation time using the *time.clock()* function from the Python Standard Library. It should be noted that the algorithms compared in this study are also implemented in Python, which can be 10-100 times slower than C during runtime. Therefore, the findings drawn in this paper only reveal the effect of LAC-RRT when using high-level languages.

B. ACCURACY AND COMPUTATION TIME

In this section, we compare the compliance rate, error, and computation time of different approaches to evaluate their performance during the extension process, which is a crucial step in sampling-based motion planning methods (see Sec. III-C). To simulate the extension process during motion planning, we first set a random valid configuration to the manipulator without considering obstacles, and then we evaluate the converted configuration obtained from the different methods. To ensure a fair comparison among the methods, we used the average error of the proposed method, i.e., 0.01, as the termination condition of other methods that involve iterative processes. For more comprehensive information on the parameters used in our experiments, please refer to Appendix A.

The comparison results with equality constraints on end-effector position are presented in Table 1, where the error is evaluated based on the L2 norm between the desired and achieved constraints. The table shows that CTM outperforms the two conventional projection techniques in terms of speed, requiring only 1.4 - 3.1% of the computation time to generate a feasible configuration. However, since CTM is a learning-based approach, it is subject to relatively high distance errors because of model error. While CTM and gradient descent projection exhibit similar errors, CTM's errors are 10 times higher than those of the pseudoinverse method.

Table 2 presents a comparison of optimization methods for meeting the equality constraints on the end-effector position and orientation. The L2 norm is utilized to measure position error, whereas orientation error is measured by the following equation:

$$Error_{orieantation} = 1 - (quat1 \cdot quat2)^2$$

where quat1 and quat2 represent orientations in quaternion. A smaller error indicates that the two quaternions are closer. For more information about the constraints, please refer to Appendix A. Similar to Table 1, CTM exhibits high efficiency but may have higher errors. In comparison to least squares optimization, CTM requires only 4.3% and 14.8% of computation time to meet position and orientation constraints, respectively. Moreover, compared to L-BFGS-B, CTM reduces computation time by 80% while meeting orientation constraints and has similar computation time in meeting position constraints. We also show the results in the form of plots, please refer to Appendix B.

Another noteworthy finding is that the compliance rates of other approaches in meeting constraints are higher than those of CTM. By examining failed cases, it is noticed that CTM struggles to convert configurations near the boundary of the configuration space, which may be because of the fewer datapoints in those areas. This issue is not present in other conventional methods. However, other methods still have their limitations and problems, such as getting stuck in local minima while approaching the manifold, which prevents obtaining valid configurations.

To summarize, CTM significantly improves efficiency in transforming random configurations to meet specific constraints. Although it has higher error rates compared to other methods, it can be complemented by methods such as numerical optimizations to achieve the required precision with a much reduced iteration time.

C. COMPARISONS OF CTMs UNDER THE SAME OBJECTIVE

The previous section evaluates the performance of LAC-RRT in meeting position and orientation constraints. The

TABLE 2. Comparisons of the compliance rate, distance error, and average computation time using the least squares optimization, the L-BFGS-B optimization, and the proposed method over 1000 trials. The distance error was measured in meters, and a converted configuration was considered compliant if its constrained features were within 0.05 meters of the constraints. For orientation constraints, compliance required achieving an orientation error of less than 0.05. Please note that CR stands for compliance rate.

Type of constraints	Value of constraints	Least Squares			L-BFGS-B			CTM (Ours)		
		CR (%)	Error	Time (sec)	CR (%)	Error	Time (sec)	CR (%)	Error	Time (sec)
EE Position	(_, _, 0.4)	98.5	<0.001	0.015	92.9	0.007	0.005	61.2	0.017	**0.004**
	(_, -0.3, _)	97.0	0.001	0.016	85.9	0.011	0.006	55.0	0.016	**0.004**
	(0.5, _, _)	94.0	0.003	0.205	72.9	0.007	0.007	50.8	0.016	**0.004**
	(0.2, -0.2, _)	92.7	0.001	0.128	70.8	0.014	0.007	34.3	0.023	**0.004**
EE Orientation	Downwards	29.7	0.001	0.017	17.6	0.003	0.016	22.1	0.005	**0.003**
	Upwards	22.8	0.003	0.018	24.1	0.004	0.015	22.3	0.006	**0.004**
	Random1	18.5	0.031	0.023	21.2	0.030	0.017	23.1	0.028	**0.003**
	Random2	32.3	0.038	0.030	27.6	0.035	0.015	22.8	0.034	**0.003**

TABLE 3. Comparisons of the average compliance rate and average error using different CTMs over 1000 trials. Please note that CR stands for compliance rate.

	(x, y, z)		Z only	
	CR (%)	Error (m)	CR (%)	Error (m)
(_, _, 0.2)	27.2	0.02	54.2	0.02
(_, _, 0.3)	32.0	0.02	53.7	0.02
(_, _, 0.4)	35.3	0.02	58.7	0.02
(_, _, 0.5)	32.3	0.02	61.0	0.02

TABLE 4. Comparisons of the average steps in each plan, and average computation time using the least squares optimization, the L-BFGS-B optimization, and the proposed method over 100 trials.

	S1-1		S1-2	
	Steps	Time (sec)	Steps	Time (sec)
Least Squares	114.74	10.55	144.02	79.54
L-BFGS-B	91.58	2.06	160.87	52.20
LAC-RRT (Ours)	79.02	**1.41**	106.44	**4.47**

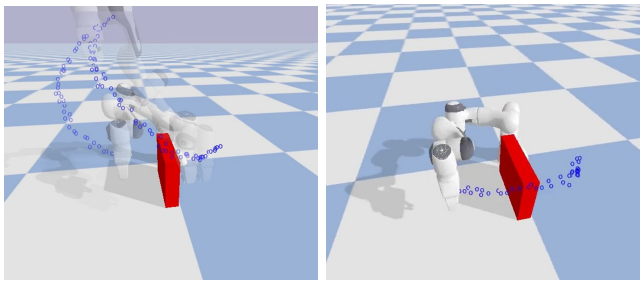

FIGURE 3. Two paths planned by our method for Scenario 1-2.

experiments utilize CTM to calculate a feasible configuration with the full constraints, such as tuple (x, y, z), even if the desired constraint only pertained to z. Another architecture option is to pass only z through CTM rather than the entire position, which includes both current x and y. This section aims to compare the performance of different CTMs that target the same objective, to determine the optimal design.

The comparison of CTMs with input constraints of z only and (x, y, z) is shown in Table 3. The experimental results suggest that employing the complete position as input achieves higher compliance rates than relying solely on z. Therefore, it is recommended to incorporate the complete constraint when designing the CTM architecture. Further investigation is left as future work.

D. LAC-RRT EVALUATION

Section IV-B compares the efficiency of various methods in converting a random configuration to a valid one based on the imposed constraints. This section evaluates the performance of these methods in complete motion planning. To evaluate the performance, we employ two distinct motion planning scenarios, each consisting of a start and goal configuration on the constraint manifold. The primary objective is to determine a collision-free path from the start configuration to the goal. BiRRT serves as the foundation for LAC-RRT, as outlined in Sec. III-C, which may result in a suboptimal planned path. For other techniques, the algorithm remains the same, except for the extension stage. We repeat each scenario 100 times to ensure accuracy.

1) SCENARIO 1: PLANNING A PATH WHILE KEEPING THE ORIENTATION

For Scenario 1, the end-effector orientation must remain downwards throughout the motion. This is typically used when the manipulator is holding a liquid-filled container, such as a glass of water, and needs to move it from one surface to another. In Scenario 1-1, the objective is simply to reach the goal configuration from the starting configuration. In Scenario 1-2, in addition to approaching the goal configuration, the manipulator must also avoid obstacles between the configurations, as shown in Fig. 3.

Table 4 displays the average number of steps per plan and computation time, based on 100 trials. Results indicate that LAC-RRT requires 94.4% and 91.4% less computation time to plan a motion in Scenario 1-2, compared to least squares optimization and L-BFGS-B, respectively. It is important to note that the average number of steps only accounts for the length of the final path and not discarded configurations, but the computation time is estimated throughout the entire

TABLE 5. Comparisons of the success rate, distance error, and average computation time using the proposed method and other 5 approaches over 20 trials. Hybrid-LAC is a hybrid method that combines our method and L-BFGS-B, please refer to IV-D2.

	S2-1			S2-2			S2-3		
	Success (%)	Steps	Time (sec)	Success (%)	Steps	Time (sec)	Success (%)	Steps	Time (sec)
Gradient Descent	100	213.05	118.14	85	207.7	529.66	100	141.35	833.40
PseudoInverse	100	196.60	47.27	100	145.1	111.67	100	182.80	309.92
Least Squares	100	263.49	94.26	100	345.3	355.4	100	188.55	1396.98
L-BFGS-B	100	193.15	**2.22**	100	241.2	**13.46**	100	196.35	64.70
LAC-RRT (Ours)	100	203.11	5.37	100	204.6	46.00	100	168.35	**40.22**
Hybrid-LAC*	100	183.20	3.67	100	237.9	26.48	100	180.05	97.38

planning process. Therefore, the number of steps may not have a strong positive correlation with computation time.

2) SCENARIO 2

In Scenario 2, the end-effector is restricted to moving either vertically or horizontally, resembling tasks such as painting a wall or writing a note on a table. Scenario 2-1 involves reaching the goal configuration from the starting configuration on a horizontal plane. In Scenario 2-2, apart from approaching the goal configuration on a horizontal plane, the manipulator must also avoid obstacles. For Scenario 2-3, the manipulator needs to avoid collisions and reach the goal configuration on a vertical plane. Fig. 4 illustrates two different paths found by our methods.

Table 5 presents the success rate, average number of steps per plan, and computation time based on 20 trials. In Scenarios 2-1 and 2-2, the proposed method and the L-BFGS-B optimization exhibit higher efficiency than the other methods, requiring only about 10% of the computation time to achieve planning. Notably, the L-BFGS-B optimization outperforms LAC-RRT in these scenarios. This finding is supported by Table 2, which shows that L-BFGS-B offers not only high efficiency but also a higher success rate concerning cases of position constraints. In other words, if the planning involves areas where LAC-RRT struggles, our method may need to sample more configurations and attempt alternate paths, resulting in longer computation time. In the worst-case scenario, LAC-RRT may not guarantee finding a solution, even if feasible paths exist. We also show the results in the form of plots, please refer to Appendix B.

To overcome this issue, one possible solution is to collect more training data and enhance the areas where LAC-RRT does not perform well. In this paper, we propose an alternative solution, which involves a hybrid approach that combines the strengths of LAC-RRT and L-BFGS-B techniques. By combining these two techniques, we can take advantage of the efficiency of LAC-RRT in certain scenarios where L-BFGS-B is comparatively slower, e.g., orientation constraints, while simultaneously addressing the shortcomings of LAC-RRT, which can be susceptible to model error. Specifically, Hybird-LAC involves passing a sampled configuration through LAC-RRT to obtain a converted configuration, and then we verify if the converted configuration violates any desired constraints. If so, we use L-BFGS-B to calculate the projected configuration again. We drop the configuration and resample one if both approaches cannot obtain a valid configuration. Although the Hybrid-LAC approach is slower than using L-BFGS-B alone in Scenario 2 due to the additional overhead of passing through LAC-RRT, it should still outperform the sole use of L-BFGS-B in cases where orientation constraints are involved.

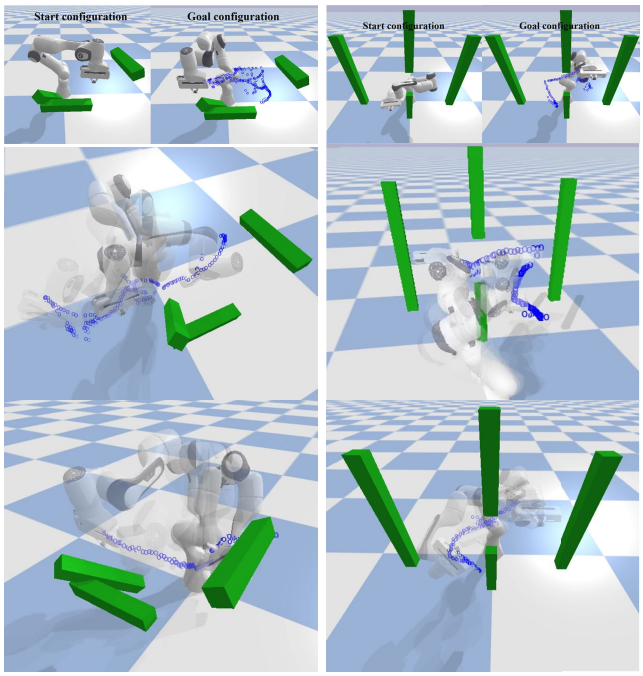

FIGURE 4. Two paths planned by our method for Scenario 2-2 (left) and Scenario 2-3 (right).

V. CONCLUSION AND FUTURE WORKS

This paper proposes a learning-assisted RRT-based algorithm for constrained motion planning. LAC-RRT exploits a self-supervised model to efficiently convert a random configuration to a valid one that satisfies the imposed equality constraints, reducing computation time during planning. To handle equality constraints, LAC-RRT utilizes the CTM module to obtain valid joint configurations. For inequality constraints, such as obstacle avoidance, LAC-RRT incorporates off-the-shelf collision detection algorithms and rejection sampling to discover feasible configurations. The proposed

TABLE 6. The value of the constraints in the experiments.

Name of Constraint	Value
Downwards	(0, 1, 0, 0)
Upwards	(0, 0, 1, 0)
Random1	(0.028, -0.709, -0.365, 0.629)
Random2	(1, 0.2, 0, 0)

TABLE 7. Parameters used for the projections and optimizations.

Method	Learning rate	Max. iter.	threshold(tol)
Gradient Descent	0.01	1000	0.01
PseudoInverse Projection	0.01	1000	0.01
Least Squares	-	1000	0.01
L-BFGS-B	-	1000	0.01

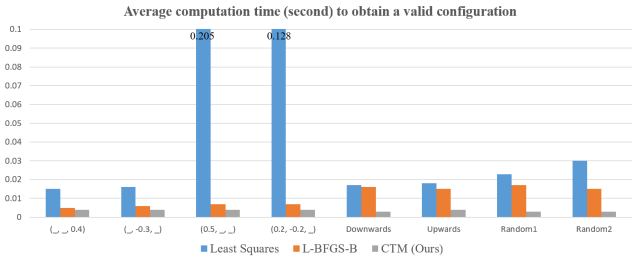

FIGURE 5. Comparisons of the average computation time using the least squares optimization, the L-BFGS-B optimization, and the proposed method over 1000 trials.

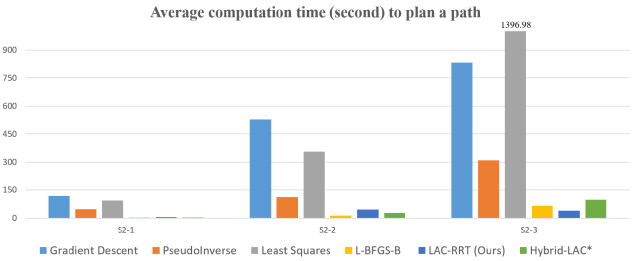

FIGURE 6. Comparisons of the average computation time using the proposed method and other 5 approaches over 20 trials. Hybrid-LAC is a hybrid method that combines our method and L-BFGS-B.

method outperforms other methods in the majority of the cases. The experiments show that LAC-RRT requires only 1.4 - 3.1% of the computation time to generate a feasible configuration compared to conventional projection methods. In comparison to least squares optimization, our method requires only 4.3% and 14.8% of computation time to meet position and orientation constraints, respectively. For motion planning, LAC-RRT only needs about 10% computation time to plan a constrained motion. This is the first paper that accelerates the planning process by learning the relationship between configurations for solving the contained motion planning problem.

Although LAC-RRT can be highly efficient during planning, it requires sufficient and comprehensive training data that covers the entire configuration space. Determining the exact quantity of necessary data can be challenging and may vary for different manipulators. Future research should focus on improving the CTM module after training has been completed. One possible approach is to plan multiple motions without considering obstacles and gather training data from areas where LAC-RRT experiences higher error rates. Moreover, our experiments employed a single CTM dedicated to handling a specific type of feature (constraint). To further advance the field, a future study could explore the integration of multiple features within a single CTM.

APPENDIX A
IMPLEMENTATION DETAILS

See Tables 6 and 7.

APPENDIX B
EXPERIMENTAL RESULTS IN THE FORM OF PLOTS

See Figures 5 and 6.

REFERENCES

[1] D. Berenson, S. S. Srinivasa, D. Ferguson, and J. J. Kuffner, "Manipulation planning on constraint manifolds," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 625–632.
[2] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *Int. J. Robot. Res.*, vol. 30, no. 12, pp. 1435–1460, Oct. 2011.
[3] L. Chen, Z. Jiang, L. Cheng, A. C. Knoll, and M. Zhou, "Deep reinforcement learning based trajectory planning under uncertain constraints," *Frontiers Neurorobotics*, vol. 16, Aug. 2022, Art. no. 883562.
[4] E. Coumans and Y. Bai, "Pybullet, a Python module for physics simulation for games, robotics and machine learning," Tech. Rep., 2016. [Online]. Available: http://pybullet.org/
[5] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.
[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011.
[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.
[8] A. T. Khan, S. Li, S. Kadry, and Y. Nam, "Control framework for trajectory planning of soft manipulator using optimized RRT algorithm," *IEEE Access*, vol. 8, pp. 171730–171743, 2020.
[9] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. ICRA. Millennium Conf. IEEE Int. Conf. Robot. Automat. Symposia*, vol. 2, 2000, pp. 995–1001.
[10] S. M. LaValle, J. H. Yakey, and L. E. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, Jul. 1999, pp. 1671–1676.
[11] T. S. Lembono, E. Pignat, J. Jankowski, and S. Calinon, "Learning constrained distributions of robot configurations with generative adversarial network," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 4233–4240, Apr. 2021.
[12] P. Manceron. (2022). *IKPy: An Inverse Kinematics Library Aiming Performance and Modularity (V3.3.3)*. [Online]. Available: https://github.com/Phylliade/ikpy
[13] P. Rousseas, C. P. Bechlioulis, and K. J. Kyriakopoulos, "Optimal robot motion planning in constrained workspaces using reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 6917–6922.
[14] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 3483–3491.
[15] M. W. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Hoboken, NJ, USA: Wiley, 2020.
[16] M. Stilman, "Task constrained motion planning in robot joint space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2007, pp. 3074–3081.

[17] I. A. Sucan and S. Chitta, "Motion planning with constraints using configuration space approximations," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2012, pp. 1904–1910.
[18] G. Sutanto, I. R. Fernández, P. Englert, R. K. Ramachandran, and G. Sukhatme, "Learning equality constraints for motion planning on manifolds," in *Proc. Conf. Robot Learn.*, 2021, pp. 2292–2305.
[19] P. Virtanen et al., "SciPy 1.0: Fundamental algorithms for scientific computing in Python," *Nature Methods*, vol. 17, pp. 261–272, Feb. 2020.
[20] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong, and W. Rui, "Bidirectional potential guided RRT* for motion planning," *IEEE Access*, vol. 7, pp. 95046–95057, 2019.
[21] Z. Yao and K. Gupta, "Path planning with general end-effector constraints," *Robot. Auto. Syst.*, vol. 55, no. 4, pp. 316–327, Apr. 2007.
[22] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization," *ACM Trans. Math. Softw.*, vol. 23, no. 4, pp. 550–560, Dec. 1997.

CHI-KAI HO received the bachelor's degree from the Department of Computer Science, National University of Kaohsiung, Taiwan. He is currently pursuing the Ph.D. degree with the Department of Computer Science, National Tsing Hua University, Taiwan. His current research interests include reinforcement learning and robotic manipulation.

CHUNG-TA KING received the B.S. degree in electrical engineering from National Taiwan University, Taiwan, in 1980, and the M.S. and Ph.D. degrees in computer science from Michigan State University, East Lansing, MI, USA, in 1985 and 1988, respectively. From 1988 to 1990, he was an Assistant Professor of computer and information science with the New Jersey Institute of Technology, Newark, NJ, USA. In 1990, he joined the Faculty of the Department of Computer Science, National Tsing Hua University, Taiwan, where he is currently a Professor. He was the Chair of the Department of Computer Science, National Tsing Hua University, from 2009 to 2012. His research interests include parallel and distributed processing and networked embedded systems.