

# Adaptive Diffusion Constrained Sampling for Bimanual Robot Manipulation

Haolei Tong<sup>1,\*</sup>, Yuezhe Zhang<sup>1,\*</sup>, Sophie Lueth<sup>1</sup>, Georgia Chalvatzaki<sup>1,2,3</sup>

**Abstract**— Coordinated multi-arm manipulation requires satisfying multiple simultaneous geometric constraints across high-dimensional configuration spaces, which poses a significant challenge for traditional planning and control methods. In this work, we propose Adaptive Diffusion Constrained Sampling (ADCS), a generative framework that flexibly integrates both equality (e.g., relative and absolute pose constraints) and structured inequality constraints (e.g., proximity to object surfaces) into an energy-based diffusion model. Equality constraints are modeled using dedicated energy networks trained on pose differences in the Lie algebra space, while inequality constraints are represented via Signed Distance Functions (SDFs) and encoded into learned constraint embeddings, allowing the model to reason about complex spatial regions. A key innovation of our method is a Transformer-based architecture that learns to weigh constraint-specific energy functions at inference time, enabling flexible and context-aware constraint integration. Moreover, we adopt a two-stage batch-wise sampling strategy that improves precision and sample diversity by combining Langevin dynamics with resampling and density-aware re-weighting. Experimental results on dual-arm manipulation tasks show that ADCS significantly improves sample diversity and generalization in settings demanding precise coordination and adaptive constraint handling. Our website is made publicly available at: [thomasstong.github.io/ADCS.github.io/](https://thomasstong.github.io/ADCS.github.io/)

## I. INTRODUCTION

Robots operating in human environments must be capable of performing increasingly complex tasks that involve interaction with diverse objects, unstructured scenes, and coordination across multiple end-effectors. As tasks grow in complexity [1], such as moving large furniture, assembling components, individual manipulators often reach their physical and functional limits. Multidegrees-of-freedom systems, such as bimanual robots and mobile manipulators, provide a scalable way to address such limitations by enabling spatial coordination and collaborative manipulation [2].

One core capability that these systems unlock is collaborative transport, where multiple agents work together to move objects that are too large, heavy, or geometrically constrained for a single robot to handle. However, this capability comes at the cost of dramatically increased complexity: planning motion trajectories for such systems requires satisfying a wide range of spatial and physical constraints, e.g., maintaining stable grasps, ensuring collision-free motion among robots and with the environment, and respecting the kinematic limits of each agent. This results in high-dimensional

problems with complex nonlinear constraints in configuration and task spaces [3].

Traditional constrained optimization methods and sampling techniques such as Markov Chain Monte Carlo (MCMC) [4] offer principled approaches to constraint satisfaction in such settings. However, their reliance on local gradient information often limits their capacity to explore multimodal distributions or reason globally about the scene geometry. Recent generative approaches have shown promise; diffusion models were applied to SE(3) grasp synthesis [5] and to compositional object placement [6], but they typically assume fixed constraint structures and manually specified weighting schemes. Moreover, they often struggle to scale to systems with high-dimensional, interdependent constraints, such as those in multi-arm or mobile manipulation platforms.

To address these limitations, we introduce **Adaptive Diffusion Constrained Sampling (ADCS)**, a generative framework for high-dimensional, constraint-aware sampling in multi-DoF robotic systems. ADCS is designed to integrate both equality and inequality constraints in SE(3) task space while remaining adaptable to different robot morphologies and task conditions. The key novelty lies in two adaptive mechanisms. First, we introduce an *Adaptive Constrained Conditioning*, in which the model during training learns to generate feasible SE(3) poses conditioned on the scene geometry via Signed Distance Fields (SDFs). During sampling, we apply differentiable constraints in joint space using a chain rule, enabling flexible post-hoc adaptation to robot-specific constraints such as joint limits, reachability, and grasp feasibility. Second, we propose a *Compositional Weighting Transformer (CWT)* method, in which, instead of manually specifying energy-function weights, we incorporate a Transformer-based architecture that learns to dynamically compose task-specific energy terms, allowing the system to prioritize constraints adaptively across varying tasks and contexts, to allow for optimizing challenging tasks with varied co-occurring constraints. Moreover, the CWT can handle new types of constraints without requiring retraining.

In addition, ADCS uses a two-stage sampling strategy that combines Langevin dynamics, density-aware re-weighting, and Gauss-Newton refinement, which leads to faster convergence and better precision in constrained sampling. We evaluate ADCS in several collaborative manipulation scenarios, including object transport and pattern stippling, using both simulated and real-world multi-arm systems – a two Franka Emika Panda arm system and a bimanual TIAGo robot. Across these settings, ADCS consistently outperforms baseline approaches in terms of task success, sampling efficiency,

\*This project has received funding from the European Union’s Horizon Europe programme under Grant Agreement No. 101120823, project MANiBOT. <sup>1</sup>TU Darmstadt, <sup>2</sup>Hessian.AI, <sup>3</sup>Robotics Institute Germany. \* indicates equal contribution. Email: [haoleitong24@gmail.com](mailto:haoleitong24@gmail.com)

generalization to novel scenes, and constraint satisfaction.

To sum up, our main contributions are as follows. **(i)** We propose *ADCS*, a diffusion-based framework for constraint-aware generative sampling in multi-DoF robotic systems. **(ii)** We introduce CWT, a Transformer-based mechanism that dynamically composes constraint energies, eliminating manual weight tuning and generalizing to new constraints. **(iii)** We formulate a two-stage batch-wise sampling process that enables the application of task- and robot-specific constraints in joint space, even post training.

## II. RELATED WORKS

**Constrained Sampling.** Sampling under constraints is central to problems in MCMC, optimization, and robotics. With equality constraints, the task reduces to sampling on a manifold [7], [8], while linear inequalities lead to polytope sampling [9]. [4] proposed a restart-based NLP sampler for nonlinear constraints in robotics, assuming only local access to cost functions. In contrast, we focus on sampling from complex, multimodal constrained distributions.

**Generative Models for Constrained Optimization.** Deep generative models such as DDPM [10] and SVGD [11] are powerful tools for sampling from multimodal distributions. Their ability to capture disconnected regions makes them attractive for constraint-aware inference [12]. Existing works [13] [6] apply diffusion models to constraint-aware tasks, but often focus on low-dimensional object poses or categorical approximations. Diffusion-based motion planners [14]–[16] reframe planning as sampling in energy-based models, but struggle to scale with multiple, nonlinear constraints. Our method addresses this by learning a compositional energy model with transformer-based adaptive weighting for rich constraint integration.

**Automatic Weight Tuning.** Designing appropriate constraint weights is tedious and sensitive [5] [17]. Prior methods either optimize weights [18] [19] or learn them from data [20] [21], but these are not directly applicable to diffusion-based models. Recent approaches [5] [6], [14] use fixed weights, whereas our method trains a transformer (CWT) to reason over constraint types and adaptively weight them at inference.

Compared to the prior work, Diffusion-CCSP [6], it relies on fixed weights to compose constraints and utilizes basic Langevin dynamics for sampling. In contrast, our method incorporates the task-specific feature functions from *ADCS*, which are essential for enhancing generalization. And also our method benefits from transformer-based learnable constraint weighting and a more expressive two-stage sampling strategy, resulting in better performance across tasks.

## III. PRELIMINARIES

Here, we formally define the constrained sampling problem, describe the types of constraints considered, and outline the NLP-based sampling method used to generate training data for our model.

**Problem Statement.** We consider a multi-DoF robotic system composed of multiple manipulators operating in a

shared workspace. Let  $\mathcal{Q} \subset \mathbb{R}^{n \times d}$  denote the joint configuration space, where  $n$  denotes the number of robots/end-effectors and  $d$  denotes the DoF of each robot, and let  $\mathcal{C} = \{c_i\}$  be a set of task-specific constraints on configurations  $q \in \mathcal{Q}$ . Each constraint is either an equality  $c_i(q) = 0$  or an inequality  $c_i(q) \leq 0$ . These constraints capture spatial relationships, kinematic feasibility, and task semantics across the robot and the environment.

**Constraint Types.** We consider the following types of constraints: **(i) SE(3) Pose Constraints:** enforce absolute or relative poses between end-effectors. Given two poses  $H_1, H_2 \in \text{SE}(3)$ , the constraint is  $H_1^{-1}H_2 = T_{\text{target}}$ ; **(ii) Orientation Constraints:** constrain either a specific axis (e.g., Y-axis of the gripper aligned with Z-axis of the world) or the full rotation  $R = R_{\text{goal}}$ ; **(iii) Midpoint Constraints:** enforce the midpoint between two end-effectors to match a target (equality) or lie on a surface (inequality); **(iv) Signed Distance Constraints:** encode proximity to object surfaces for surface contact or collision avoidance; **(v) Joint and Self-Collision Constraints:** enforce joint limits and prevent inter-link collisions using a set of differentiable bounding spheres.

**NLP Sampling.** To generate training data, we use *NLP Sampling* [4], a method for sampling from the constraint-satisfying region defined by  $\mathcal{C}$ . Each constraint  $c_i(q)$  is mapped to a slack term: inequality constraints use  $[c_i(q)]_+$ , and equality constraints use  $|c_i(q)|$ , where  $[\cdot]_+$  denotes the ReLU function. The aggregated slack vector  $s(q)$  defines the relaxed energy:

$$F_{\gamma, \mu}(q) = \gamma f(q) + \mu \|s(q)\|^2, \quad (1)$$

where  $f(q)$  is a task-specific density and  $\gamma, \mu$  balance task likelihood and constraint satisfaction. Sampling proceeds in two stages: first, Gauss-Newton descent minimizes  $\|s(q)\|^2$  over  $K_{\text{down}}$  steps to find a feasible configuration; then, interior sampling, such as manifold-RRT [7] or Langevin dynamics [10], is applied for  $K_{\text{burn}}$  iterations to generate diverse samples from the feasible region. This approach produces high-quality training data for constraint-aware generative modeling.

## IV. ADAPTIVE DIFFUSION CONSTRAINED SAMPLING

In this section, we describe our framework for generating diverse, constraint-satisfying solutions in multi-DoF robotic systems. We first introduce our compositional energy-based formulation for modeling constraints in a diffusion framework. We then describe the model architecture, training loss, and how the learned model can be used for efficient sampling at inference time.

### A. Adaptive Compositional Diffusion with Conditioning

Given a set of constraints  $\mathcal{C} = \{c_i\}$ , we aim to sample robot poses  $H \in \text{SE}(3)^n$  (for  $n$  end-effectors) that satisfy all constraints in  $\mathcal{C}$ . We model the conditional distribution over poses using an energy-based model (EBM), enabled by the use of diffusion models. Unlike standard DDPMs [10] that generate samples from noise via learned denoisers, we use diffusion training solely to learn a score function for

constraint-driven energy modeling based on denoising score matching [22]. For each individual constraint  $c \in \mathcal{C}$ , we define an energy function  $E_\theta(H | c)$ , and the associated probability is given by  $p(H | c) \propto \exp(-E_\theta(H | c))$ . We assume these models assign approximately uniform mass over the feasible region of each constraint. To satisfy all constraints jointly, we construct a composed distribution by minimizing the sum of individual energy functions

$$H_0 = \arg \min_H \sum_{c \in \mathcal{C}} E_\theta(H | c). \quad (2)$$

Similarly to [23], the joint diffusion distribution over noisy latent variables  $H_k$  (at timestep  $k$ ) under all constraints  $c_0, \dots, c_N \in \mathcal{C}$  is given by

$$p(H_k | c_0, \dots, c_n) \propto p(H_k) \prod_{i=0}^N \frac{p(H_k | c_i)}{p(H_k)}, \quad (3)$$

which leads to the following form for the composed energy

$$E_\theta(H_k | c_0, \dots, c_n) = w_0 E_\theta(H_k | c_0) + \sum_{i=1}^N w_i (E_\theta(H_k | c_0, c_i) - E_\theta(H_k | c_0)), \quad (4)$$

where  $c_0$  is a fundamental constraint (e.g., task-space goal, which should be satisfied in every task), and the remaining energies are conditioned on the satisfaction of  $c_0$ , since in all of our tasks, the EEF's relative pose constraints  $c_0$  should be satisfied. The weights  $w_i$  determine the influence of each constraint during composition and can be either fixed or learned.

To enable learning in the Lie group  $\text{SE}(3)^n$ , we inject Gaussian noise in the Lie algebra  $\mathfrak{se}(3)^n$ , to maintain manifold consistency. Specifically, for a clean pose  $H$ , we apply a perturbation via the exponential map [5]:

$$\hat{H} = H \cdot \text{Expmap}(\epsilon), \quad \epsilon \sim \mathcal{N}(0, \sigma_k^2 I), \quad \epsilon \in \mathbb{R}^{6n}, \quad (5)$$

where  $\sigma_k$  is the noise scale at timestep  $k$ , and  $\text{Expmap}$  maps noise vectors in the tangent space to elements of the Lie group. To train the model, we adopt a denoising score matching objective [22], which minimizes the discrepancy between the model's gradient and the gradient of the Gaussian perturbation distribution. The training loss is

$$\mathcal{L} = \frac{1}{L} \sum_{k=0}^L \mathbb{E}_{H, \hat{H}} \left[ \left\| \nabla_{\hat{H}} E_\theta(\hat{H} | \mathcal{C}, k) - \nabla_{\hat{H}} \log q(\hat{H} | H, \sigma_k^2 I) \right\|^2 \right] \quad (6)$$

where  $q(\hat{H} | H, \sigma_k^2 I)$  is the Gaussian perturbation distribution in the Lie algebra  $\mathfrak{se}(3)^n$ , and  $\nabla_{\hat{H}}$  denotes the gradient w.r.t. the perturbed pose  $\hat{H} \in \text{SE}(3)^n$ . Since  $\text{SE}(3)^n$  is a Lie group, gradients are computed in the associated tangent space via a local reparameterization (details in the Appendix).

This formulation learns an EBM implicitly through a diffusion process. While standard diffusion models learn a generative process via denoising predictions, our framework instead learns the score function, i.e., the gradient of the log-probability, directly. The trained energy function  $E_\theta(\hat{H} | \mathcal{C}, k)$  thus estimates the gradient of the log-density at different noise levels. At inference time, we discard the

forward diffusion and use the learned score function as input to Langevin dynamics, iteratively refining a sample by ascending the composed energy landscape.

In addition, we introduce an *Adaptive Constrained Conditioning* mechanism to bridge robot-level and task-level constraints. During training, task-level conditions are incorporated into the energy network via a FiLM-based modulation mechanism. For SDF constraints, we jointly train a network that predicts signed distance values from point cloud inputs. At inference time, gradients are propagated through forward kinematics using the chain rule, which enables flexible enforcement of robot-specific constraints (e.g., joint limits and collision) without retraining and ensures adaptability across both robot and task-level constraints.

## B. Compositional Weighting Transformer

To enable flexible and context-aware integration of multiple constraint energies, we introduce CWT. Rather than statically assigning scalar weights to each constraint model, the CWT learns dynamic composition weights conditioned on the current energy values and their interrelations. The key intuition is that the Transformer [24] enables *contextual weighting*: it attends to all constraint-specific energy values jointly and can adapt the relative importance of each constraint depending on their magnitudes, types, or combinations. This makes the composition mechanism *permutation-invariant* (no fixed order of constraints is required) and *adaptive to task conditions*, as the model can, for instance, downweight a nearly satisfied constraint and prioritize violated ones during inference.

As illustrated in Figure 1, each energy scalar  $E_\theta(\hat{H} | c_i)$  for constraint  $c_i \in \mathcal{C}$ , together with the constraints identifier, is treated as a token. Since each type of constraint has its own independent energy encoder, all constraint types are also implicitly encoded. To handle unseen constraints during inference, we also introduce a special UNK identifier, allowing the model to generalize beyond the training set. These tokens are concatenated and passed through a Transformer, which outputs a set of normalized weights  $\{w_i\}$ . The final composed energy is  $E_{\text{total}}(\hat{H}) = \sum_i w_i E_\theta(\hat{H} | c_i)$ . We also use positional encodings for the input tokens, but unlike the original Transformer [24], in CWT, those are randomly sampled at each iteration since there is no intrinsic ordering among constraints. This design supports modularity and compositional generalization across tasks and scenes without retraining. Compared with approaches that fuse constraint features directly via fully connected layers [25], [26], CWT preserves the identity of each constraint's energy and defers fusion to the composition stage. This enables each constraint energy to preserve both geometric structure and semantics, while allowing adaptive trade-offs between them as the constraint energies change with the environment. CWT is trained jointly with the energy networks and feature encoders using the diffusion-based loss in Eq. 6, without requiring additional supervision.

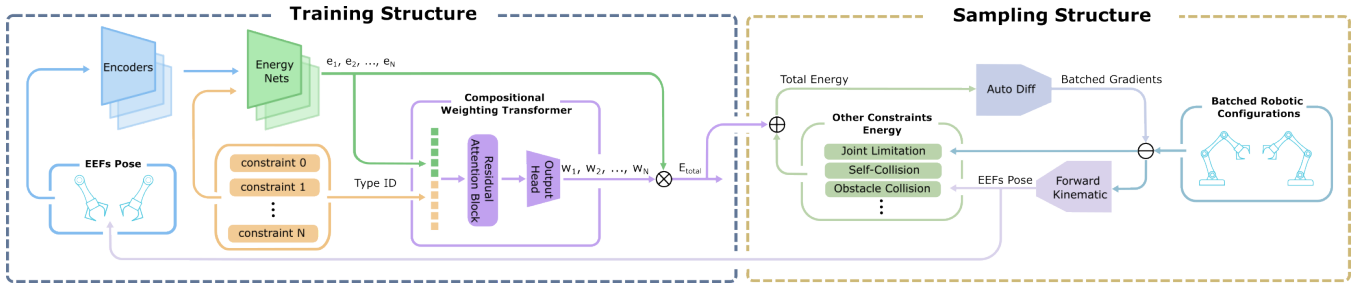


Fig. 1: Overview of the Adaptive Diffusion Constrained Sampling (ADCS) architecture. The model consists of three components: a constraint feature encoder, an energy network (MLP), and the Compositional Weighting Transformer (CWT). During training, noisy poses in  $SE(3)^n$  are used to learn constraint-aware energy functions. In inference, sampling is performed in batched joint space using Langevin dynamics guided by the learned energy landscape.

### C. Two-stage Constrained Sampling

At inference time, we use the learned diffusion model to generate batched joint configurations that satisfy a set of geometric constraints, where the batch size is the number of drawn samples. To this end, we adopt a two-stage sampling strategy inspired by [4], which combines Annealed Langevin Monte Carlo (ALMC) sampling with resampling and correction steps, as shown in Algo. 1.

**Annealed Langevin Sampling.** In each ALMC call, we first draw a batch of joint configurations  $q_L^{n_s}$  from a standard Gaussian distribution. Using PyTorch Kinematics [27], we compute the corresponding end-effector poses  $H_L^{n_s} \in SE(3)^n$  via forward kinematics. These poses are then passed through the energy network to compute their corresponding energies  $E_\theta(H_L^{n_s} | C, k)$ . Next, we compute the batched gradient of the energy w.r.t. the joint configuration  $q_k^{n_s}$  using PyTorch’s AutoDiff and vmap [28]. This gradient is obtained via the chain rule  $\nabla_q E_\theta = \frac{\partial E_\theta(H_k | C, k)}{\partial H_k} \cdot \frac{\partial H_k}{\partial q_k^{n_s}}$ . The joint configurations are then updated using a Langevin step. To improve convergence, we adopt a second-order strategy similar to [14], but instead of relying solely on first-order gradients, we introduce an approximate batch-wise Gauss-Newton update during this correction phase. The update direction is computed as  $g_{n_s} = -(J^\top J)^{-1} J^\top r$ , where  $J = \frac{\partial r}{\partial q} = \frac{\partial r}{\partial H} \frac{\partial H}{\partial q}$  and  $r$  is the residual value. This approximation accelerates convergence while preserving the structure of the learned data distribution. Importantly, we restrict this correction to the refinement phase to avoid distorting the learned distribution during stochastic sampling, as discussed in [29], [30].

**Resampling Phase.** After the initial ALMC sampling and second-order refinement, we perform a resampling step to promote diversity among the generated configurations. We begin by sorting all samples based on their energy values and selecting a fixed number of the top-performing configurations. To avoid over-representing densely clustered solutions, we estimate the sample density  $\rho(x)$  using Kernel Density Estimation (KDE) with a Gaussian kernel [31]:

$$\rho(x) = \frac{1}{Mh} \sum_{i=1}^M K\left(\frac{x-x_i}{h}\right), \quad K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right), \quad (7)$$

where  $M$  is the number of retained samples,  $x_i$  are the sampled configurations, and  $h$  is the kernel bandwidth. Repli-

cation weights are assigned inversely proportional to  $\rho(x_i)$ : samples in dense regions are duplicated less frequently, while those in sparse regions are upweighted. This reweighting encourages broader exploration of the configuration space, improving the final sample diversity.

---

#### Algorithm 1: Adaptive Diffusion Constrained Sampler

---

**Input:** Hyper-Parameter set  $H$ , Constraint set  $C$   
**Output:** Final batched samples  $q_0^{n_s}$

- 1 Initialize  $q_L^{n_s} \sim p_L(q)$ ;
- 2  $q_0^{n_s}, e_0^{n_s} \leftarrow \text{ALMC}(q_L^{n_s}, H, C)$ ; //  $\triangleright$  Algo. 2
- 3  $q_0^{n_r} \leftarrow \text{sort}(q_0^{n_s}, e_0^{n_s})$
- 4  $\rho(q) \leftarrow \text{KDE}(q_0^{n_r})$ ; // Estimate density
- 5  $w_i \propto 1/\rho(q_i)$ ; // Get the weights
- 6  $q_L^{n_s} \leftarrow \text{repeat}(q_0^{n_r}, w_i)$ ; // Repeat data
- 7  $q_0^{n_s}, e_0^{n_s} \leftarrow \text{ALMC}(q_L^{n_s}, H, C)$ ; // Final  $q_0^{n_s}$

---

#### Algorithm 2: Annealed Langevin Markov Chain Monte Carlo Sampler (ALMC)

---

**Input:**  $H, C$ , initial batched samples  $q_L^{n_s}$   
**Output:** Final batched samples  $q_0^{n_s}$

- 1 **for**  $k \leftarrow L$  **to** 1 **do**
- 2     Compute forward kinematics  $H_k$  and energy  $E(H_k, k, C)$ ;
- 3     Select step size  $\alpha_k$  and sample Gaussian noise  $\xi$ ;
- 4     Update samples by Langevin step:
- 5      $q_{k-1} \leftarrow q_k - \frac{\alpha_k^2}{2} \nabla E(H_k, k, C) + \alpha_k \xi$ ;
- 6 **Final refinement:** Apply Langevin updates with fixed step  $\alpha_f$  combined with Gauss-Newton refinement for a fixed numbers of iterations;
- 7 **return**  $q_0^{n_s}$

---

## V. EXPERIMENTAL RESULTS

We evaluate our method, **ADCS**, across a suite of constrained motion generation tasks and compare it to representative baselines: **Gauss-Newton** uses the predefined sampling cost, with updates computed as  $\Delta q = -(J^\top J)^{-1} J^\top r$ . **Diffusion-CCSP** [6] follows a similar energy-based formulation, with fixed (uniform) constraint weights and annealed

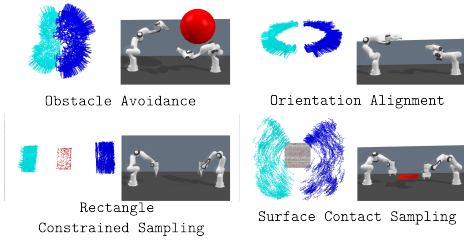


Fig. 2: Sampled configurations for simulation tasks. Each image shows the robots’ EE poses satisfying various constraints. Red dots indicate end-effector midpoints. Gray regions are constrained surfaces. Tasks increase in complexity from relative pose and midpoint constraints to full orientation and surface contact via SDFs.

Langevin dynamics as the sampler. *NLP-Sampling* uses the method from [4], note that its internal optimization loop is not externally controllable, so we specify only the number of final samples. All methods are evaluated over 10 random seeds (0–9), and we report average metrics across these runs. For fair comparison, Gauss–Newton and CCSP are run for 600 iterations, matching the sampling budget of ADCS.

#### A. Simulation Experiment

We design four simulation tasks featuring increasingly complex geometric and pose constraints (see Figure 2). All tasks are performed using 2 Franka robots and visualized using the Robotic package. For each task, we generate 500 samples using each method (batch size is 500).

All tasks involve a relative pose constraint between the two end-effectors and a midpoint constraint. (1) *Obstacle Avoidance*: relative pose + symmetric midpoint constraint and adds a spherical obstacle to avoid. (2) *Orientation Alignment*: adds orientation alignment with the table plane without obstacle, i.e., the EEF’s Y-axis is aligned with the world’s Z-axis. (3) *Rectangle Constrained Sampling*: midpoint lies inside a box (defined via bounds), with fixed EEF rotation matrix. (4) *Surface Contact Sampling*: replaces the region with a surface defined by a point cloud and loosens the orientation constraint to be normal to the surface, i.e., the EEF’s Z-axis is aligned with the negative world Z-axis.

In Table II, we report **mean, median, and third-quartile values for position and rotation errors** across four tasks. Under a fixed sampling budget (600 iterations), ADCS consistently yields significantly lower constraint satisfaction errors than Gauss–Newton and Diffusion-CCSP. Especially in tasks with smaller feasible regions, such as Rectangle Constrained Sampling, our method demonstrates stronger performance compared to the baselines. And compared to the state-of-the-art NLP-Sampling, it outperforms in most metrics while requiring shorter sampling time. In contrast, ADCS offers strong accuracy with significantly improved efficiency and broader applicability.

We also compare the **spatial quality of samples** generated by NLP Sampling, CCSP, and ADCS in Table I, evaluating both **coverage** and **uniformity** of the resulting distributions. Task *Obstacle Avoidance* and *Orientation Alignment* focus

	Data Distribution Coverage			
	Obstacle Avoidance	Orientation Alignment	Rectangle Constrained Sampling	Surface Contact Sampling
NLP Sampling	<b>0.0321</b>	0.0235	0.0425	-
CCSP	0.0310	0.0342	0.41	0.2407
ADCS (ours)	0.0315	<b>0.0362</b>	<b>0.6</b>	<b>0.6425</b>
	Data Distribution Uniformity(density variance)			
NLP Sampling	0.3388	0.5260	126.7000	-
CCSP	<b>0.1687</b>	0.1548	13.7969	5.0834
ADCS (ours)	0.3669	<b>0.1338</b>	<b>1.9097</b>	<b>1.5696</b>

TABLE I: Comparison of **coverage as sampled** and **data uniformity** by NLP Sampling, CCSP and ADCS.

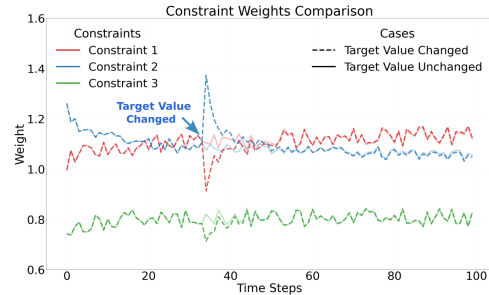


Fig. 3: **Dynamic performance of CWT** with three different constraint weights under two scenarios: (i) changing target values during inference, and (ii) fixed target values.

on the 3D position of a single end-effector, while Task *Rectangle Constrained Sampling* and *Surface Contact Sampling* assess the midpoint distribution between two arms. Coverage is computed as the fraction of occupied voxels in a predefined 3D grid, while uniformity is measured by the variance in density across voxels, lower variance indicates a more even spread of samples. As shown in the table, ADCS achieves superior coverage in most tasks and lower variance in most cases, highlighting its ability to explore constraint-satisfying regions more thoroughly.

In order to validate the **dynamic performance of CWT**, we intentionally modified the target value of one constraint during inference, such as the desired relative pose between the two EEFs, which causes the sampled points to fall outside the feasible region of that constraint, thereby testing the model’s adaptability. As illustrated in Figure 3, during inference, a change in the constraint target value leads to a corresponding adjustment of the weights, which enables the system to quickly recover back to the feasible region.

#### B. Real-World Experiment

We have designed eight different tasks that integrate our ADCS with motion planning. Under various specified constraints, these eight tasks involve stippling operations, i.e., two Franka robots collaboratively grasp a pen to perform different stippling tasks as shown in Figure 4. In addition, we conducted tests on the TIAGo robot, allowing it to carry objects using both hands.

To integrate with motion planning, we first perform radius-based thinning on the original sampled points to filter out all points within a specified radius of the selected points. We then sort the remaining points using different, task-specific sampling and sorting strategies. After sorting, we inspect the distance between consecutive points in the joint space. Whenever the distance exceeds a preset threshold, we use

	RPE (mm)		MPE (mm)		RRE (rad)		ERE (rad)		Time (s) Mean
	Median	Q3	Median	Q3	Median	Q3	Median	Q3	
Obstacle Avoidance									
NLP Sampling	0.0180	0.0660	0.0070	0.0292	4.652e-5	0.0001	-	-	61.4200
Gauss-Newton	<b>0.0001</b>	0.0003	7.649e-5	0.0001	1.718e-7	3.215e-7	-	-	<b>6.5329</b>
CCSP	5.7985	23.6867	9.9820	31.7004	0.0128	0.0373	-	-	9.1025
ADCS (ours)	<b>0.0001</b>	<b>0.0002</b>	<b>6.837e-5</b>	<b>8.652e-5</b>	<b>1.560e-7</b>	<b>2.149e-7</b>	-	-	10.3321
Orientation Alignment									
NLP Sampling	0.0358	0.0639	0.0177	0.0350	7.012e-5	0.0001	0.0002	0.0003	72.8657
Gauss-Newton	13.5566	109.0678	2.1668	46.6404	0.0366	0.3812	3.576e-07	0.0185	<b>7.6832</b>
CCSP	12.5211	38.7049	11.1668	38.5522	0.0101	0.0309	0.0341	0.0671	9.3460
ADCS (ours)	<b>0.0050</b>	<b>0.0060</b>	<b>0.0026</b>	<b>0.0029</b>	<b>1.495e-5</b>	<b>1.611e-5</b>	<b>1.192e-7</b>	<b>1.788e-7</b>	10.0325
Rectangle Constrained Sampling									
NLP Sampling	0.0176	0.0712	1.0019	4.4726	0.00003	0.00009	0.00001	0.00004	61.3578
Gauss-Newton	0.0008	121.098	0.3174	9.4909	1.412e-06	0.2704	5.277e-7	3.418e-5	<b>7.7406</b>
CCSP	170.4515	257.2366	21.8248	56.5533	0.0458	0.1007	0.1334	0.2218	10.3574
ADCS (ours)	<b>0.0001</b>	<b>0.0002</b>	<b>0.1690</b>	<b>0.2267</b>	<b>1.727e-7</b>	<b>2.394e-7</b>	<b>1.334e-7</b>	<b>1.883e-7</b>	11.3247
Surface Contact Sampling									
NLP Sampling	-	-	-	-	-	-	-	-	-
Gauss-Newton	66.8540	247.1499	4.4389	44.8222	0.0875	0.5902	0.0003	0.0188	<b>8.0048</b>
CCSP	23.7275	58.6257	11.3875	23.6884	0.0282	0.0691	0.0710	0.1411	13.7951
ADCS (ours)	<b>0.0010</b>	<b>0.0013</b>	<b>1.8522</b>	<b>2.8253</b>	<b>7.439e-7</b>	<b>9.614e-7</b>	<b>0</b>	<b>5.960e-8</b>	14.9514

TABLE II: Comparison of Gauss-Newton, CCSP, and ADCS across 4 tasks. For each task and method, we report RPE (relative position error (mm)), MPE (mid-point position error (mm)), RRE (relative rotation error (rad)), and ERE (end-effector rotation error in the world frame (rad)) by median and third quartile (Q3), and average computation time (s).



Fig. 4: The real-world experimental setup involves two Franka robots collaboratively grasping a pen for stippling operations and a TIAGo to carry objects using both hands.

the Gauss-Newton [32] method to project to obtain a closer joint configuration, which usually converges in just two to three iterations.

As illustrated in Figure 5, we evaluate eight real-world tasks. All tasks involve a relative pose constraint between the two end-effectors, as well as orientation constraints for the end-effectors. (1) *Fixed Midpoint with Orientation Alignment*: constrains the midpoint between the two arms to a fixed location. (2) *Circular Constrained Stippling* and (3) *Inclined Circular Constrained Stippling*: restrict the midpoint to lie within a (inclined) circular bounding box. (4) *Rectangle Constrained Stippling* and (5) *Inclined Rectangle Constrained Stippling*: restrict the midpoint to lie within a (inclined) rectangle bounding box. (6) *Letter Pattern Stippling* requires the midpoint to follow point clouds generated from mesh models of letters. Finally, (7) *Cube Surface Stippling* and (8) *Cylinder Surface Stippling*: constrain the

	RPE	MPE	RRE	ERE	VPR
Fixed Midpoint with Orientation Alignment	6.9647	1.6872	0.0074	0.0042	72.60
Circular Constrained Stippling	0.0706	0.0024	0.0001	4.27e-05	98.40
Inclined Circular Constrained Stippling	0.2818	0.0018	0.0003	0.0001	88.40
Rectangle Constrained Stippling	0.1541	0.0097	0.0002	0.0002	90.60
Inclined Rectangle Constrained Stippling	0.6734	0.0312	0.0006	0.0007	84.80
Letter Pattern Stippling	2.7895	0.0316	0.0085	0.0101	80.40
Cube Surface Stippling	2.0677	0.0387	0.0055	0.0063	82.45
Cylinder Surface Stippling	2.3240	0.0282	0.0058	0.0087	80.70

TABLE III: The constraint errors (Q3) and the ratio of valid points (VPR (%)) of different tasks in the real-world.

midpoint to the top surface of real objects, with point clouds obtained directly from a camera.

In Table III, we present the constraint error and the ratio of valid points for each task. Note that each task was tested ten times, and in every task, 500 points were generated through sampling. We consider a point valid if its positional error is less than 3 mm and its orientational error is less than 0.005 rad. As shown, the probability of valid points exceeds 80% in most tasks.

### C. Generalization and Robustness Capability.

To evaluate the generalization capability of our model, we tested the constraint values of the In-Distribution (ID) and Out-of-Distribution (OOD) data under different tasks. In Task *Orientation Alignment*, we set the relative pose to values out of distribution and the different positions of the midpoint together. In Task *Rectangle Constrained Sampling*, we tested

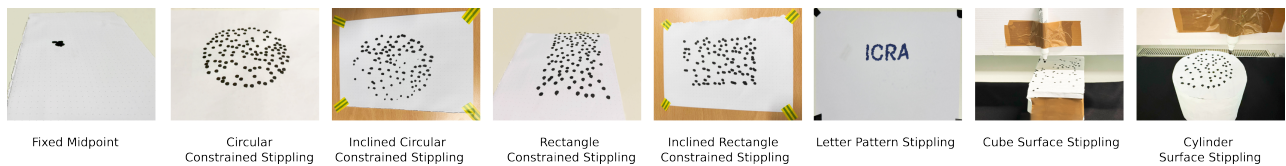


Fig. 5: Final stipling layout of 8 different real-world tasks.

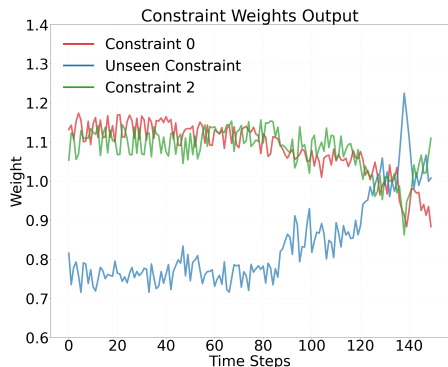


Fig. 6: Constraint weight outputs with an **unseen constraint**.

	RPE	MPE	RRE	ERE
Orientation Alignment				
ID	0.0060	0.0029	1.6e-5	-
OOD	0.0535	0.0159	9.8e-5	-
Rectangle Constrained Sampling				
ID	0.0002	0.2267	2.3e-7	1.8e-7
OOD	0.0002	0.2775	2.3e-7	1.9e-7

TABLE IV: **Sampling results of both ID and OOD datasets** to demonstrate the generalization capability. The errors are reported in Q3.

the different bounding box sizes. As shown in Table IV, even when we input data from outside the training set, our model can still produce valid sampling results.

In addition, we evaluated the **generalization ability of CWT** to unseen constraints. We trained an additional energy network corresponding to an unseen constraint. As illustrated in Figure 6, CWT is able to assign reasonable weights even to unseen constraints, thereby ensuring that the generated samples remain compliant with the imposed constraints.

#### D. Ablation Studies

We conduct ablations to evaluate the contributions of key components in our method. Specifically, we analyze: (1) The effect of the CWT for dynamic constraint composition. (2) The role of KDE for diversity-aware resampling. (3) The impact of two-stage sampling strategies.

**Dynamic Weighting via CWT.** We compare three composition mechanisms for constraint weighting: fixed scalar weights, MLP-based dynamic weights, and our proposed CWT. All models share the same architecture for the energy and feature networks and are trained without task likelihoods to isolate the effect of the weighting mechanism. As shown in Table V, CWT consistently outperforms the other approaches, achieving lower errors across all tasks, underscoring its ability to adaptively balance constraint satisfaction in

	RPE	MPE	RRE	ERE
Orientation Alignment				
Fixed Weights	42.5589	43.4433	0.0403	0.0660
MLP-based	40.9558	43.1051	0.0403	0.0668
CWT	<b>35.0300</b>	<b>34.7487</b>	<b>0.0337</b>	<b>0.0640</b>
Surface Contact Sampling				
Fixed Weights	63.3622	27.1177	0.0682	0.1195
MLP-based	73.4915	27.7264	0.0724	0.1014
CWT	<b>62.1286</b>	<b>20.2202</b>	<b>0.0393</b>	<b>0.0857</b>

TABLE V: Comparison of constraint composition mechanisms in ADCS: fixed weights, MLP-based dynamic weights, and the proposed CWT. And there is no two-stage sampling involved in this ablation and all variants use the same energy and feature networks and exclude task likelihood information during training. The errors are reported in Q3.

Data Distribution Uniformity (Density Variance)		
	After Replication	Final Sampled
Uniform replication	78.0000	79.6028
KDE replication	<b>50.4320</b>	<b>57.4560</b>

TABLE VI: The table presents the variance of the end-effector’s positional **density distribution for the uniform-replication and KDE-based-replication methods**, both after the resampling stage and after the final sampling. A smaller variance value indicates a more uniform distribution.

diverse compositions.

**KDE-based Resampling.** Under Task *Orientation Alignment*, we compared KDE-based resampling against uniform-replication resampling. We estimate the data distribution density with a voxel size of 5. From Table VI, we can conclude that the KDE-based resampling method achieves a more uniform data distribution both after the resampling stage and in the final sampling results.

**Two-stage Batch-wise Sampling.** To evaluate the performance gains brought by two-stage sampling, we tested ADCS and ADCS without two-stage sampling on three different tasks. And each task uses the same threshold, a sample point is considered valid if its error is smaller than the threshold. As shown in Table VII, for each task, two-stage sampling consistently yields performance improvements, especially on tasks with smaller feasible regions.

	Valid Sample Rate (%)		
	Obstacle Avoidance	Orientation Alignment	Rectangle Constrained Sampling
w/o Two-stage	82.3	3.74	1.54
Two-stage	<b>99.46</b>	<b>99.24</b>	<b>99.7</b>

TABLE VII: Comparison of **Valid Sample Rates for ADCS with and without two-stage Sampling**.

**Limitations.** In this work, the SE(3) constraints are defined by manually specifying the corresponding numerical values, rather than through instructions like “align both hands”. An interesting direction would be to combine this with Vision-Language Models (VLMs) and define the related constraints through Large Language Models (LLMs). Furthermore, LLMs could be employed to define cost functions for the Gauss-Newton refinement, thereby allowing a more general optimization framework. Lastly, the current training process does not incorporate information about the corresponding joint configurations. Future improvements include joint information during the training phase to improve model performance and accelerate convergence during inference.

## VI. CONCLUSION

In this paper, we introduced ADCS, a generative framework designed to effectively integrate equality and inequality geometric constraints within multi-DoF robot manipulation tasks. Our approach uses a dedicated energy network to calculate the cost of equality and inequality constraints, and introduces an SDF network to incorporate external environment perception, while combined with a Transformer-based architecture for dynamically weighting these constraints during inference. Experimental evaluations across both simulation and real-world tasks demonstrated that ADCS outperforms baseline methods, also applicable to scenarios with small feasible areas and strict spatial constraints. Additionally, through ablation studies, we verified that transformer-based dynamic weights, KDE-based resampling, and batch-wise constrained sampling significantly enhance sample uniformity and performance. Moreover, our sampler can be used to construct a roadmap for enabling motion planning. Finally, our method’s robust generalization was validated on out-of-distribution data, confirming its adaptability and efficiency in realistic robotic applications.

## REFERENCES

- [1] J. Canny, *The complexity of robot motion planning*. MIT press, 1988.
- [2] O. Khatib, “Mobile manipulation: The robotic assistant,” *Robotics and Autonomous Systems*, vol. 26, no. 2-3, pp. 175–183, 1999.
- [3] Z. Kingston, M. Moll, and L. E. Kavraki, “Sampling-based methods for motion planning with constraints,” *Annual review of control, robotics, and autonomous systems*, vol. 1, no. 1, pp. 159–185, 2018.
- [4] M. Toussaint, C. V. Braun, and J. Ortiz-Haro, “Nlp sampling: Combining mcmc and nlp methods for diverse constrained sampling,” *arXiv preprint arXiv:2407.03035*, 2024.
- [5] J. Urain, N. Funk, J. Peters, and G. Chalvatzaki, “Se (3)-diffusionfields: Learning smooth cost functions for joint grasp and motion optimization through diffusion,” in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5923–5930.
- [6] Z. Yang, J. Mao, Y. Du, J. Wu, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling, “Compositional diffusion-based continuous constraint solvers,” *arXiv preprint arXiv:2309.00966*, 2023.
- [7] C. Suh, T. T. Um, B. Kim, H. Noh, M. Kim, and F. C. Park, “Tangent space rrt: A randomized planning algorithm on constraint manifolds,” in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 4968–4973.
- [8] B. Kim, T. T. Um, C. Suh, and F. C. Park, “Tangent bundle rrt: A randomized algorithm for constrained motion planning,” *Robotica*, vol. 34, no. 1, pp. 202–225, 2016.
- [9] Y. Chen, R. Dwivedi, M. J. Wainwright, and B. Yu, “Fast mcmc sampling algorithms on polytopes,” *Journal of Machine Learning Research*, vol. 19, no. 55, pp. 1–86, 2018.

- [10] J. Ho, A. Jain, and P. Abbeel, “Denoising diffusion probabilistic models,” *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
- [11] Q. Liu and D. Wang, “Stein variational gradient descent: A general purpose bayesian inference algorithm,” *Advances in neural information processing systems*, vol. 29, 2016.
- [12] J. Urain, A. Mandlekar, Y. Du, M. Shafiqullah, D. Xu, K. Fragkiadaki, G. Chalvatzaki, and J. Peters, “Deep generative models in robotics: A survey on learning from multimodal demonstrations,” *arXiv preprint arXiv:2408.04380*, 2024.
- [13] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint, “Structured deep generative models for sampling on constraint manifolds in sequential manipulation,” in *Conference on Robot Learning*. PMLR, 2022, pp. 213–223.
- [14] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters, “Motion planning diffusion: Learning and planning of robot motions with diffusion models,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 1916–1923.
- [15] V. Kurtz and J. W. Burdick, “Equality constrained diffusion for direct trajectory optimization,” *arXiv preprint arXiv:2410.01939*, 2024.
- [16] C. Pan, Z. Yi, G. Shi, and G. Qu, “Model-based diffusion for trajectory optimization,” *Advances in Neural Information Processing Systems*, vol. 37, pp. 57914–57943, 2024.
- [17] Y. Zhang, C. Pezzato, E. Trevisan, C. Salmi, C. H. Corbato, and J. Alonso-Mora, “Multi-modal mppi and active inference for reactive task and motion planning,” *IEEE Robotics and Automation Letters*, 2024.
- [18] A. R. Kumar and P. J. Ramadge, “Diffloop: Tuning pid controllers by differentiating through the feedback loop,” in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*. IEEE, 2021, pp. 1–6.
- [19] S. Cheng, M. Kim, L. Song, C. Yang, Y. Jin, S. Wang, and N. Hovakimyan, “Diffune: Auto-tuning through auto-differentiation,” *IEEE Transactions on Robotics*, 2024.
- [20] A. Marco, P. Hennig, J. Bohg, S. Schaal, and S. Trimpe, “Automatic lqr tuning based on gaussian process global optimization,” in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 270–277.
- [21] A. Loquercio, A. Saviolo, and D. Scaramuzza, “Autotune: Controller tuning for high-speed flight,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4432–4439, 2022.
- [22] Y. Song and S. Ermon, “Improved techniques for training score-based generative models,” *Advances in neural information processing systems*, vol. 33, pp. 12438–12448, 2020.
- [23] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, “Compositional visual generation with composable diffusion models,” in *European Conference on Computer Vision*. Springer, 2022, pp. 423–439.
- [24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [25] D. Ha, A. Dai, and Q. V. Le, “Hypernetworks,” *arXiv preprint arXiv:1609.09106*, 2016.
- [26] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool, “Dynamic filter networks,” *Advances in neural information processing systems*, vol. 29, 2016.
- [27] S. Zhong, T. Power, A. Gupta, and P. Mitrano, “PyTorch Kinematics,” 2024.
- [28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [29] M. Girolami and B. Calderhead, “Riemann manifold langevin and hamiltonian monte carlo methods,” *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 73, no. 2, pp. 123–214, 2011.
- [30] B. Leimkuhler and C. Matthews, “Rational construction of stochastic numerical methods for molecular sampling,” *Applied Mathematics Research eXpress*, vol. 2013, no. 1, pp. 34–56, 2013.
- [31] B. W. Silverman, *Density estimation for statistics and data analysis*. Routledge, 2018.
- [32] C. F. Gauss and C. H. Davis, “Theory of the motion of the heavenly bodies moving about the sun in conic sections,” *Gauss’s Theoria Motus*, vol. 76, no. 1, pp. 5–23, 1857.