FlashBack: Consistency Model-Accelerated Shared Autonomy

Luzhe Sun, Jingtian Ji, Xiangshan Tan, Matthew R. Walter
Toyota Technological Institute at Chicago
{luzhesun, jijingtian, vincenttann, mwalter}@ttic.edu

Abstract: Shared autonomy is an enabling technology that provides users with control authority over robots that would otherwise be difficult if not impossible to directly control. Yet, standard methods make assumptions that limit their adoption in practice—for example, prior knowledge of the user's goals or the objective (i.e., reward) function that they wish to optimize, knowledge of the user's policy, or query-level access to the user during training. Diffusion-based approaches to shared autonomy do not make such assumptions and instead only require access to demonstrations of desired behaviors, while allowing the user to maintain control authority. However, these advantages have come at the expense of high computational complexity, which has made real-time shared autonomy all but impossible. To overcome this limitation, we propose Consistency Shared Autonomy (CSA), a shared autonomy framework that employs a consistency model-based formulation of diffusion. Key to CSA is that it employs the distilled probability flow of ordinary differential equations (PF ODE) to generate high-fidelity samples in a *single* step. This results in inference speeds significantly than what is possible with previous diffusion-based approaches to shared autonomy, enabling real-time assistance in complex domains with only a single function evaluation. Further, by intervening on flawed actions at intermediate states of the PF ODE, CSA enables varying levels of assistance. We evaluate CSA on a variety of challenging simulated and real-world robot control problems, demonstrating significant improvements over state-of-the-art methods both in terms of task performance and computational efficiency. Our code is available at https://ripl.github.io/CSA-website

Keywords: Consistency Model, Shared Autonomy, ODE Distillation

1 Introduction

Shared autonomy is a collaborative control paradigm where a human operator and an autonomous agent jointly control a robotic system to achieve common objectives [1, 2, 3, 4, 5]. By complementing human intuition with machine precision, shared autonomy enhances performance, ensures safety, and reduces operator workload. This is particularly valuable in complex control scenarios where it enables a human **pilot** to provide high-level input while the robotic **copilot** autonomously manages low-level motion corrections to maintain safety and operational efficiency [6, 7, 8, 9, 10, 11].

Traditionally, shared autonomy algorithms have assumed the existence of a fixed set of known goals from which the user's goal is be inferred at test time based on their input [12, 13, 14, 15]. While effective in some settings, these approaches struggle in unstructured and semi-structured settings where the space of goals is not well defined. Recent advancements in generative modeling, particularly diffusion probabilistic models that have revolutionized domains such as image, audio, and 3D generation [16, 17, 18, 19, 20, 21, 22, 23, 24], offer a promising avenue to address these challenges by framing shared autonomy as sampling from a learned distribution over actions [25, 1, 26].

In shared autonomy specifically, diffusion-based methods have demonstrated effectiveness in stabilizing user actions without explicit assumptions regarding the goal space. For instance, Yoneda et al. [1] introduced partial diffusion with noise injection to maintain user intention, and Yin et al.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

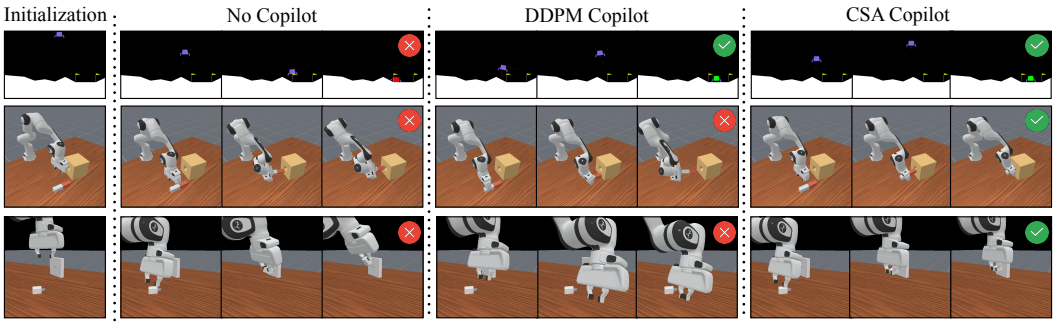


Figure 1: A visualization of the result of using our proposed Consistency Shared Autonomy (CSA) algorithm in comparison to the state-of-the-art DDPM-based shared autonomy baseline on three challenging control tasks.

[27] utilized motion gradients as denoising guidance for context-rich manipulation tasks. However, deploying diffusion models in real-time robotics applications remains challenging due to the computational cost of the diffusion process. Diffusion-denoising probabilistic models (DDPMs) [16] solve a reverse-time stochastic differential equation (SDE) through a denoising process, however this process typically requires hundreds of steps. This can lead to high time complexity for generation/inference, which precludes real-time deployment. Further, DDPMs employed for shared autonomy are prone to generating samples that may ignore information provided by the user. Thus, real-time shared autonomy faster generative methods that respect the user's input.

Consistency models (CM) [28] have emerged as a compelling solution to these concerns. Rather than iteratively removing noise as in traditional diffusion models, CM directly learns a denoising function to "jump" efficiently to high-quality results. Leveraging a pretrained ordinary differential equation (ODE)-based probability flow from EDM models [18], CM effectively distills the complex ODE flow into rapid, one-step (or few-step) generative processes without sacrificing fidelity. Empirical successes in image synthesis demonstrate the resulting improvements in efficiency. Notably, Song et al. [28] demonstrates state-of-the-art image generation performance using significantly fewer sampling iterations compared to standard diffusion methods. Similarly, consistency policies applied in robotic planning have achieved inference speeds up to 100 times faster than traditional DDPM-based approaches while maintaining competitive performance [29, 25].

Motivated by the use of partial diffusion as a means of balancing user control with task performance [1, 30] and the computational efficiency of ODE-based distillation methods [28], we propose the Consistency Model for Shared Autonomy (CSA, illustrated in Fig. 1). Unlike contemporary shared autonomy methods, CSA provides accelerated inference, ensures proximity to nearest expert actions, and requires minimal training data. We evaluate CSA on a series of simulated and real-world control tasks, demonstrating its improvements over the previous state-of-the-art.

2 Related Work

Early approaches to shared autonomy assumed prior knowledge of the user's goal [31, 32] or that it could be explicitly inferred from user input [33, 12, 13, 14, 15]. Reddy et al. [7] introduced model-free deep reinforcement learning (RL) for shared autonomy, removing the requirement for known environment dynamics; subsequent works further developed this approach [8, 34].

Diffusion model-based action planning has proven powerful in policy learning [25, 35, 29] as well as shared autonomy. Yoneda et al. [1] propose a *partial diffusion* mechanism that adds noise to the user's action and then employs a partial reverse diffusion process to refine the action towards the training distribution, thereby preserving the user's intent (*fidelity*) while correcting the action (*conformity*). However, in such an SDE-based model, the injection of fresh noise at each reverse step is a double-edged sword: it increases diversity but can push the outcome into undesirable modes. DexterityGen (DexGen) [27] takes a different approach, forgoing partial diffusion in favor of a *user motion promotion* strategy to find a trajectory that aligns with the user's intention.

Despite these advances, both of the above diffusion-based approaches (using either a DDPM or DDIM formulation) are constrained by iterative inference. In contrast, CSA uses a one-step inference paradigm that enables microsecond-scale generation, requires only a small amount of training data, does not rely on explicit user goal prediction, yet is performant in high-precision tasks.

Compared to the standard formulation of diffusion (e.g., DDPM [16]) discussed above, Denoising Diffusion Implicit Models (DDiM) [17] recast the reverse process as a deterministic ordinary differential equation (ODE). EDM [18] unifies SDE and ODE sampling through improved preconditioning and step weighting. Notably, ODE-based sampling yields samples to their nearest neighbor along the ideal denoising path, a guarantee that SDE-based diffusion lacks. However, aggressively reducing the number of steps can still degrade output quality despite faster sampling.

Distillation techniques have been explored to further mitigate inference costs [36, 37]. These approaches typically involve training an ODE-based *teacher model* to generate detailed, high-quality trajectories, followed by training a *student model* that learns to approximate these trajectories with fewer intermediate steps. For instance, Consistency Models (CM), proposed by Song et al. [28], leverage the inherent self-consistency of ODE trajectories. Given two distinct intermediate states $\{x^u, x^v\}$ along the same trajectory, the CM enforces predictions to converge to an identical clean target $\hat{x}_0$. By optimizing this consistency, CMs facilitate rapid sampling in as few as one or several inference steps, significantly improving efficiency without sacrificing generation quality.

3 Method

Underlying our proposed method is the formulation of shared autonomy as a generative process, whereby we sample actions from a learned expert distribution that are consistent with user's latent intent. Diffusion models provide a compelling way to generate these samples since user-provided actions can be thought of as noisy inputs that lie on the trajectory swept via reverse diffusion (we refer the reader to Appendix 7.1 for a discussion of probability flow ODEs). This is the approach that we adopt here. However, as we have previously discussed, the computational cost of this reverse diffusion process precludes real-time operation critical to shared autonomy.

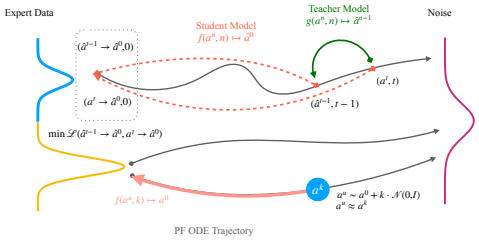


Figure 2: Distillation of PF ODE flow: Select two distinct states $\{a^t, a^{t-1}\}$ along the same trajectory, the CM enforces that predictions converge to the same target $\hat{a}^0$.

Rather than relying on many sequential denoising steps, CSA uses consistency-model distillation to collapse the entire ODE-based diffusion trajectory into a single, efficient step. Concretely, we first train a high-fidelity *teacher* diffusion model by solving the PF ODE iteratively, a process that produces reliable samples but requires dozens or more solver steps. During distillation, we then train a *student* model to learn a direct mapping from the teacher's initial noisy input all the way to its final clean output. In effect, the student "jumps" over the intermediate flow trajectory in one shot, approximating the teacher's multi-step refinement with a single forward pass (or very few steps) while retaining comparable sample quality.

3.1 Training Phase

We train our CSA model by first integrating state prediction into the EDM framework to serve as a teacher model. Subsequently, the consistency model (CM)–based denoiser CSA distills ODE-flow knowledge from this teacher to perform one-step denoising, substantially accelerating inference.

Teacher Model Training data consists of state-action-next state transitions (s, a, s_n), where s is the current state, a is the expert action, and s_n is the next state.

We preprocess the state into two conditions, $\text{cond}_1 = s$ and $\text{cond}_2 = \frac{s_n - s}{\|s_n - s\|_2}$, where the latter measures the direction in which the state is changing as an indication of near-term user intent. During training, we may condition solely on $\{\text{cond}_1\}$ or jointly on $\{\text{cond}_1, \text{cond}_2\}$. Following the EDM

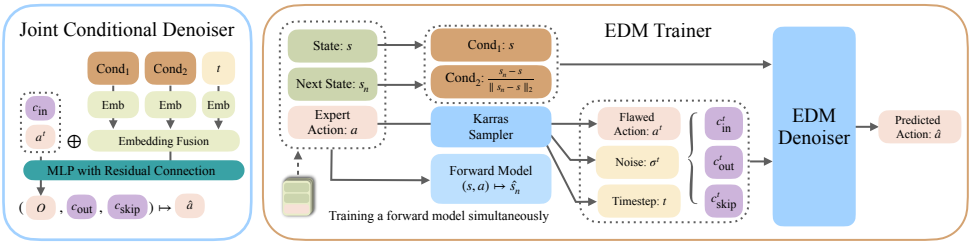


Figure 3: Training Process of EDM (teacher) model.

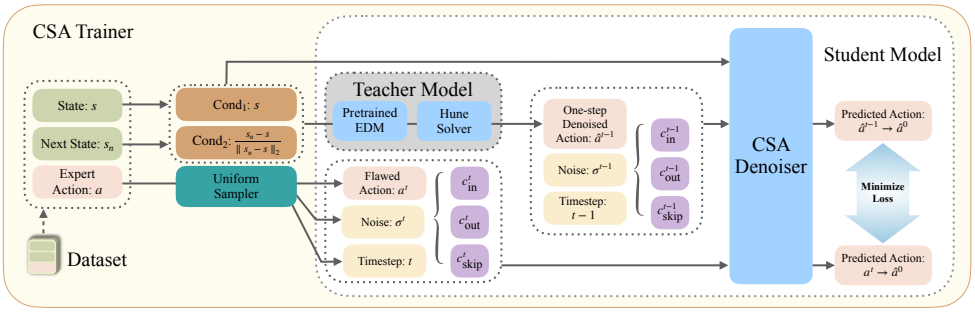


Figure 4: Training Process of CSA (student) model.

framework, we employ the Karras sampler [18] to partition a noise schedule $\{\sigma^0, \ldots, \sigma^t\}$.[1] At noise step t, we obtain a perturbed action a^t by injecting noise $\sigma^t \cdot z, z \sim \mathcal{N}(0, 1)$. The hyperparameters $\{c_{\text{in}}, c_{\text{out}}, c_{\text{skip}}\}$ are functions of the noise σ^t (Appendix 7.4). We use the joint conditional denoiser $D(\cdot)$ to remove noise (Fig. 3). The model embeds all conditions $\{\text{Cond}_1, \text{Cond}_2, t\}$ using the same hidden dimension as the action embedding, sums them, and then passes them into a three-layer MLP to predict the output O. Each layer incorporates conditioning embeddings via residual connections.

$$O = F_\theta(c_{\text{in}} \cdot a^t, t, \text{cond}_1, \text{Optional}(\text{cond}_2)) \tag{1a}$$

$$D = c_{\text{out}} \cdot O + c_{\text{skip}} \cdot a^t \tag{1b}$$

A learnable adaptive weighting model $\lambda(\cdot)$ is introduced to track the loss in EDM $\mathcal{L}_{\text{EDM}}(\theta) = \mathbb{E}_{(t,a,a^t|a)}[\text{dist}(\lambda(\sigma^t), a, D(\cdot))]$, where

$$\mathcal{L} = \text{dist}(\lambda(\sigma^t), a, D(\cdot)) = e^{\lambda(\sigma^t)} \cdot \|(a - D(c_{\text{in}}, c_{\text{out}}, c_{\text{skip}}, a^t, t, \text{cond}_1, \text{cond}_2))\|_2 - \lambda(\sigma^t) \tag{2}$$

We use conditional loss and joint conditional loss to track the training metric of $\{\text{cond}_1\}$ and $\{\text{cond}_1, \text{cond}_2\}$ situation, during loss compute we drop out the Cond_2 with a probability γ.

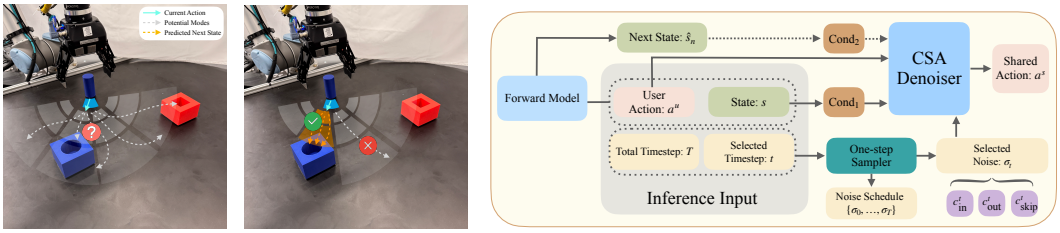
$$\mathcal{L}_{\text{EDM}}(\theta) = (1 - \gamma)L_{\text{joint}-\text{cond}} + \gamma L_{\text{cond}} \tag{3}$$

EDM model inference also employs a discretized numerical solver; as a result, its recursive process remains slow.

Forward model At inference, the true next state s_n cannot be observed without executing the predicted action $\hat{a}$. To overcome this, we concurrently train a forward model $\Phi(s, a) \mapsto \hat{s}_n$, implemented as an MLP with normalization layers.

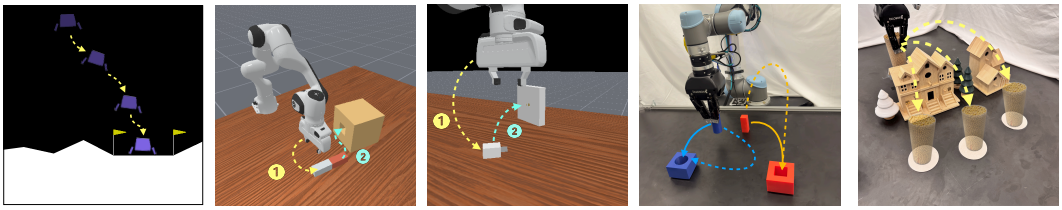
Student Model As visualized in Fig. 2, the distilling training strategy involves finding two samples $\{a^i, a^j\}, i, j \in \{0, \ldots, T\}$ on the same PF ODE trajectory, learning a student denoiser $f(a^t, t, o^t) \mapsto \hat{a}^0$, where o^t denotes all other conditions including $\{c^t_{\text{in}}, c^t_{\text{out}}, c^t_{\text{skip}}, \text{cond}_1, \text{cond}_2, \sigma^t\}$. We denote the process of denoising $\{a^i, a^j\}$ as $f(a^i, i, o^i) \mapsto \hat{a}^{i0}$, $f(a^j, j, o^j) \mapsto \hat{a}^{j0}$. Since the trajectory is a deterministic ODE flow, they should trace back to the same starting point. Then, we can formulate the objective as one of minimizing the distance between them , i.e., $\min \text{MSE}(\hat{a}^{i0}, \hat{a}^{j0})$.

[1] Throughout the paper, we use superscripts to denote diffusion steps. In Fig. 3, we label these as noise steps (which are the same as diffusion time steps) to avoid ambiguity.



(a) Forward model for next state prediction (b) Inference phase of CM model

Figure 5: Using forward model during inference time



(a) Lunar Lander (b) Peg Insertion (c) Charger Insertion (d) Real Peg (e) Fluid Carrying

Figure 6: Environment Setting

In practice, we choose $i = j - 1$ and use t and $t - 1$ in the following expression. The challenge lies in sampling a^t and a^{t-1} from the same flow. We start from expert action a^0, inject noise $\sigma^t \cdot z$ to get a^t. To get a^{t-1} on the same flow we will use the teacher model $g(\cdot)$ (pretrained EDM denoiser) with a Huen solver: $g(a^t, t, o^t) \mapsto \hat{a}^{t-1}$. The loss function can now be defined as:

$$\mathcal{L}_{\text{CSA}}(\theta) = \text{MSE}\left(f(a^t, t, o^t), f(\hat{a}^{t-1}, t-1, o^{t-1})\right) \tag{4}$$

3.2 Inference

Unlike previous methods [1, 29, 25], the inference phase of CSA does not start from Gaussian noise nor inject noise to the user sample. Given the user's action a^u, we assume $a^u \sim \mathcal{N}(a, \sigma^t)$ where t is a hyperparameter determines at which step of the ODE flow the user's action corresponds to (Fig. 5b). We define $\alpha = \frac{t}{T}$ which is controlling the amount of noise we assume, this regulates the balance between the fidelity and the conformity of the generated actions. Using the pretrained forward model, we can get a next state estimation based on user action a^u and current state s, $\Phi(a^u, s) \mapsto \hat{s}_n$, providing a short term user intention. Process $\{s, \hat{s}_n\}$ to $\{\text{cond}_1, \text{cond}_2\}$ serve as condition for CSA model. Then we will use One-step Sampler to partition the total noise into T segments, choose noise step t and the noise level σ^t that matches a^u. Finally apply CM noiser $f(a^u, t, o^t) \mapsto \hat{a}^0$ to finish the one step denosing. We refer to the model conditioned solely on $\{\text{cond}_1\}$ as CSA, and the variant conditioned on $\{\text{cond}_1, \text{cond}_2\}$ as CSA†. We show that CSA† preserves the user's intent and thereby broadens the effective inference "sweet spot."

4 Experiments

We evaluate the performance of our proposed CSA model in simulation on a several continuous control tasks as well as through real-robot experiments.

4.1 Evaluation Tasks

We consider the following four continuous control domains as a means of comparing CSA to a contemporary DDPM-based baseline. Appendix 7.5 provides further experimental details.

(a) Lunar Lander: The Lunar Lander environment (Fig. 6a) is a two-dimensional, continuous-control task adopted from Open AI Gym [38] that requires landing a spacecraft on a fixed landing pad. An episode ends when it lands safely, crashes, drifts off course, or when the episode times out after 1000 steps.

(b) Peg Insertion: Peg Insertion (Fig. 6b) tasks a robotic arm with inserting a 10 cm long, 2 cm radius peg into a hole that affords only 1 cm of clearance. The poses of the peg and the target box

are randomized at the start of each episode. Success is declared when the peg tip enters the hole more than 1.5 cm; otherwise the episode terminates after 200 steps.

(c) Charger Plug Insertion: Charger Plug Insertion (Fig. 6c) tightens the tolerance further: a dual-peg charger must slide into its receptacle with only 0.5 mm of clearance. The poses of the charger and socket are randomized for each episode, and the agent again controls the end-effector using compact kinematic and visual inputs. An episode is determined to be successful if the plug is inserted within 5 mm and 0.2 radians of its target pose. Episodes are terminated upon success, when there is a high-impact collision, or if they time out after 300 steps.

(d) Real Peg Insertion: Real Peg Insertion (Fig. 6d) is performed with a UR5 robot arm equipped with a Robotiq 3-Finger gripper. The task involves inserting either a red square peg or a blue cylinder peg into its corresponding hole, with clearance tolerances of 5 mm and 4.5 mm, respectively.

(e) Fluid Carrying: Real Robot Fluid Carrying (Fig. 6e) uses the same robotic setting as Real Peg Insertion. The task aims to assist user to move a cup full with soybeans to one of the three potential target based on user's intent smoothly. Any spillage will consider as a failure case.

4.2 Data Collection

Training the assistive policy requires an oracle operator to provide expert transitions (s_t, a_t, s_{t+1}). We use Soft Actor-Critic (SAC) [39] algorithm to train expert policy for Lunar Lander and Peg Insertion, and Proximal Policy Optimization (PPO) [40] combined with curriculum learning (CL) to train the expert for Charger Plug Insertion, relaxing the success check to 1.8 cm then gradually shrinking down to 5 mm to get an expert for this high precision task [41]. After we get the expert policy, we apply rejection sampling to collect transits only from successful episodes. All three simulation experiments, regardless of hardness, are trained with 200K transitions. For the Real Peg Insertion task, demonstrations are collected by experienced human teleoperators using a Meta Quest 3 VR controller. We collected 180 demonstrations for a total of 62K transitions in the same form of (s_t, a_t, s_{t+1}). This process takes 1 hour with 2 operators. For Fluid Carrying task, we used a scripted motion planner to collect 150 rollouts for a total of 90K transitions in around 50 minutes.

4.3 Surrogate Pilots

While CSA does not require access to a pilot (real or surrogate) for training, we use surrogate pilots in order to scale the number of evaluations. Building off previous work [7, 8, 1], we define four different surrogates *noisy*, *laggy*, *noised*, and *slow* as corrupted versions of an expert policy π.

$$\text{Noisy}\,(s_t, g) = \begin{cases} a_t^e, & \text{if } p \geq \epsilon \\ a_t^r, & \text{if } p < \epsilon \end{cases} \qquad \text{Laggy}\,(s_t, g) = \begin{cases} a_t^e, & \text{if } p \geq \epsilon \\ a_{t-1}^e, & \text{if } p < \epsilon \end{cases}$$

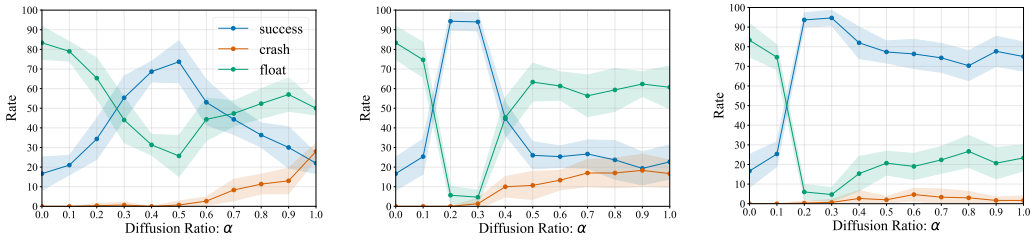
$$\text{Noised}\,(s_t, g) = a_t^e + \mathbb{N}(0, \epsilon) \qquad \text{Slow}\,(s_t, g) = (1 - \epsilon) \cdot a_t^e$$

where $\epsilon \in [0, 1]$ is a *flaw* parameter that determines the amount of corruption, the subscript t is the timestep (in episode), a_t^e is an expert action, a_t^r is a random action, s_t is the state and g refers to goal. We evaluate four surrogates under different flaw levels ϵ with 10 random seeds and 30 rollouts each. We provide the parameter ϵ values in the Appendix 7.3 Table 4.

4.4 Evaluation Results

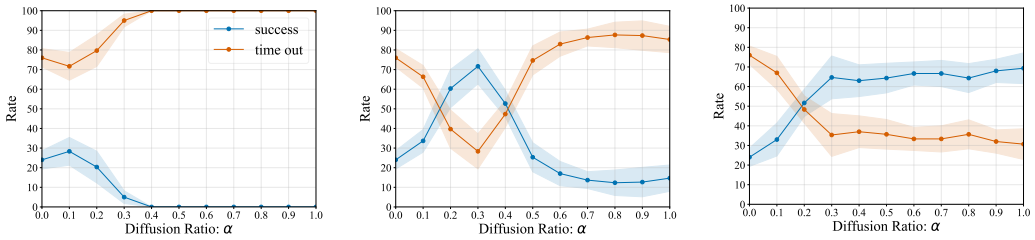
Simulation Result: Given that our method is a goal-excluded, state-conditional shared control model, we compare against the SDE-based DDPM baseline of [1], adopting their forward diffusion ratio as α to trade off fidelity and conformity. Figures 7 and 8 show that in the low-dimensional Lunar Lander task, CSA achieves performance on par with the baseline, while in the higher-dimensional Peg Insertion task , CSA substantially outperforms it—where the DDPM policy yields under a 10% gain under noise.

In the high-precision Charger Plug Insertion task, we tested the DDPM baseline across ten noise schedules ($\beta_{\max} \sim \text{Unif}[0.01, 0.25]$) and observed zero successful runs, underscoring its acute sensitivity to schedule tuning and the associated computational cost. By contrast, CSA attains robust



(a) DDPM Policy (b) CSA (c) CSA†

Figure 7: Lunar Lander Noised Simulation Result



(a) DDPM Policy (b) CSA (c) CSA†

Figure 8: Peg Insertion Noisy Simulation Result

Actor	Method	Success Rate (%) ↑	Crash Rate (%) ↓	NFE	Inference Time (ms) ↓
Laggy	Surrogate	40.00 ± 4.16	52.66 ± 4.92	–	–
	DDPM	75.67 ± 9.30	14.67 ± 6.32	24	13.62 ± 0.22
	CSA	**87.67 ± 5.89**	6.67 ± 4.16	1	0.92 ± 0.04
	CSA†	**91.00 ± 3.87**	**5.00 ± 3.24**	1	1.22 ± 0.30
Noisy	Surrogate	26.00 ± 6.44	6.67 ± 3.85	–	–
	DDPM	55.67 ± 7.86	4.67 ± 3.58	24	17.67 ± 1.49
	CSA	**85.00 ± 8.05**	2.67 ± 3.44	1	1.79 ± 0.24
	CSA†	**89.00 ± 4.17**	**2.33 ± 2.25**	1	2.03 ± 0.35
Noised	Surrogate	16.67 ± 8.46	0.00 ± 0.00	–	–
	DDPM	73.67 ± 10.71	10.43 ± 2.11	24	18.60 ± 1.51
	CSA	**94.33 ± 4.46**	**0.00 ± 0.00**	1	1.94 ± 0.25
	CSA†	**94.67 ± 3.91**	0.67 ± 1.41	1	1.82 ± 0.59
Slow	Surrogate	75.33 ± 7.57	18.33 ± 5.93	–	–
	DDPM	**92.67 ± 4.09**	**3.00 ± 2.92**	14	10.44 ± 0.07
	CSA	80.33 ± 5.32	11.33 ± 6.13	1	1.72 ± 0.27
	CSA†	79.00 ± 6.10	12.67 ± 6.81	1	1.77 ± 0.24

Table 1: Lunar Lander Performance

success across all tasks with a single hyperparameter setting, eliminating per–task noise tuning and saving considerable time.

As detailed in Section 3.2, the CSA† variant further preserves user intent at large α by conditioning on the predicted next state. In Fig. 7b, 7c and Fig. 8b, 8c, its success rate remains flat beyond the optimal noise ratio, obviating the need for meticulous noise-ratio selection during inference.

Real Robot Result: We measure the effectiveness of our assistive policy through human user experiments in the Real Peg Insertion environment and Fluid Carrying.

For Real Peg Insertion, we recruited 10 participants (6 identified as male, 3 as female, 1 prefer not to say, with an average age of 25.9).[2] We presented participants with two unidentified copilots, one being our assistive policy CSA† and the other being direct teleoperation (i.e., no assistance). [3].

[2]None of the participants were co-authors or otherwise involved in this research.

[3]DDPM was not included as a baseline in the real-robot experiments because its performance proved extremely sensitive to hyperparameters; despite extensive tuning, we were unable to find any setting that yielded stable convergence on this task.

Pilot	Copilot	Peg Insertion			Charger Plug Insertion		
		Success Rate(%) ↑	NFE	Inference Time(ms) ↓	Success Rate(%) ↑	NFE	Inference Time(ms) ↓
Laggy	None	39.33 ± 10.16	–	–	22.00 ± 7.57	–	–
	DDPM	55.00 ± 8.35	9	5.25 ± 0.42	0.00 ± 0.00	–	–
	CSA	**56.33 ± 12.22**	1	1.00 ± 0.19	**29.00 ± 10.89**	1	1.35 ± 0.37
	CSA†	45.67 ± 8.61	1	1.02 ± 0.16	22.33 ± 6.30	1	1.01 ± 0.06
Noisy	None	24.00 ± 4.66	–	–	17.00 ± 4.57	–	–
	DDPM	28.33 ± 7.07	4	2.72 ± 0.59	0.00 ± 0.00	–	–
	CSA	**71.67 ± 8.92**	1	1.04 ± 0.15	36.67 ± 8.31	1	1.34 ± 0.41
	CSA†	**69.33 ± 7.83**	1	1.18 ± 0.31	**56.67 ± 5.67**	1	1.04 ± 0.12
Noised	None	20.00 ± 7.03	–	–	31.67 ± 9.06	–	–
	DDPM	40.00 ± 8.16	9	5.12 ± 0.15	0.00 ± 0.00	–	–
	CSA	**69.00 ± 8.90**	1	1.18 ± 0.15	**70.00 ± 8.89**	1	1.59 ± 0.33
	CSA†	58.00 ± 8.34	1	1.29 ± 0.20	60.67 ± 7.50	1	1.71 ± 0.40
Slow	None	29.67 ± 8.38	–	–	29.33 ± 5.40	–	–
	DDPM	29.33 ± 4.10	–	–	0.00 ± 0.00	–	–
	CSA	**59.67 ± 12.32**	1	0.99 ± 0.18	36.33 ± 10.12	1	1.25 ± 0.36
	CSA†	35.00 ± 7.58	1	1.09 ± 0.22	**50.33 ± 8.23**	1	1.25 ± 0.37

Table 2: Peg Insertion and Charger Plug Insertion

Copilot	Square Peg		Cylinder Peg		Peg Overall		Fluid Carrying	
	SR (%) ↑	Time (s) ↓	SR (%) ↑	Time (s) ↓	SR (%) ↑	Time (s) ↓	SR (%) ↑	Time (s) ↓
None	60.0	25.9	73.3	30.4	66.7	28.4	53.3	59.9
CSA† (ours)	**73.3**	**22.6**	**93.3**	**24.3**	**83.3**	**24.1**	**83.3**	**39.6**

Table 3: Real Peg Insertion and Fluid Carrying performance w/ and w/o our CSA† copilot.

At the beginning of the experiment, we allowed the participant to practice with both copilots for two trials. In the subsequent testing phase, every participant controlled the robot arm to insert the square peg and the cylinder peg three times each with one randomly chosen copilot, and then repeated this process with the other copilot. Participants provided their user actions a^u through a handheld Quest3 VR controller and viewed the scene only through a side-mounted camera feed, thereby removing normal binocular depth cues. Any trial that exceeded the 60-second limit or triggered an emergency stop due to excessive contact force was marked as a failure.

We consider both the quantitative and qualitative performance of our assistive policy CSA†. Quantitatively, Table 3 compares the average success rate(SR) and the average completion time with and without our assistive policy, showing that providing participants with our assistive policy improved their success rate and reduced the completion time. For Fluid Carrying, similarly, every participant are required to move the cup to these three targets on their own order. Any spillage will be consider as failure, more details in Appendix 7.5. For qualitative result, we provided it in Appendix 7.6.

5 Conclusion

We present CSA, a model-free shared autonomy framework that leverages an ODE-based, distilling diffusion model with partial diffusion to realize one-step denoising collaborative control. Without any explicit goal prediction, CSA relies solely on goal-excluded expert demonstrations to "flash back" each user action to its nearest expert distribution—ensuring smooth trajectory corrections while faithfully preserving the user's intent. In extensive experiments, CSA outperforms prior methods on classic 2D hard-control benchmarks (e.g., Lunar Lander) and delivers powerful assistance in high-dimensional, high-precision tasks such as charger plug insertion—settings where existing approaches falter due to limited data, hyperparameter sensitivity, or slow inference. Moreover, our study of the CSA† variant uncovers how implicit short-term intention modeling underlies robust fidelity maintenance. These results underscore the promise of consistency-model–based shared autonomy for real-time, goal-agnostic assistance in complex control domains.

6 Limitations

While CSA demonstrates strong performance and real-time responsiveness, several areas invite deeper exploration. First, our next-state prediction variant CSA† can degrade when the surrogate noise level (ϵ) becomes large(≥ 0.5) under such conditions the surrogate's outputs may stray from rational human behavior, yielding predictions that add little value. Addressing this will likely require either more robust surrogate models that maintain fidelity across noise regimes or adaptive noise calibration strategies.

Equally important is understanding how the choice of demonstration data shapes assistive performance. Training on "clean," low-variance expert trajectories may yield highly precise corrections but risks rejecting safe but unconventional user motions; conversely, embracing a broader, higher-variance dataset could improve flexibility yet makes it harder to distinguish core expert maneuvers from benign exploratory actions. Characterizing this trade-off—and developing methods to automatically identify and prioritize "causal" expert patterns—stands as a fertile avenue for future work.

Finally, these two threads converge on the challenge of dynamically tuning the assistance strength (α) at inference time. An ideal system would gauge the professionalism of each user action—perhaps via confidence metrics or learned intent classifiers—and adjust α accordingly, providing stronger corrections when needed and stepping back when the user demonstrates competence. We view the development of such context-aware, real-time modulation mechanisms as a highly promising direction to enhance both performance and user trust in consistency-model–based shared autonomy.

Acknowledgments

We would like to thank Bingjie Tang (University of Southern California) and Yiqing Xu (National University of Singapore) for their valuable suggestions on the design of real robot experiments. We are also grateful to Xiaodan Du and Haochen Wang (Toyota Technological Institute at Chicago), as well as Xiao Zhang (University of Chicago), for their insightful feedback on diffusion models. Finally, we would like to thank the entire Robot Intelligence through Perception Lab (RIPL) at TTIC for their support with real robot experiments.

References

[1] T. Yoneda, L. Sun, G. Yang, B. Stadie, and M. Walter. To the noise and back: Diffusion for shared autonomy. *arXiv preprint arXiv:2302.12244*, 2023.

[2] C. Costen, M. Rigter, B. Lacerda, and N. Hawes. Mixed observability MDPs for shared autonomy with uncertain human behaviour. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[3] P. Aigner and B. McCarragher. Human integration into robot control utilising potential fields. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 291–296, 1997. doi:10.1109/ROBOT.1997.620053.

[4] R. C. Goertz. Manipulators used for handling radioactive materials. *Human Factors in Technology*, pages 425–443, 1963.

[5] D. A. Abbink, T. Carlson, M. Mulder, J. C. F. de Winter, F. Aminravan, T. L. Gibo, and E. R. Boer. A topology of shared control systems—finding common ground in diversity. *IEEE Transactions on Human-Machine Systems*, 48(5):509–525, 2018. doi:10.1109/THMS.2018.2791570.

[6] B. McMahan, Z. M. Peng, B. Zhou, and J. Kao. Shared autonomy with IDA: Interventional diffusion assistance. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 128330–128354, 2024.

[7] S. Reddy, A. D. Dragan, and S. Levine. Shared autonomy via deep reinforcement learning. *arXiv preprint arXiv:1802.01744*, 2018.

[8] C. Schaff and M. R. Walter. Residual policy learning for shared autonomy. *arXiv preprint arXiv:2004.05097*, 2020.

[9] M. Ghorbel, J. Pineau, R. Gourdeau, S. Javdani, and S. Srinivasa. A decision-theoretic approach for the collaborative control of a smart wheelchair. *International Journal of Social Robotics*, 10:131–145, 2018.

[10] S. Schröer, I. Killmann, B. Frank, M. Völker, L. Fiederer, T. Ball, and W. Burgard. An autonomous robotic assistant for drinking. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6482–6487, 2015.

[11] A. Phung, G. Billings, A. F. Daniele, M. R. Walter, and R. Camilli. Enhancing scientific exploration of the deep sea through shared autonomy in remote manipulation. *Science Robotics*, 8(81), 2023.

[12] K. Muelling, A. Venkatraman, J.-S. Valois, J. E. Downey, J. Weiss, S. Javdani, M. Hebert, A. B. Schwartz, J. L. Collinger, and J. A. Bagnell. Autonomy infused teleoperation with application to brain computer interface controlled manipulation. *Autonomous Robots*, 41:1401–1422, 2017.

[13] S. Javdani, S. S. Srinivasa, and J. A. Bagnell. Shared autonomy via hindsight optimization. In *Proceedings of Robotics: Science and Systems (RSS)*, pages 10–15607, 2015.

[14] C. Pérez-D'Arpino and J. A. Shah. Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 6175–6182, 2015.

[15] A. D. Dragan and S. S. Srinivasa. A policy-blending formalism for shared control. *International Journal of Robotics Research*, 32(7):790–805, 2013.

[16] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6840–6851, 2020.

[17] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

[18] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. 2022.

[19] Z. Kong, W. Ping, J. Huang, K. Zhao, and B. Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2021.

[20] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[21] L. Ruan, Y. Ma, H. Yang, H. He, B. Liu, J. Fu, N. J. Yuan, Q. Jin, and B. Guo. MM-Diffusion: Learning multi-modal diffusion models for joint audio and video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10219–10228, 2023.

[22] S. Sheynin, A. Polyak, U. Singer, Y. Kirstain, A. Zohar, O. Ashual, D. Parikh, and Y. Taigman. Emu Edit: Precise image editing via recognition and generation tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8871–8879, 2024.

[23] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[24] Y. Xu, J. Mao, Y. Du, T. Lozáno-Pérez, L. P. Kaebling, and D. Hsu. " set it up!": Functional object arrangement with compositional generative models. *arXiv preprint arXiv:2405.11928*, 2024.

[25] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *International Journal of Robotics Research*, 2024.

[26] J. Carvalho, A. T. Le, M. Baierl, D. Koert, and J. Peters. Motion planning diffusion: Learning and planning of robot motions with diffusion models. *arXiv preprint arXiv:2308.01557*, 2024.

[27] Z.-H. Yin, C. Wang, L. Pineda, F. Hogan, K. Bodduluri, A. Sharma, P. Lancaster, I. Prasad, M. Kalakrishnan, J. Malik, et al. DexterityGen: Foundation controller for unprecedented dexterity. *arXiv preprint arXiv:2502.04307*, 2025.

[28] Y. Song, P. Dhariwal, M. Chen, and I. Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.

[29] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[30] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2022.

[31] T. Debus, J. Stoll, R. D. Howe, and P. Dupont. Cooperative human and machine perception in teleoperated assembly. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, pages 51–60, 2001.

[32] J. Kofman, X. Wu, T. J. Luu, and S. Verma. Teleoperation of a robot manipulator using a vision-based human-robot interface. *IEEE Transactions on Industrial Electronics*, 52(5):1206–1219, 2005.

[33] B. D. Argall. Modular and adaptive wheelchair automation. In *Proceedings of the International Symposium on Experimental Robotics (ISER)*, pages 835–848, 2015.

[34] W. Tan, D. Koleczek, S. Pradhan, N. Perello, V. Chettiar, V. Rohra, A. Rajaram, S. Srinivasan, H. S. Hossain, and Y. Chandak. On optimizing interventions in shared autonomy. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 5341–5349, 2022.

[35] M. Janner, Y. Du, J. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2022.

[36] C. Meng, R. Rombach, R. Gao, D. Kingma, S. Ermon, J. Ho, and T. Salimans. On distillation of guided diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14297–14306, 2023.

[37] T. Salimans and J. Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022.

[38] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[39] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. Pmlr, 2018.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[41] B. Tang, M. A. Lin, I. Akinola, A. Handa, G. S. Sukhatme, F. Ramos, D. Fox, and Y. Narang. Industreal: Transferring contact-rich assembly tasks from simulation to reality. *arXiv preprint arXiv:2305.17110*, 2023.

7 Appendix

7.1 Preliminary: Probability Flow ODEs

Diffusion models can be interpreted as discretized numerical solvers for the reverse-time differential equations that govern a forward diffusion process. A diffusion model is characterized by two processes. A *forward diffusion process* is a first-order Markov chain that iteratively adds noise to a (clean) sample drawn from a data distribution $\boldsymbol{y} \sim p_{\text{data}}(\boldsymbol{y})$ A *reverse diffusion process* serves to generate samples from the data distribution by reversing this process to denoise a noisy input $\boldsymbol{x}$. In the general approach, a neural network $D(\cdot)$ is trained to approximate the instantaneous score function $\nabla_x \log p_t(x)$ as a parameterization of the reverse process. By instantiating the generative process using the probability-flow (PF) ODE, the general form that describes the evolution of a sample (forward or backward in time) can be defined as [28, 18]:

$$\mathrm{d}\boldsymbol{x} = -\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}} \log p_t(\boldsymbol{x}; \sigma(t))\mathrm{d}t, \tag{5}$$

where $\sigma(t)$ is the desired noise level at time t and in practice setting as t of the EDM model [18] this paper applies. In the forward process, we represent $p_t(\boldsymbol{x}; \sigma) = p_{\text{data}} \cdot \mathcal{N}\left(\mathbf{0}, \sigma(t)^2\mathbf{I}\right)$. The denoiser function $D(\boldsymbol{x}; \sigma)$ is designed to minimize the L^2 error for samples $\boldsymbol{y}$ drawn from $p_{\text{data}}(\boldsymbol{y})$

$$\mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})} \|D(\boldsymbol{y} + \boldsymbol{n}; \sigma) - \boldsymbol{y}\|_2^2, \tag{6}$$

where $\boldsymbol{n}$ is the noise. The relationship between $D(\boldsymbol{x}; \sigma)$ and the score function is given by

$$\nabla_{\boldsymbol{x}} \log p(\boldsymbol{x}; \sigma) = (D(\boldsymbol{x}; \sigma) - \boldsymbol{x})/\sigma^2. \tag{7}$$

As stated in Section 2, the ODE training objective consider the nearest target distribution in the multi-modal scenario. Expanding Eqn. 6 (see Appendix 7.2), the loss becomes

$$\mathcal{L}(D; \boldsymbol{x}, \sigma) = \int_{\mathbb{R}^d} \frac{1}{Y} \sum_{i=1}^{Y} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}_i, \sigma^2\mathbf{I}\right) \|D(\boldsymbol{x}; \sigma) - \boldsymbol{y}_i\|_2^2 \, \mathrm{d}\boldsymbol{x} \tag{8}$$

Minimizing $\mathcal{L}(D; \boldsymbol{x}, \sigma)$ is equivalent to solving a convex optimization problem, and the closed-form solution follows as:

$$D(\boldsymbol{x}; \sigma) = \frac{\sum_i \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}_i, \sigma^2\mathbf{I}\right) \boldsymbol{y}_i}{\sum_i \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}_i, \sigma^2\mathbf{I}\right)}, \tag{9}$$

The above involves a softmax operation over all target distributions $\boldsymbol{y}_i$ that is feasible to noise sample $\boldsymbol{x}$, therefor guarantee to choose the nearest neighbor. Fig. 9 shows that SDE based DDPM model has no guarantee on finding nearest expert, while ODE based model could do this in one step after distilling.

7.2 Full proof of nearest expert

Let us assume that our training set consists of a finite number of samples $\{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_Y\}$. This implies $p_{\text{data}}(\boldsymbol{x})$ is represented by a mixture of Dirac delta distributions:

$$p_{\text{data}}(\boldsymbol{x}) = \frac{1}{Y} \sum_{i=1}^{Y} \delta(\boldsymbol{x} - \boldsymbol{y}_i)$$

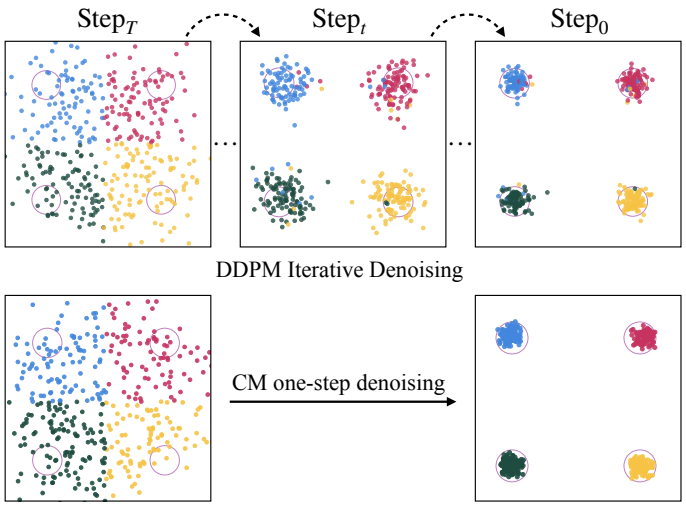


Figure 9: Consistency model and DDPM on a 2D example

which allows us to also express $p(x;\sigma)$ in closed form,

$$\begin{aligned}
p(\boldsymbol{x};\sigma) &= p_{\text{data}} * \mathcal{N}\left(\mathbf{0}, \sigma(t)^2\mathbf{I}\right) \\
&= \int_{\mathbb{R}^d} p_{\text{data}}\left(\boldsymbol{x}_0\right) \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_0, \sigma^2\mathbf{I}\right) \mathrm{d}\boldsymbol{x}_0 \\
&= \int_{\mathbb{R}^d} \left[\frac{1}{Y}\sum_{i=1}^{Y} \delta\left(\boldsymbol{x}_0 - \boldsymbol{y}_i\right)\right] \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_0, \sigma^2\mathbf{I}\right) \mathrm{d}\boldsymbol{x}_0 \\
&= \frac{1}{Y}\sum_{i=1}^{Y} \int_{\mathbb{R}^d} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{x}_0, \sigma^2\mathbf{I}\right) \delta\left(\boldsymbol{x}_0 - \boldsymbol{y}_i\right) \mathrm{d}\boldsymbol{x}_0 \\
&= \frac{1}{Y}\sum_{i=1}^{Y} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}_i, \sigma^2\mathbf{I}\right).
\end{aligned}$$

Let us now consider the denoising score matching loss of Eq. 6. By expanding the expectations, we can rewrite the formula as an integral over the noisy samples $\boldsymbol{x}$:

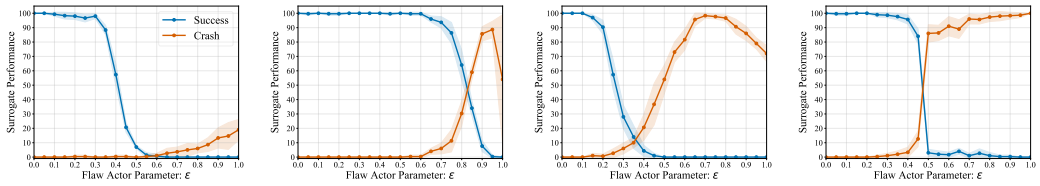
$$\begin{aligned}
\mathcal{L}(D;\sigma) &= \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{n}\sim\mathcal{N}(\mathbf{0},\sigma^2\mathbf{I})} \|D(\boldsymbol{y}+\boldsymbol{n};\sigma) - \boldsymbol{y}\|_2^2 \\
&= \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \mathbb{E}_{\boldsymbol{x}\sim\mathcal{N}(\boldsymbol{y},\sigma^2\mathbf{I})} \|D(\boldsymbol{x};\sigma) - \boldsymbol{y}\|_2^2 \\
&= \mathbb{E}_{\boldsymbol{y}\sim p_{\text{data}}} \int_{\mathbb{R}^d} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}, \sigma^2\mathbf{I}\right) \|D(\boldsymbol{x};\sigma) - \boldsymbol{y}\|_2^2 \, \mathrm{d}\boldsymbol{x} \\
&= \frac{1}{Y}\sum_{i=1}^{Y} \int_{\mathbb{R}^d} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}_i, \sigma^2\mathbf{I}\right) \|D(\boldsymbol{x};\sigma) - \boldsymbol{y}_i\|_2^2 \, \mathrm{d}\boldsymbol{x} \\
&= \int_{\mathbb{R}^d} \underbrace{\frac{1}{Y}\sum_{i=1}^{Y} \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{y}_i, \sigma^2\mathbf{I}\right) \|D(\boldsymbol{x};\sigma) - \boldsymbol{y}_i\|_2^2}_{=:\mathcal{L}(D;\boldsymbol{x},\sigma)} \mathrm{d}\boldsymbol{x}.
\end{aligned}$$

The proof is adopted from EDM appendix [18].

7.3 Surrogate hyperparameter choose and performance

In Fig. 10, we can see for noised surrogate, when $\epsilon = 0.0$, the surrogate is actually the expert, and when $\epsilon = 0.45$ (Fig. 10a) the success rate drop down significantly to 20%, so we choose $\epsilon^{\text{noised}} =$

0.45 for lunar lander environment. Similarly, we can so the same choosing for peg insertion and charger insertion, table 4 shows the hyperparameter ϵ we use.



(a) Noised Surrogate (b) Laggy Surrogate (c) Noisy Surrogate (d) Slow Surrogate

Figure 10: Surrogate performance over different flaw parameter ϵ for lunar lander

Table 4: Surrogate Flaw Parameters

Environment	ϵ^{laggy}	ϵ^{noisy}	ϵ^{noised}	ϵ^{slow}
Lunar Lander	0.85	0.30	0.45	0.47
Peg Insertion	0.80	0.15	0.20	0.65
Charge Plug Insertion	0.60	0.15	0.10	0.40

7.4 EDM Details

An EDM model is fully specified once the *noise schedule* $\{\sigma_t\}_{t=0}^{T-1}$ is fixed. The schedule assigns a noise standard deviation σ_t to each discrete diffusion step t. Throughout this paper we adopt the default hyperparameters of Karras et al. [18]:

$$\sigma_{\min} = 0.002, \qquad \sigma_{\max} = 80.0, \qquad \sigma_{\text{data}} = 0.5, \qquad \rho = 7.$$

Karras ρ-schedule. For T noise levels and index $t \in \{0, \dots, T-1\}$, the schedule is

$$\sigma^t \;=\; \left(\sigma_{\max}^{1/\rho} + \frac{t}{T-1}\left(\sigma_{\min}^{1/\rho} - \sigma_{\max}^{1/\rho}\right)\right)^{\rho}. \tag{A.1}$$

Network pre-conditioning coefficients. At any noise level σ^t, EDM rescales the network input, skip connection, and output using

$$\begin{aligned} c_{\text{in}}^t &= \frac{1}{\sqrt{(\sigma^t)^2 + \sigma_{\text{data}}^2}}, \\ c_{\text{skip}}^t &= \frac{\sigma_{\text{data}}^2}{(\sigma^t)^2 + \sigma_{\text{data}}^2}, \\ c_{\text{out}}^t &= \frac{\sigma^t\,\sigma_{\text{data}}}{\sqrt{(\sigma^t)^2 + \sigma_{\text{data}}^2}}. \end{aligned} \tag{A.2}$$

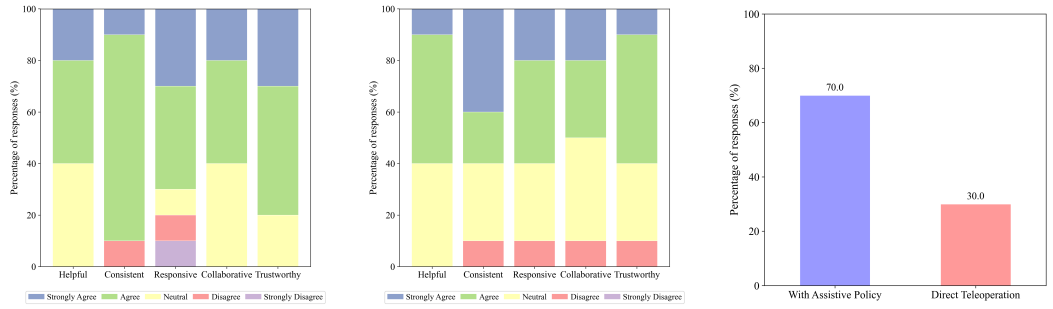
These coefficients stabilise both training and sampling across the wide dynamic range between $\sigma_{\max}$ and $\sigma_{\min}$.

Sampling a diffusion time step. Whenever a single training batch requires a noise level, we draw the index t *uniformly* from the schedule:

$$t \;\sim\; \mathcal{U}\{0, \dots, T-1\}, \qquad \sigma \;\leftarrow\; \sigma_t.$$

7.5 Experimental Details

(a) Lunar Lander: Lunar Lander (Fig. 6a) is a 2D continuous control environment adapted from OpenAI Gym, where the goal is to land a spaceship on a designated landing pad. The action space



(a) Responses with Assistive Policy (b) Responses with Teleoperation (c) Participant Overall Preference

Figure 11: Human participant qualitative survey result in the Real Peg Insertion task

controls the thrust applied to the left and right engines; to turn left, for example, the agent must apply slightly more thrust to the right engine than to the left. This indirect control strategy is counterintuitive compared to natural human intuition (e.g., pushing left to move left), making the task particularly challenging. The state space includes the spaceship's position, orientation, linear and angular velocities, ground contact indicators for each leg, and the landing pad location (the latter provided only to the pilot). An episode terminates when the spaceship successfully lands and becomes idle, crashes, flies out of bounds, or reaches a timeout of 1000 steps.

(b) Peg Insertion: Peg Insertion (Fig. 6b) is a continuous control environment from ManiSkill, involving a robotic arm tasked with inserting the orange end of an orange-white peg into a hole in a box. The peg has a fixed half-length of 10 cm and radius of 2 cm, while the box hole radius provides a 1 cm clearance. At episode start, both peg and box positions and orientations are randomized on a flat table surface. Actions involve continuous end-effector movements. The state includes robot joint positions and velocities, end-effector pose, peg pose and dimensions, and the hole's pose and radius. Success occurs when the peg's white end is within 1.5 cm of the hole center. Episodes end upon success, boundary violation, or reaching a 200-step limit.

(c) Charger Plug Insertion: Charger plug (Fig. 6c) insertion is an advanced version of peg insertion where a robotic arm must pick up a charger and insert it into a receptacle. The charger has a base of fixed size and a dual-peg design, with a clearance of *0.5 mm* for insertion. At episode start, the charger and receptacle are randomized in XY position and orientation on the table. Actions control the robot's end-effector. The state includes robot joint positions and velocities, end-effector pose, charger pose, receptacle pose, and goal pose. Success is achieved when the charger is fully inserted within a *5mm* positional tolerance and 0.2 radian angular tolerance. Episodes end upon success, boundary violation, or reaching a 300-step limit.

(d) Real Peg Insertion: Real Peg Insertion (Fig. 6d) is performed with a UR5 robot arm equipped with a Robotiq 3-Finger gripper. The task involves inserting either a red square peg or a blue cylinder peg into its corresponding hole, with clearance tolerances of 5 mm and 4.5 mm, respectively. The state s_t includes the current pose of the end-effector and the wrench readings from a wrist-mounted FT-300 force/torque sensor. The action a_t is the target end-effector position.

(e) Fluid Carrying: Fluid Carrying (Fig. 6e) is conducted under the same hardware configuration as the *Real Peg Insertion* task. In this task, users are required to transport a cup filled with beans from a fixed starting location to one of three predefined target locations, each measuring 9.5 cm in diameter. Unstructured Obstacles are positioned between the start and target positions, increasing the difficulty of the task. Any spillage of beans is considered a failure. The state s_t is the current pose of the end-effector and the action a_t includes the target position and orientation of the end-effector.

Environment	State Dimension	Action Dimension
Lunar Lander (Fig. 6a)	8	2
Peg Insertion (Fig. 6b)	35	8
Charger Plug Insertion (Fig. 6c)	32	8
Real Peg Insertion (Fig. 6d)	9	3
Fluid Carrying (Fig. 6e)	6	6

Table 5: State and Action Dimensions for Each Environment

7.6 Human Participant Qualitative Survey Result

In a qualitative survey, we asked participants to rate how "helpful," "consistent," "responsive," "collaborative," and "trustworthy" each copilot felt, using a five-point Likert scale. They were also asked which copilot they preferred overall. To avoid bias, we did not reveal which system was our assistive policy; the survey simply labeled them "Policy A" and "Policy B."

As shown in Fig. 11, participants rated our assistive copilot higher for being helpful, collaborative, and trustworthy, and the majority preferred it overall. We also received comments such as "The manipulation and motion are smoother under Policy B (our assistive policy) from the user's perspective," "It felt like I got some assistance during insertion with B," and "Policy A (direct teleoperation) is easy to overshoot, especially when moving down."