APCtrl: Adding Conditional Control to Diffusion Models by Alternative Projection

Anonymous authors
Paper under double-blind review

Abstract

Enhancing the versatility of pretrained diffusion models through advanced conditioning techniques is crucial for improving their applicability. We present APCtrl, a novel conditional image generation approach that formulates the latent $\mathbf{z}_t$ at timestep t as the projection $\mathbf{z}_t = \text{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$ onto the denosing set $\mathfrak{D}_t$. For conditional control, APCtrl integrates the condition set $\mathfrak{C}_t$, defined by a latent control network $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\cdot,\cdot)$. Our method simplifies conditional sampling to recursive projections $\mathbf{z}_t = \text{Proj}_{\mathfrak{I}_t} \circ \text{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$, where each projection step integrates both the diffusion and condition priors. By employing Alternative Projection, our approach offers several key advantages: 1. Multi-Condition Generation: easily expandable with additional conditional sets; 2. Model and Sampling Agnosticism: works with any model or sampling method; 3. Unified Control Loss: simplifies the management of diverse control applications; 4. Efficiency: delivers comparable control with reduced training and sampling times. Extensive experiments demonstrate the superior performance of our method.

1 Introduction

Unconditional diffusion models, first introduced by Ho et al. (2020), laid the foundation for generative image modeling. Characterized by the latent sequence $\mathbf{z}_0, \mathbf{z}_1, \ldots, \mathbf{z}_T$, they have significantly advanced the generation of high-fidelity images (Yang et al., 2023a). In this sequence, $\mathbf{z}_t$ represents progressively noisier data samples for $t \in (0, T]$ and $\mathbf{z}_0$ corresponds to the true data samples. The forward process introduces noise gradually, transitioning from $\mathbf{z}_{t-1}$ to $\mathbf{z}_t$ according to the distribution $q(\mathbf{z}_t|\mathbf{z}_{t-1}) \coloneqq \mathcal{N}(\mathbf{z}_t \mid \sqrt{\alpha_t}\mathbf{z}_{t-1}, (1-\alpha_t)\boldsymbol{I})$, where α_t is a constant hyperparameter. The objective of diffusion models is to generate a sample $\mathbf{z}_0$ from the data distribution $p(\mathbf{z}_0)$, which can be formulated as an optimization problem: $\text{argmax}_{\mathbf{z}_0} \log p(\mathbf{z}_0)$, seeking the optimal $\mathbf{z}_0$ that maximizes $p(\mathbf{z}_0)$.

The distribution $p(\mathbf{z}_0)$ is not directly accessible. In the reverse process, the diffusion models offer an approximation through the marginal distribution $p_{\boldsymbol{\theta}}(\mathbf{z}_0)$. The model parameters $\boldsymbol{\theta}$ are optimized using the Evidence Lower Bound (ELBO), which serves as a lower bound for $\log p_{\boldsymbol{\theta}}(\mathbf{z}_0)$. Specifically, we have: $\log p_{\boldsymbol{\theta}}(\mathbf{z}_0) \geq \mathbb{E}_{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\left[\log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}_{1:T})}{q(\mathbf{z}_{1:T}|\mathbf{z}_0)}\right]$. The right ELBO term can be further expanded as follows: $\mathbb{E}_{q(\mathbf{z}_1|\mathbf{z}_0)}[\log p_{\boldsymbol{\theta}}(\mathbf{z}_0|\mathbf{z}_1)] - D_{\text{KL}}(q(\mathbf{z}_T|\mathbf{z}_0) \parallel p_{\boldsymbol{\theta}}(\mathbf{z}_T)) - \sum_{t>1} \mathbb{E}_{q(\mathbf{z}_t|\mathbf{z}_0)}[D_{\text{KL}}(q(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0) \parallel p_{\boldsymbol{\theta}}(\mathbf{z}_{t-1}|\mathbf{z}_t))]$. Thus, the goal of the diffusion model becomes to maximize the reverse transition distribution $p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{t+1})$, which in turn maximizes $\log p_{\boldsymbol{\theta}}(\mathbf{z}_0)$. Consequently, sampling from the reverse transition distribution $p_{\boldsymbol{\theta}}(\mathbf{z}_t|\mathbf{z}_{t+1})$ can then be expressed as Equation (1), where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ is the neural network designed to predict the score function $\nabla_{\mathbf{z}_t} \log p(\mathbf{z}_t)$ (Song et al., 2020b).

$$\mathbf{z}_t = \frac{1}{\sqrt{\alpha_{t+1}}}\mathbf{z}_{t+1} + \frac{(1-\alpha_{t+1})}{\sqrt{\alpha_{t+1}}}\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \sqrt{\frac{(1-\alpha_{t+1})(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t+1}}}\boldsymbol{\epsilon} \tag{1}$$

Unconditional diffusion models was further extended by text-conditional models (Rombach et al., 2022; Yang et al., 2024b). However, these models faced the inherent challenge of

Table 1: A Feature-Rich Approach for Conditional Image Generation. APCtrl boasts an array of beneficial features, surpassing the capabilities of previous methods.

Methods	Latent Control	Controlled by Multi-Condition	Backbones Agnosticism	Sampling Agnosticism	Unified Control Loss	Control on Sampling
Control-on-Training						
ControlNet (Zhang et al., 2023)	✔	✔	✘	✔	✘	✘
ControlNet++ (Li et al., 2024b)	✔	✔	✘	✔	✘	✘
T2I-Adapter (Mou et al., 2024)	✔	✔	✘	✔	✘	✘
UniCtrlNet (Zhao et al., 2024)	✔	✔	✘	✔	✘	✘
UniControl (Qin et al., 2023)	✔	✔	✘	✔	✘	✘
GLIGEN (Li et al., 2023)	✔	✘	✘	✔	✘	✘
Control-on-Sampling						
UniGuid (Bansal et al., 2024)	✘	✘	✔	✘	✘	✔
DSG (Yang et al., 2024c)	✘	✘	✔	✘	✘	✔
FreeDoM (Yu et al., 2023)	✘	✘	✔	✘	✘	✔
APCtrl (Ours)	✔	✔	✔	✔	✔	✔

accurately capturing all image details from text descriptions alone. To address this, diffusion models have incorporated additional conditioning signals, such as bounding boxes (Li et al., 2023; Yang et al., 2023b; Zhao et al., 2024), reference images (Li et al., 2024a; Ruiz et al., 2023), and segmentation maps (Zhang et al., 2023; Bansal et al., 2024; Zhao et al., 2024; Qin et al., 2023), offering more granular control over the generated images.

Conditional image generation falls into two camps: methods that integrate control networks, and those that adjust the inference process for direct control. **Control-on-Training** approaches like ControlNets (Zhang et al., 2023) train networks to refine latent spaces and match images to attributes, incurring retraining costs due to feature space inconsistencies. On the other hand, **Control-on-Sampling** techniques, such as Universal Guidance (Bansal et al., 2024), use pre-trained models to guide sampling, offering flexibility without retraining. However, this comes with potential downsides, such as suboptimal gradient estimations that may degrade sampling quality and prolong sampling times.

APCtrl solves these challenges. Let $\mathfrak{D}_0$ denote the set of natural images, and $\mathfrak{D}_t$ represent the noisy versions, generated by adding noise to $\mathfrak{D}_0$, such that $\mathfrak{D}_t := \{\mathbf{z}_t \mid \mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \mathbf{z}_0 \in \mathfrak{D}_0\}$ with $\bar{\alpha}_t = \prod_{i=1}^{t}\alpha_i$. The denoising projection $\mathbf{z}_t = \text{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$, as defined by Equation (1), maps a noisy point $\mathbf{z}_{t+1} \in \mathfrak{D}_{t+1}$ to a less noisy point $\mathbf{z}_t \in \mathfrak{D}_t$. The diffusion generation process is thus a sequence of such projections. To enhance control, we introduce a condition set $\mathfrak{C}_t$, which defines points that satisfy specific constraints at each step t. The intersection $\mathfrak{I}_t := \mathfrak{D}_t \cap \mathfrak{C}_t$ identifies points that conform to both $\mathfrak{D}_t$ and $\mathfrak{C}_t$. By defining the intersection projection $\text{Proj}_{\mathfrak{I}_t}(\cdot)$, conditional generation is redefined as a recursive sequence of projections $\mathbf{z}_t = \text{Proj}_{\mathfrak{I}_t} \circ \text{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$, as shown in Algorithm 1.

Our method lies in the condition projection $\text{Proj}_{\mathfrak{C}_t}(\cdot)$, implemented through a latent control network that imposes constraints and calculates projections onto the conditional sets. This approach offers several key advantages: enhanced adaptability to diverse backbones, more precise and efficient synthesis via latent control, and a unified MSE latent control loss applicable to numerous conditions. By applying Alternative Projection with the denoising projection $\text{Proj}_{\mathfrak{D}_t}(\cdot)$ and the condition projection $\text{Proj}_{\mathfrak{C}_t}(\cdot)$, we compose the intersection projection $\text{Proj}_{\mathfrak{I}_t}(\cdot)$. This method outperforms other sampling techniques. A key feature is the straightforward implementation of multi-condition control via projections onto intersections of condition sets. Table 1 presents a detailed comparison, highlighting how APCtrl surpasses previous methods with its array of beneficial features.

2 Related Works

Alternative Projection is a technique with a long-standing history. It aims to find a point within the intersection of multiple sets through a sequence of successive projections onto each set, and was seminally studied by Von Neumann (1951), and has since been applied in a myriad of contexts (Deutsch, 1992). Numerous variants, such as relaxed projections (Agmon, 1954; Motzkin & Schoenberg, 1954; Gubin et al., 1967; Brègman,

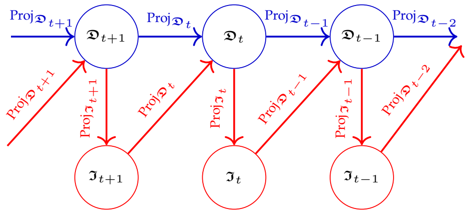


Figure 1: APCtrl Sampling: APCtrl transforms traditional diffusion sampling into a recursive projection, as illustrated in blue in the figure. It integrates easily, needing just one extra line of code, highlighted in red and noted as line 5 in Algorithm 1.

Algorithm 1 APCtrl Sampling

Input: Initial noise $\mathbf{z}_T$, denoising and condition set $\mathfrak{D}_t$ and $\mathfrak{C}_t$.

1 **for** $t = T - 1$ *to* 1 **do**
2   $\mathbf{z}_t = \mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$
3   **if** *conducting APCtrl Sampling* **then**
4     $\mathfrak{I}_t = \mathfrak{D}_t \cap \mathfrak{C}_t$
5     $\mathbf{z}_t = \mathrm{Proj}_{\mathfrak{I}_t}(\mathbf{z}_t)$, which be recursively computed by
      for n=1 to N **do**
        $\mathbf{z}_t = \widehat{\mathrm{Proj}}_{\mathfrak{I}_t}(\mathbf{z}_t)$ in Equation (9)
      end
6   **end**
7 **end**

Output: $\mathbf{z}_t$

1965), inexact projections (Kruger & Thao, 2016), Dykstra's algorithm (Boyle & Dykstra, 1986), Douglas–Rachford splitting (Douglas & Rachford, 1956; Lions & Mercier, 1979), ADMM (Boyd, 2010), and generalized alternating projections (Fält & Giselsson, 2024), have been proposed.

Diffusion Models (Croitoru et al., 2023; Yang et al., 2023a) constitute a class of models that incrementally introduce noise to data in a controlled manner, with the goal of learning to reverse this process for generating samples. The current research landscape is primarily dominated by three formulations: Denoising Diffusion Probabilistic Models (Ho et al., 2020; Nichol & Dhariwal, 2021; Sohl-Dickstein et al., 2015), Score-Based Generative Model (Song & Ermon, 2019; 2020), and Stochastic Differential Equations (Song et al., 2021; 2020b). The interconnections between them are elucidated by Luo (2022) and Chan (2024).

Control-on-Training takes supplementary networks to modify the latent representations of diffusion models according to particular image conditions. Researchers (Bansal et al., 2023; Nichol et al., 2022; Rombach et al., 2022) have expanded $\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ in Equation (1) to include both text and image conditions. A notable example of this approach is ControlNet (Zhang et al., 2023), which has become a significant focus within the field. The broader community has contributed to this area by sharing a variety of ControlNets trained across diverse input conditions. Other prominent examples include ControlNet++ (Li et al., 2024b), T2I-Adapter (Mou et al., 2024), UniControlNet (Zhao et al., 2024), UniControl (Qin et al., 2023), GLIGEN (Li et al., 2023), and Ctrl-Adapter (Lin et al., 2024).

Control-on-Sampling utilizes frozen pre-trained models, with modifications to the sampling method to reconstruct an image from a given guidance. Prior work has approached this task with various constraints (Dhariwal & Nichol, 2021; Kawar et al., 2022; Wang et al., 2022; Chung et al., 2023; Lugmayr et al., 2022; Chung et al., 2022; Graikos et al., 2022). For instance, Dhariwal & Nichol (2021) trained a classifier on images of different noise scales to serve as the guidance and incorporated the classifier's gradients into the sampling process. However, classifiers for noisy images are often domain-specific and not generally available. To address the challenge, several state-of-the-art sampling methods have been introduced, including DSG (Yang et al., 2024c), UniGuidance (Bansal et al., 2024), FreeDoM (Yu et al., 2023), MultiDiffusion (Bar-Tal et al., 2023), and ReSample (Song et al., 2023).

3 APCtrl Sampling

Diffusion generation involves the successive application of the projection $\mathbf{z}_t = \mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$, as outlined in Equation (1). To ensure that the denoised point from $\mathfrak{D}_t$ also satisfies the constraint from $\mathfrak{C}_t$, *i.e.* to maintain $\mathbf{z}_t$ within the intersection $\mathfrak{I}_t = \mathfrak{D}_t \cap \mathfrak{C}_t$, we apply the intersection projection $\mathrm{Proj}_{\mathfrak{I}_t}(\cdot)$ to $\mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$. This results in the iterative formula $\mathbf{z}_t = \mathrm{Proj}_{\mathfrak{I}_t} \circ \mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$, detailed in Algorithm 1 and depicted in Figure 1. Building upon the definition of $\mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$ from Equation 1, **this section is dedicated to explaining the use of Alternative Projection to implement the intersection projection** $\mathrm{Proj}_{\mathfrak{I}_t}(\cdot)$**.**

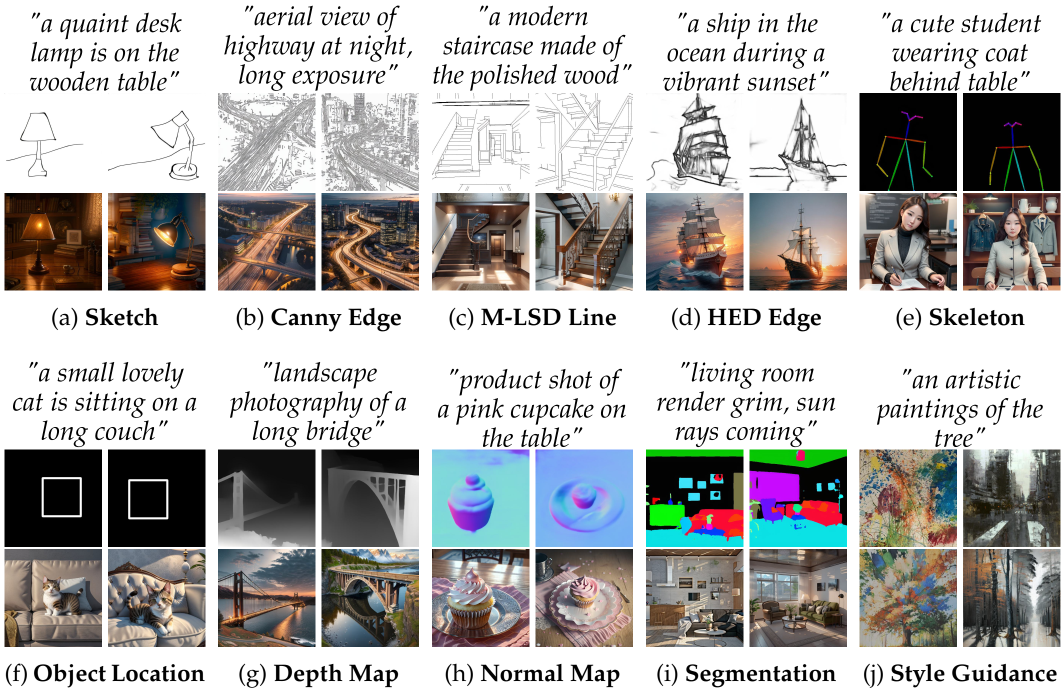


Figure 2: Image Generation with Single Control: APCtrl facilitates the integration of various conditions into diffusion models. Each subfigure is structured with the prompt text in the first row, the conditional image control in the second row, and the resulting controlled generation in the third row.

3.1 Alternative Projection

The alternative projection method is an iterative process used to identify a point that belongs to the intersection of two sets, $\mathfrak{S}_1$ and $\mathfrak{S}_2$. Although projecting onto each set separately is easy, projecting directly onto their intersection $\mathfrak{S}_1 \cap \mathfrak{S}_2$ is challenging. Let the projection operators onto $\mathfrak{S}_1$ and $\mathfrak{S}_2$ be denoted by $\mathrm{Proj}_{\mathfrak{S}_1}$ and $\mathrm{Proj}_{\mathfrak{S}_2}$, respectively. The alternative projection method is straightforward: beginning with any point, the vector $\boldsymbol{x}$ is iteratively updated by applying the composition of projections, such that $\mathbf{z} = \mathrm{Proj}_{\mathfrak{S}_1} \circ \mathrm{Proj}_{\mathfrak{S}_2}(\mathbf{z})$.

3.2 From Latent Control to Latent Control

Pixel Control, used in previous Control-on-Sampling methods, computes the controlled intermediate latent code $\mathbf{z}_t = \mathrm{argmin}_{\mathbf{z}}\, \boldsymbol{\mathcal{L}}(\boldsymbol{I}_c, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{D}}(\mathbf{z})))$ with the initial point $\boldsymbol{\mathcal{Z}}(\mathbf{z}_{t+1}) := \sqrt{\bar{\alpha}_{t+1}}^{-1}(\mathbf{z}_{t+1} + (1-\bar{\alpha}_{t+1})\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1))$. In this formulation, $\boldsymbol{\mathcal{Z}}(\mathbf{z}_{t+1})$ acts as a denoiser at time step $t+1$ for the latent variable at time step 0, $\boldsymbol{I}_c$ represents the control image, such as segmentation, depth map, or HED. $\boldsymbol{\mathcal{D}}(\cdot)$ is the decoder of the diffusion model, and $\boldsymbol{\mathcal{B}}(\cdot)$ is the pre-trained condition network, such as networks for segmentation, depth estimation, or HED edge. The metric $\boldsymbol{\mathcal{L}}(\cdot,\cdot)$ can be any loss function, such as MSE for depth or HED images similarity, or Cross-Entropy for segmentation similarity.

Latent Control offers a paradigm shift pixel-level manipulation to operations within the latent space. This shift to a lower-dimensional and more compact latent space, allows for more precise control over image generation. Additionally, it simplifies the optimization process by eliminating the need for a decoder $\boldsymbol{\mathcal{D}}(\cdot)$, thus enhancing efficiency. The method specifically utilizes an encoder $\boldsymbol{\mathcal{E}}(\cdot)$ in conjunction with a latent control network $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\cdot,\cdot)$, to determine the controlled intermediate latent representation $\mathbf{z}_t$.

$$\mathbf{z}_t = \mathrm{argmin}_{\mathbf{z}} \|\boldsymbol{\mathcal{E}}(\boldsymbol{I}_c) - \boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}, t)\|^2 \quad \text{solving with the initial point } \mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1}). \tag{2}$$

For a well-trained model $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\cdot,\cdot)$, the approximation should hold: $\|\boldsymbol{\mathcal{E}}(\boldsymbol{I}_c) - \boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)\|^2 \approx \boldsymbol{\mathcal{L}}(\boldsymbol{I}_c, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{D}}(\mathbf{z}_{0|t})))$, which allows us to use the latent control $\mathbf{z}_t = \operatorname{argmin}_{\mathbf{z}} \|\boldsymbol{\mathcal{E}}(\boldsymbol{I}_c) - \boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}, t)\|^2$ as a substitute for the pixel-level control $\mathbf{z}_t = \operatorname{argmin}_{\mathbf{z}} \boldsymbol{\mathcal{L}}(\boldsymbol{I}_c, \boldsymbol{\mathcal{B}}(\boldsymbol{\mathcal{D}}(\mathbf{z})))$. This applies to various types of control, across different control types such as segmentation guidance, depth map guidance and HED edge guidance.

Training the Latent Control Network focuses on refining the operator $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\cdot,\cdot)$. This operator is built upon the U-Net architecture of the stable diffusion model, with initialization from the SDv1.5 checkpoint. The training process is conducted as Equation (3). During optimization, two primary objectives are achieved: image denoising and feature translation. Denoising improves the latent representation z_t by reducing noise, thereby enhancing data clarity and adherence to constraints. Meanwhile, feature translation converts denoised image features into control-relevant features, which are essential for specific improvements. The efficacy of diffusion models in image translation has been demonstrated in previous work (Parmar et al., 2024). For further details, please refer to Appendix A.

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{\mathcal{E}}(\boldsymbol{I}_c) - \boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)\|^2 \tag{3}$$

Feasible Sets encompass all points fulfilling specific criteria. APCtrl involves two kinds of feasible sets: the denosing set $\mathfrak{D}_t$ and the condition set $\mathfrak{C}_t$. Let $\mathfrak{D}_0$ denote the set of natural images, the denosing set at time step t can be expressed as

$$\mathfrak{D}_t = \{\mathbf{z}_t \mid \mathbf{z}_t = \sqrt{\bar{\alpha}_t}\mathbf{z}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},\ \mathbf{z}_0 \in \mathfrak{D}_0\} \tag{4}$$

with the inclusion relation $\mathfrak{D}_t \subseteq \mathfrak{D}_{t+1}$. Upon the successful training of $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}, t)$, for any point $\mathbf{z}$ within the feasible set, the loss $\|\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}, t) - \boldsymbol{\mathcal{E}}(\boldsymbol{I}_c)\|^2$ is expected to be minimal. Thus, with δ as a predefined threshold, the condition set at time step t can be formulated as

$$\mathfrak{C}_t = \{\mathbf{z} \mid \|\boldsymbol{\mathcal{E}}(\boldsymbol{I}_c) - \boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}, t)\|^2 < \delta\} \tag{5}$$

3.3 Intersection Projection Implementation

In this section, we reveal that the intersection projection $\mathrm{Proj}_{\mathfrak{I}_t}(\mathbf{z}_t)$ can be effectively computed through the iterative application of the joint up/down projection $\widehat{\mathrm{Proj}}_{\mathfrak{I}_t}(\mathbf{z}_t)$.

Up/Down Projections are integral to our method. We will introduce two down projections and one up projection here. Specifically, the denoising projection $\mathrm{Proj}_{\mathfrak{D}_t}(\cdot)$ is defined as

$$\mathrm{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1}) = \frac{1}{\sqrt{\alpha_{t+1}}}\mathbf{z}_{t+1} + \frac{(1-\alpha_{t+1})}{\sqrt{\alpha_{t+1}}}\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \sqrt{\frac{(1-\alpha_{t+1})(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t+1}}}\boldsymbol{\epsilon} \tag{6}$$

in accordance Equation (1). This operator serves to map elements from the set $\mathfrak{D}_{t+1}$ to the set $\mathfrak{D}_t$. According to Equation (2), we define the condition projection as

$$\mathrm{Proj}_{\mathfrak{C}_t}(\mathbf{z}_{t+1}) = \arg\min \mathbf{z}_t \|\boldsymbol{\mathcal{E}}(\boldsymbol{I}_c) - \boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)\|^2 \quad \text{solving with initial point } \mathbf{z}_{t+1}. \tag{7}$$

This projection is also considered as a mapping from $\mathfrak{D}_{t+1}$ to $\mathfrak{D}_t$. Collectively, these two projections are termed 'Down Projections' due to their index decreasing from $t+1$ to t. Conversely, we introduce an 'Up Projection', which maps a point $\mathbf{z}_t$ from $\mathfrak{D}_t$ into $\mathfrak{D}_{t+1}$:

$$\mathrm{Proj}_{\mathfrak{D}_{t+1}}(\mathbf{z}_t) = \sqrt{\alpha_{t+1}}\mathbf{z}_t + \sqrt{1-\alpha_{t+1}}\boldsymbol{\epsilon} \tag{8}$$

Joint Up/Down Projection $\widehat{\mathrm{Proj}}_{\mathfrak{I}_t}(\mathbf{z}_t)$ is devised to calculate $\mathrm{Proj}_{\mathfrak{I}_t}(\cdot)$. For the computation of $\mathrm{Proj}_{\mathfrak{I}_t}(\mathbf{z}_t)$, we define the projection $\widehat{\mathrm{Proj}}_{\mathfrak{I}_t}(\mathbf{z}_t)$ as:

$$\widehat{\mathrm{Proj}}_{\mathfrak{I}_t}(\mathbf{z}_t) = \mathrm{Proj}_{\mathfrak{D}_t} \circ \mathrm{Proj}_{\mathfrak{D}_{t+1}} \circ \mathrm{Proj}_{\mathfrak{C}_t} \circ \mathrm{Proj}_{\mathfrak{D}_{t+1}}(\mathbf{z}_t) \tag{9}$$

This equation recursively projects onto the intersection of sets by leveraging the subset relationship $\mathfrak{D}_t \subseteq \mathfrak{D}_{t+1}$, facilitating convergence towards $\mathfrak{I}_t = \mathfrak{D}_t \cap \mathfrak{C}_t$. The result is a point that lies within both $\mathfrak{C}_t$ and $\mathfrak{D}_t$. Thus, employing Alternative Projection, we iteratively obtain the value of $\mathrm{Proj}_{\mathfrak{I}_t}(\mathbf{z}_t)$ through the repeated application of $\mathbf{z}_t = \widehat{\mathrm{Proj}}_{\mathfrak{I}_t}(\mathbf{z}_t)$ over N iterations, as illustrated in Algorithm 1. For more details, please refer to Appendix B.

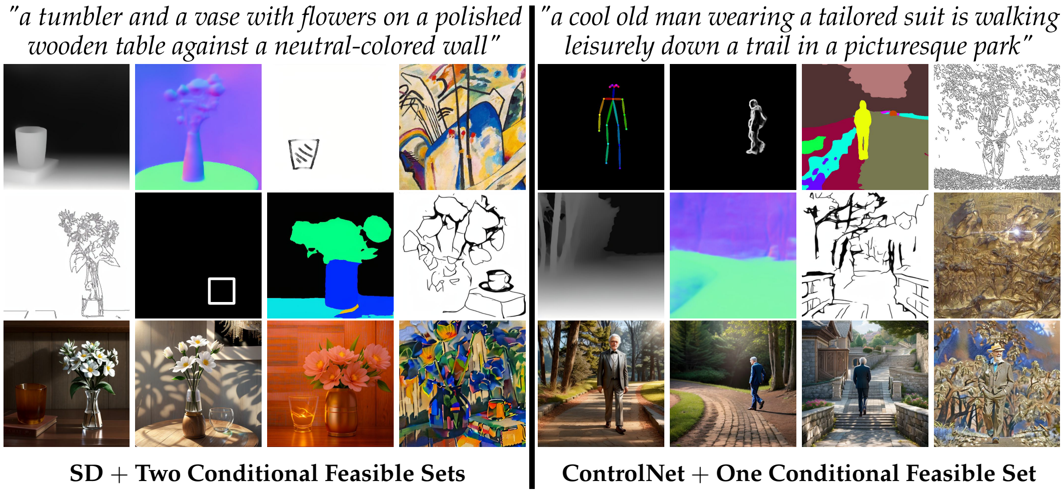


Figure 3: Image Generation with Multiple Controls: APCtrl incorporates multiple conditions into diffusion models. To showcase its capabilities, we present two illustrative examples, each detailed in dedicated sections of the figure. The left example demonstrates the fusion of two conditional sets. The right example leverages ControlNet to project onto the feasible space. Each subfigure includes the prompt text at the top, followed by rows for conditional image controls, concluding with the controlled generation results at the bottom.

4 Experiments

In this section, we provide a comprehensive evaluation of our method through both quantitative and qualitative analyses, demonstrating its effectiveness. Additionally, we highlight its versatility by showcasing compatibility with a range of diffusion backbones and samplers. Finally, we underscore the efficiency of our approach.

4.1 Experimental Setup

We utilize SDv1.5, a prevalent checkpoint, as the backbone for constructing latent control networks. This section details the latent control networks' training and elucidates the nuances of APCtrl sampling.

Latent Control Networks Training: Our model utilizes the identical U-Net architecture found in SDv1.5 and was initialized using the SDv1.5 checkpoint. The training process was conducted for fewer than 24 GPU hours on a single NVIDIA 3090 GPU. We employed approximately 118,000 images from the COCO2017 dataset (Lin et al., 2014) across various tasks. For human pose estimation, we specifically selected a subset of around 6,500 images from the COCO2017 dataset, concentrating on the "people" category. Additionally, for Style Guidance, we incorporated approximately 81,000 images from the Wiki-Art dataset.

APCtrl Sampling: A multitude of sampling strategies, including DDPM (Ho et al., 2020), DDIM (Song et al., 2020a), and LCM (Luo et al., 2023), can be applied to stable diffusion models. These strategies can be unified by the concept of recursive projection, expressed as $\mathbf{z}_t = \text{Proj}_{\mathfrak{D}_t}(\mathbf{z}_{t+1})$. Consequently, $\text{Proj}_{\mathfrak{I}_t}(\cdot)$ is able to integrate with all of these approaches to form our APCtrl sampling based on Algorithm 1.

4.2 Conditional Generation Results

APCtrl is adept at integrating a diverse set of conditions directly into the image generation process of diffusion models, offering a framework for sophisticated control over the generation outcomes. To showcase this capability, we demonstrate ten single-condition cases in Figure 2, spanning a spectrum of techniques from Canny edge (Canny, 1986), M-LSD line (Gu et al., 2022), HED edge (Xie & Tu, 2015), Skeleton (Cao et al., 2017), Object Loca-

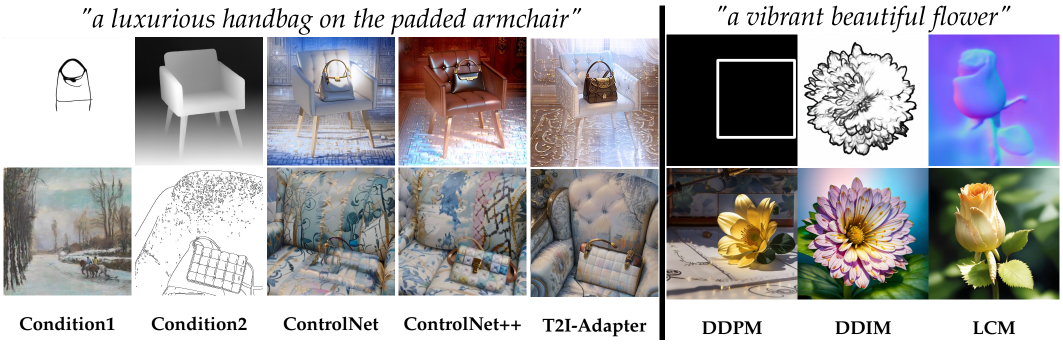


Figure 4: Compatibility Demonstration for Different Diffusion Backbones and Samplers. In left part, APCtrl introduces Condition 1, while the conditional diffusion backbone of the other model introduces Condition 2. The right part shows the generation results using DDPM (Ho et al., 2020), DDIM (Song et al., 2020a), and LCM (Luo et al., 2023).

Table 2: Quantitative Comparison for Controllable Generation on Single Conditions. The best results are in bold. "-" indicates that the method does not provide a public model.

	Method	Depth	Canny	HED	M-LSD	Segmentation	Normal	Skeleton	Location	Sketch
FID↓	ControlNet	19.3801	**16.5297**	19.9832	**19.7612**	**20.7019**	27.8210	57.4144	-	-
	ControlNet++	**17.4087**	20.1109	**15.8372**	-	24.3453	-	-	-	-
	T2I-Adapter	23.8945	17.0756	-	-	21.7609	-	**34.3569**	-	28.8883
	UniCtrlNet	24.9604	17.9107	17.4471	27.7329	22.7066	-	66.5560	-	**24.0166**
	UniControl	24.2885	18.9211	19.2913	-	29.8068	29.5817	40.7635	29.6951	-
	GLIGEN	23.2859	24.8351	26.3622	-	27.3867	27.6916	53.2974	**23.1426**	-
	APCtrl	25.0148	25.3836	24.7542	26.9950	25.1360	**27.4390**	43.5083	33.8875	26.6784
CLIP-scores↑	ControlNet	0.2840	0.2897	0.2870	0.2843	0.2838	0.2730	0.2610	-	-
	ControlNet++	0.3061	**0.3085**	0.3008	-	0.2997	-	-	-	-
	T2I-Adapter	0.2990	0.3045	-	-	0.2956	-	0.3111	-	0.2708
	UniCtrlNet	0.3017	0.3044	**0.3039**	0.2873	0.3052	-	0.2795	-	0.2927
	UniControl	**0.3063**	0.3032	0.2997	-	**0.3069**	0.2967	**0.3089**	**0.2974**	-
	GLIGEN	0.2979	0.2966	0.2806	-	0.2854	0.2718	0.2615	0.2762	-
	APCtrl	0.2952	0.3029	0.3035	**0.2951**	0.2989	**0.2943**	0.2920	0.2868	**0.3006**
CLIP-acs ↑	ControlNet	5.1861	5.2112	5.2536	5.3072	5.2954	5.0815	5.2418	-	
	ControlNet++	5.2945	5.1216	5.1125	-	4.9270	-	-	-	-
	T2I-Adapter	5.0973	5.1213	-	-	4.9737	-	5.2956	-	4.8516
	UniCtrlNet	5.0129	5.0010	5.0048	4.9704	5.0557	-	4.9568	-	5.0048
	UniControl	5.3498	5.1650	5.1683	-	**5.3920**	5.1061	**5.4802**	5.2630	-
	GLIGEN	5.1342	5.0485	4.9547	-	4.9362	4.7384	4.8970	5.3708	-
	APCtrl	**5.4225**	**5.4264**	**5.4575**	**5.4341**	5.3905	**5.4284**	5.4794	**5.3977**	**5.4730**
Controllability		MSE↓	SSIM↑	SSIM↑	SSIM↑	mIoU↑	MSE↓	mAP↑	mAP↑	SSIM↑
	ControlNet	88.9629	0.4376	0.5845	0.7552	0.4223	86.8804	0.4413	-	
	ControlNet++	86.7270	**0.5386**	**0.6907**	-	**0.5481**	-	-	-	-
	T2I-Adapter	94.5548	0.3984	-	-	0.2339	-	**0.4979**	-	0.3756
	UniCtrlNet	99.3874	0.4679	0.6159	0.7250	0.3037	-	0.2046	-	**0.6704**
	UniControl	88.8402	0.5340	0.3614	-	0.3273	104.1840	0.2463	**0.2731**	-
	GLIGEN	81.1289	0.3917	0.4094	-	0.2481	90.3527	0.1817	0.2418	-
	APCtrl	**86.6756**	0.4412	0.4752	**0.7793**	0.3916	**70.3443**	0.4043	0.2563	0.6118

tion (Redmon et al., 2016), Depth Map (Yang et al., 2024a), Normal Map (Vasiljevic et al., 2019), Segmentation (Cheng et al., 2022), Style Guidance (Radford et al., 2021). These cases serve to illustrate the versatility of APCtrl in handling different conditional inputs.

Additionally, APCtrl can be applied to multiple condition generation, which typically involve two strategies. The first strategy, as shown in the left part of Figure 3, involves augmenting the condition set and computing the projection function, denoted as $\mathrm{Proj}_{\mathfrak{I}_t}(\cdot)$, where $\mathfrak{I}_t$ is defined as the intersection of $\mathfrak{D}_t$ and $\mathfrak{C}_t$, with $\mathfrak{C}_t$ being redefined as the intersection of multiple condition sets, expressed as $\mathfrak{C}_t = \bigcap_{i=1}^{N} \mathfrak{C}_t^{(i)}$. We can then extend Equation (9) to calculate the updated projection $\mathrm{Proj}_{\mathfrak{I}_t}(\cdot)$. The alternative method, as depicted in the right part of Figure 3, involves utilizing ControlNet to define the projection onto $\mathfrak{D}_t$. Given that ControlNet includes a control mechanism, it can be effectively integrated with other constraints defining the condition set.

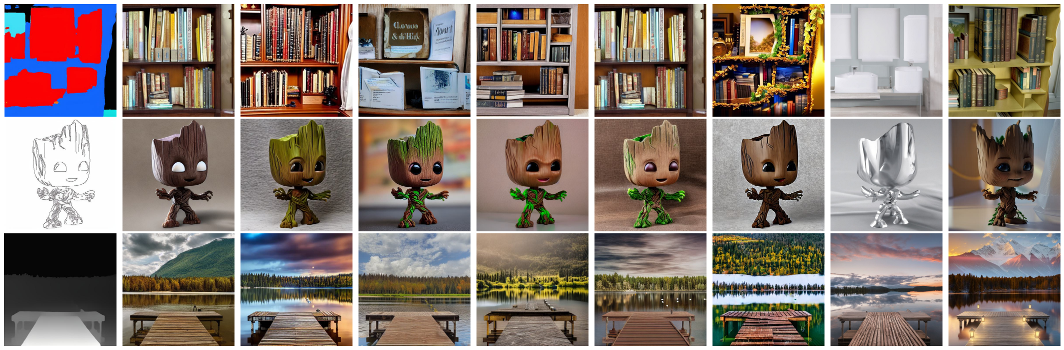


Figure 5: Visual Comparison of Single Condition for Control-on-Training Methods. The prompt for each row is "a book shelf", "a groot toy", and "brown wooden dock near lake". Although our method is a Control-on-Sampling approach that doesn't necessitate retraining the control network, the conditional results of Segmentation, Canny Edge, and Depth Map, are competitive with other methods.

4.3 Compatibility For Different Diffusion Backbones and Samplers

Figure 6: Visual Comparison of Single Condition for Control-on-Sampling Methods. The current Control-on-Sampling methods do not provide the same variety of conditions as Control-on-Training methods, as shown in Figure 5. In this comparison, we focus on two prevalent scenarios, Segmentation and Style Guidance.

APCtrl is compatible with any diffusion model that utilizes the same encoder $\mathcal{E}(\cdot)$ as adopted in Equation 3. This compatibility is exemplified in the left side of Figure 4, where APCtrl is integrated with models such as ControlNet (Zhang et al., 2023), ControlNet++ (Li et al., 2024b), and T2I-Adapter (Mou et al., 2024). In these integrations, APCtrl supplies Condition 1, complemented by Condition 2 from the respective diffusion backbones.

As per Algorithm 1, APCtrl integrates an intersection projection $\mathrm{Proj}_{\mathfrak{I}_t}(\cdot)$ into the diffusion model's reverse process, enhancing it without altering the original computations. This integration thus is sampling agnosticism, allowing APCtrl to be versatile with various sampling techniques. The right side of Figure 4 illustrates APCtrl's application and effectiveness with different sampling methods, including DDPM (Ho et al., 2020), DDIM (Song et al., 2020a), and LCM (Luo et al., 2023). These examples underscore APCtrl's adaptability across a range of diffusion model sampling approaches.

4.4 Quantitative and Qualitative Comparison

Quantitative Evaluation: Our quantitative assessment is conducted on the COCO2017 (Lin et al., 2014) validation set at a 512×512 resolution. This dataset comprises 5000 images, each with multiple captions. For our evaluation, we randomly select one caption per image, yielding 5000 generated images. Specifically, for the skeleton, we focused on the "people" category and selected2900 images. All methods employ 20 DDIM steps for fast evaluation. We evaluate generation quality using FID (Heusel et al., 2017), CLIP text-image score (Radford et al., 2021), CLIP aesthetic score (Schuhmann et al., 2022). We also evaluate controllability using SSIM(Structural Similarity), mAP(mean Average Precision),

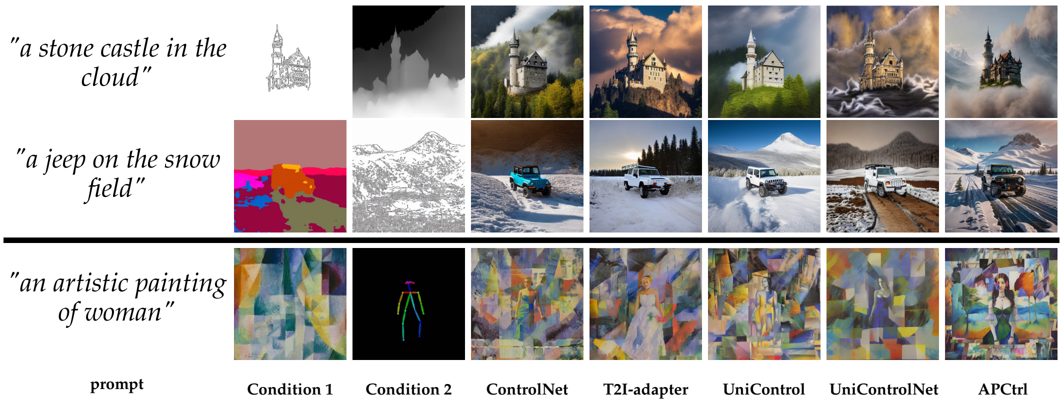


Figure 7: Comparison for Controllable Generation on Multiple Conditions. ControlNet (Zhang et al., 2023), T2I-Adapter (Mou et al., 2024), UniControl (Qin et al., 2023), UniControlNet (Zhao et al., 2024), and APCtrl combine two conditions to generate the final image. These methods initially lack style guidance capabilities, but integrating APCtrl provides this functionality to both. The results are demonstrated in the bottom row

MSE(Mean Squared Error), mIoU(Mean Intersection over Union). We take these metrics to compare the conditions extracted with the natural images (ground truth) with the generated images and report statistical data in Table 2.

Table 2 enumerates six Control-on-Training methods: ControlNet (Zhang et al., 2023), ControlNet++ (Li et al., 2024b), T2I-Adapter (Mou et al., 2024), UniControlNet (Zhao et al., 2024), UniControl (Qin et al., 2023), and GLIGEN (Li et al., 2023). The exclusion of Control-on-Sampling methods is due to their inability to uniformly address all presented conditions and the impracticality of their long inference times for 5,000 images. Notably, APCtrl stands as the pioneering Control-on-Sampling method capable of handling the full spectrum of conditional generation cases associated with Control-on-Training methods. The quantitative analysis in the table demonstrates our model's superiority over existing approaches in performance metrics—FID, CLIP-score, and CLIP-acs—and controllability, as evidenced in the respective rows and the Controllability column, across the majority of conditions.

Qualitative Evaluation: We enrich our analysis with qualitative comparisons of single and multi-condition controls, as depicted in Figures 5, 6, and 7. In Figure 5, seven Control-on-Training methods—ControlNet++ (Li et al., 2024b), T2I-Adapter (Mou et al., 2024), UniControlNet (Zhao et al., 2024), UniControl (Qin et al., 2023), GLIGEN (Li et al., 2023), and Ctrl-Adapter (Lin et al., 2024)—exhibit strong performance in Segmentation, Canny Edge, and Depth Map conditions. Our method also shows competitive alignment with the input conditions. To the best of our knowledge, APCtrl is a groundbreaking Control-on-Sampling method, capable of delivering the full range of condition controls associated with Control-on-Training methods. For more visual results, please refer to the Appendix B.

For Control-on-Sampling methods, only UniGuidance (Bansal et al., 2024) and FreeDoM (Yu et al., 2023) results are featured due to MultiDiffusion's specialization in image merging (Bar-Tal et al., 2023) and ReSample's (Song et al., 2023) focuses on solving linear inverse problems. Figure 6 documents the generation results for Segmentation and Style Guidance conditions from UniGuidance, FreeDoM, and our APCtrl.

Given the limitations of GLIGEN and Uni-Guidance in multi-condition scenarios and the restricted conditions of ControlNet++, our multi-condition comparison is narrowed down to ControlNet, T2I-Adapter, UniControlNet, and UniControl, as shown in Figure 7. APCtrl's performance in multi-condition tasks is comparable to these Control-on-Training methods.

Originally lacking in style guidance for the four methods, the integration with APCtrl has unlocked this capability, as evidenced in the figure's bottom row, highlighting APCtrl's seamless integration and multi-condition generation capabilities when paired with these methods. For more visual results, please refer to the Appendix C.

Table 3: Efficiency Comparison. APCtrl achieves a balance between training efficiency and sampling speed. Compared to Control-on-Training methods, it requires minimal training investment. When compared to Control-on-Sampling methods, APCtrl provides accelerated sampling.

Method	Control-on-Training						Control-on-Sampling			
	ControlNet	ControlNet++	T2I-Adapter	UniCtrlNet	UniControl	GLIGEN	FreeDom	DSG	UniGuid	APCtrl
Training (GPU Hours)	500	60	192	6900	5000	1000	-	-	-	20
Sampling (Seconds)	3	2	2	4	5	7	115	122	4510	13

4.5 Efficiency Comparison

This section is devoted to compare the efficiency of training and sampling. The latent control network in APCtrl projects the image space onto the conditional space, aligning it with the encoded constraints $\mathcal{E}(\boldsymbol{I}_c)$. This method is significantly different from Control-on-Training, as we believe that working with constraint spaces is less complex than the refinement processes in Control-on-Training. This suggests that APCtrl should require less training time, as indicated in Table 3. We observe that ControlNet++ requires only 60 hours because it fine-tunes from a ControlNet checkpoint. The control function of APCtrl is also applied during the sampling phase. It is essential to compare APCtrl's sampling efficiency with that of Control-on-Sampling methods. Thanks to the use of Alternative Projection, APCtrl notably decreases sampling time, as demonstrated in the last line of Table 3. This makes APCtrl a promising alternative, offering a favorable compromise for both training and sampling phases.

4.6 Impact of Iteration Count on Up/Down Alternative Projections

In Algorithm 1, we carry out N iterations of the Joint Up/Down Projection $\widehat{\text{Proj}}_{\mathfrak{J}_t}(\cdot)$, following the definition given in Equation 9. When N is small, the resulting images may not align with the desired conditions, as evidenced in Figure 8b. Conversely, a large N can lead to images that meet the conditions but display pronounced color discrepancies, diminishing their aesthetic quality, as shown in Figure 8d. Striking the optimal balance with N ensures both conditional fidelity and visual appeal, as demonstrated in Figure 8c.

(a) **Cond** (b) $N = 1$ (c) $N = 3$ (d) $N = 30$

Figure 8: Visualization For iteration Count N

5 Conclusions

Achieving controllable generation remains one of the significant challenges in diffusion models. We propose a novel direction to address this challenge. Our approach begins with reinterpreting the diffusion sampling process as a series of recursive projections onto the denosing set, denoted as $\mathfrak{D}_t$. Consequently, a conditional control diffusion model can be viewed as a sequence of recursive projections onto the intersection of feasible sets, $\mathfrak{D}_t \cap \mathfrak{C}_t$, where $\mathfrak{C}_t$ represents the condition set. We employ an alternative projection technique to effectively implement these projections onto the intersection set $\mathfrak{D}_t \cap \mathfrak{C}_t$. This methodology offers several distinct advantages over previous efforts: 1. Multi-Condition Generation: Multi-condition generation can be easily implemented. 2. Model and Sampling Agnosticism: APCtrl maintains agnosticism regarding both the underlying model backbones and the sampling process. 3. Unified Control Loss: It allows for a unified control loss, facilitating the management of various control applications. 4. Efficiency: APCtrl significantly reduces both training and sampling times. We have conducted a comprehensive evaluation of our framework, yielding state-of-the-art results.

References

Shmuel Agmon. The Relaxation Method for Linear Inequalities. *Canadian Journal of Mathematics*, 1954.

Arpit Bansal, Eitan Borgnia, Hong-Min Chu, Jie S. Li, Hamid Kazemi, Furong Huang, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Cold Diffusion: Inverting Arbitrary Image Transforms Without Noise. In *Conference on Neural Information Processing Systems*, 2023.

Arpit Bansal, Hong-Min Chu, Avi Schwarzschild, Soumyadip Sengupta, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Universal Guidance for Diffusion Models. In *International Conference on Learning Representations*, 2024.

Omer Bar-Tal, Lior Yariv, Yaron Lipman, and Tali Dekel. Multidiffusion: Fusing Diffusion Paths for Controlled Image Generation. *Proceedings of Machine Learning Research*, 2023.

Stephen Boyd. Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. *Foundations and Trends® in Machine Learning*, 2010.

James P. Boyle and Richard L. Dykstra. A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces. In *Advances in Order Restricted Statistical Inference*. 1986.

L. M. Brègman. Finding the Common Point of Convex Sets by the Method of Successive Projection. *Dokl. Akad. Nauk SSSR*, 1965.

John Canny. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

Stanley H Chan. Tutorial on Diffusion Models for Imaging and Vision. *arXiv*, 2024.

Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention Mask Transformer for Universal Image Segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2022.

Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving Diffusion Models for Inverse Problems using Manifold Constraints. In *Advances in Neural Information Processing Systems*, 2022.

Hyungjin Chung, Jeongsol Kim, Michael Thompson Mccann, Marc Louis Klasky, and Jong Chul Ye. Diffusion Posterior Sampling for General Noisy Inverse Problems. In *International Conference on Learning Representations*, 2023.

Florinel-Alin Croitoru, Vlad Hondru, Radu Tudor Ionescu, and Mubarak Shah. Diffusion Models in Vision: A Survey. *Transactions on Pattern Analysis and Machine Intelligence*, 2023.

Frank Deutsch. The Method of Alternating Orthogonal Projections. In *Approximation Theory, Spline Functions and Applications*. 1992.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, 2021.

Jim Douglas and Henry H Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society*, 1956.

Mattias Fält and Pontus Giselsson. Generalized Alternating Projections on Manifolds and Convex Sets. *Journal of Nonsmooth Analysis and Optimization*, 2024.

Alexandros Graikos, Nikolay Malkin, Nebojsa Jojic, and Dimitris Samaras. Diffusion Models as Plug-and-Play Priors. *Advances in Neural Information Processing Systems*, 2022.

Geonmo Gu, Byungsoo Ko, SeoungHyun Go, Sung-Hyun Lee, Jingeun Lee, and Minchul Shin. Towards Light-Weight and Real-Time Line Segment Detection. In *AAAI Conference on Artificial Intelligence*, 2022.

LG Gubin, Boris T Polyak, and EV Raik. The Method of Projections for Finding the Common Point of Convex Sets. *USSR Computational Mathematics and Mathematical Physics*, 1967.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Advances in Neural Information Processing Systems*, 2017.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, 2020.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising Diffusion Restoration Models. *Advances in Neural Information Processing Systems*, 2022.

Alexander Y Kruger and Nguyen H Thao. Regularity of Collections of Sets and Convergence of Inexact Alternating Projections. *Journal of Convex Analysis*, 2016.

Dongxu Li, Junnan Li, and Steven Hoi. Blip-Diffusion: Pre-Trained Subject Representation for Controllable Text-to-Image Generation and Editing. *Advances in Neural Information Processing Systems*, 2024a.

Ming Li, Taojiannan Yang, Huafeng Kuang, Jie Wu, Zhaoning Wang, Xuefeng Xiao, and Chen Chen. Controlnet++: Improving conditional controls with efficient consistency feedback. In *European Conference on Computer Vision*, 2024b.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. GLIGEN: Open-Set Grounded Text-to-Image Generation. In *Conference on Computer Vision and Pattern Recognition*, 2023.

Han Lin, Jaemin Cho, Abhay Zala, and Mohit Bansal. Ctrl-Adapter: An Efficient and Versatile Framework for Adapting Diverse Controls to Any Diffusion Model, 2024.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *European Conference on Computer Vision*, 2014.

Pierre-Louis Lions and Bertrand Mercier. Splitting Algorithms for the Sum of Two Nonlinear Operators. *SIAM Journal on Numerical Analysis*, 1979.

Andreas Lugmayr, Martin Danelljan, Andres Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting Using Denoising Diffusion Probabilistic Models. In *Conference on Computer Vision and Pattern Recognition*, 2022.

Calvin Luo. Understanding Diffusion Models: A Unified Perspective. *arXiv*, 2022.

Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent Consistency Models: Synthesizing High-Resolution Images with Few-Step Inference, 2023.

Theodore Samuel Motzkin and Isaac Jacob Schoenberg. The Relaxation Method for Linear Inequalities. *Canadian Journal of Mathematics*, 1954.

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. T2I-Adapter: Learning Adapters to Dig Out More Controllable Ability for Text-to-Image Diffusion Models. In *AAAI Conference on Artificial Intelligence*, 2024.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *International Conference on Machine Learning*, 2021.

Alexander Quinn Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob Mcgrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. In *International Conference on Machine Learning*, 2022.

Gaurav Parmar, Taesung Park, Srinivasa Narasimhan, and Jun-Yan Zhu. One-Step Image Translation with Text-to-Image Models, 2024.

Can Qin, Shu Zhang, Ning Yu, Yihao Feng, Xinyi Yang, Yingbo Zhou, Huan Wang, Juan Carlos Niebles, Caiming Xiong, Silvio Savarese, et al. UniControl: A Unified Diffusion Model for Controllable Visual Generation In the Wild. In *Conference on Neural Information Processing Systems*, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models From Natural Language Supervision. In *International Conference on Machine Learning*, 2021.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition*, 2016.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis with Latent Diffusion Models. In *Conference on Computer Vision and Pattern Recognition*, 2022.

Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation. In *Conference on Computer Vision and Pattern Recognition*, 2023.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5B: An Open Large-Scale Dataset for Training Next Generation Image-Text Models. *Advances in Neural Information Processing Systems*, 2022.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning*, pp. 2256–2265. PMLR, 2015.

Bowen Song, Soo Min Kwon, Zecheng Zhang, Xinyu Hu, Qing Qu, and Liyue Shen. Solving Inverse Problems with Latent Diffusion Models via Hard Data Consistency. In *International Conference on Learning Representations*, 2023.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising Diffusion Implicit Models. In *International Conference on Learning Representations*, 2020a.

Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. *Advances in Neural Information Processing Systems*, 2019.

Yang Song and Stefano Ermon. Improved Techniques for Training Score-based Generative Models. *Advances in Neural Information Processing Systems*, 2020.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations*, 2020b.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum Likelihood Training of Score-based Diffusion Models. *Advances in Neural Information Processing Systems*, 2021.

Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z Dai, Andrea F Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R Walter, et al. DIODE: A Dense Indoor and Outdoor DEpth Dataset, 2019.

John Von Neumann. *Functional Operators: The Geometry of Orthogonal Spaces*. Princeton University Press, 1951.

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-Shot Image Restoration Using Denoising Diffusion Null-Space Model. In *International Conference on Learning Representations*, 2022.

Saining Xie and Zhuowen Tu. Holistically-Nested Edge Detection. In *International Conference on Computer Vision*, 2015.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Conference on Computer Vision and Pattern Recognition*, 2024a.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion Models: A Comprehensive Survey of Methods and Applications. *ACM Computing Surveys*, 2023a.

Ling Yang, Jingwei Liu, Shenda Hong, Zhilong Zhang, Zhilin Huang, Zheming Cai, Wentao Zhang, and Bin Cui. Improving Diffusion-based Image Synthesis with Context Prediction. *Advances in Neural Information Processing Systems*, 2024b.

Lingxiao Yang, Shutong Ding, Yifan Cai, Jingyi Yu, Jingya Wang, and Ye Shi. Guidance with spherical gaussian constraint for conditional diffusion. In *International Conference on Machine Learning*, 2024c.

Zhengyuan Yang, Jianfeng Wang, Zhe Gan, Linjie Li, Kevin Lin, Chenfei Wu, Nan Duan, Zicheng Liu, Ce Liu, Michael Zeng, et al. ReCo: Region-Controlled Text-to-Image Generation. In *Conference on Computer Vision and Pattern Recognition*, 2023b.

Jiwen Yu, Yinhuai Wang, Chen Zhao, Bernard Ghanem, and Jian Zhang. FreeDoM: Training-Free Energy-Guided Conditional Diffusion Model. In *International Conference on Computer Vision*, 2023.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding Conditional Control to Text-to-Image Diffusion Models. In *International Conference on Computer Vision*, 2023.

Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K Wong. Uni-ControlNet: All-in-One Control to Text-to-Image Diffusion Models. *Advances in Neural Information Processing Systems*, 2024.

A Appendix: Details for Training latent control network

Our implementation, including all training and sampling files as well as the checkpoint, will be released upon acceptance of our paper. Briefly, our code is adapted from the Hugging Face Diffusers repository, specifically from the train_text_to_image.py script, which can be found at https://github.com/huggingface/diffusers. In this section, we detail the key components of the code responsible for constructing and training the latent control network $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\cdot, \cdot)$.

A.1 building Model code

```
from diffusers import UNet2DConditionModel

# Initialize the A model (UNet) from pretrained models
A_model = UNet2DConditionModel.from_pretrained(
    "runwayml/stable-diffusion-v1-5",  subfolder="unet"
    )
```

A.2 Training code

```
from diffusers import AutoencoderKL, UNet2DConditionModel

# Load the VAE model using the default configuration
vae = AutoencoderKL.from_pretrained(
    "runwayml/stable-diffusion-v1-5", subfolder="vae"
)

# Set up the optimizer for the A model
optimizer = torch.optim.AdamW(
    A_model.parameters(), lr=1e-5
)

for batch in dataloader:

    ...

    # Encode the ground truth image to the latent space
    latents = vae.encode(ground_truth_image)

    # Add noise to the latents
    noisy_latents = noise_scheduler.add_noise(
        latents, noise, timesteps)

    # Predict the conditionial latents from noisy latents
    pred_latents = A_model(
        noisy_latents, timesteps)

    # Encode the condition image to the latent space
    cond_latents = vae.encode(condition_image)

    # Calculate the loss
    loss = F.mse_loss(pred_latents_pred, cond_latents)

    ...
```

B Appendix: Details for APCtrl Sampling

B.1 Detailed Implementation for Algorithm 1

In this section, we delve into the specifics of APCtrl sampling as encapsulated by Algorithm 1. This algorithm outlines the step-by-step procedure for implementing our novel approach to conditional diffusion sampling, which leverages the power of latent control networks and alternative projections to achieve sophisticated image generation. The details provided here will walk through the algorithm's operations, explaining how each step contributes to the final output, ensuring a clear understanding of the methodology and its advantages over traditional approaches.

Based on the discussion in our paper, we have outlined the steps and principles of our method.

$$\mathrm{Proj}_{\boldsymbol{\mathcal{D}}_{t+1}}(\mathbf{z}_t) = \sqrt{\alpha_{t+1}}\mathbf{z}_t + \sqrt{1-\alpha_{t+1}}\boldsymbol{\epsilon}$$

$$\mathrm{Proj}_{\boldsymbol{\mathfrak{D}}_t}(\mathbf{z}_{t+1}) = \frac{1}{\sqrt{\alpha_{t+1}}}\mathbf{z}_{t+1} + \frac{(1-\alpha_{t+1})}{\sqrt{\alpha_{t+1}}}\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \sqrt{\frac{(1-\alpha_{t+1})(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t+1}}}\boldsymbol{\epsilon}$$

$$\mathrm{Proj}_{\boldsymbol{\mathfrak{C}}_t}(\mathbf{z}_{t+1}) = \mathrm{argmin}_{\mathbf{z}_t} \|\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \boldsymbol{\mathcal{E}}(\boldsymbol{I}_c)\|^2 \quad \text{solving with initial point } \mathbf{z}_t = \mathrm{Proj}_{\boldsymbol{\mathfrak{D}}_t}(\mathbf{z}_{t+1})$$

Note that $\mathrm{Proj}_{\boldsymbol{\mathfrak{C}}_t}(\mathbf{z}_{t+1})$ can solved by gradient decent. Considering these projections, they enable us to complete Algorithm 1 as follows:

Algorithm 2 APCtrl Sampling

Input: Initial noise $\mathbf{z}_T$ Diffusion Model $\boldsymbol{\mathcal{Z}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ Latent Control Network $\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t)$ Encoder $\mathcal{E}$ Condition $\boldsymbol{I}_c$ step size γ

Operator $\mathtt{Proj}_{\boldsymbol{\mathcal{D}}_{t+1}}(\mathbf{z}_t)$:
  return $\sqrt{\alpha_{t+1}}\mathbf{z}_t + \sqrt{1-\alpha_{t+1}}\boldsymbol{\epsilon}$
Operator $\mathtt{Proj}_{\boldsymbol{\mathfrak{D}}_t}(\mathbf{z}_{t+1})$:
  return $\frac{1}{\sqrt{\alpha_{t+1}}}\mathbf{z}_{t+1} + \frac{(1-\alpha_{t+1})}{\sqrt{\alpha_{t+1}}}\boldsymbol{\mathcal{S}}_{\boldsymbol{\theta}}(\mathbf{z}_{t+1}, t+1) + \sqrt{\frac{(1-\alpha_{t+1})(1-\bar{\alpha}_t)}{1-\bar{\alpha}_{t+1}}}\boldsymbol{\epsilon}$
Operator $\mathtt{Proj}_{\boldsymbol{\mathfrak{C}}_t}(\mathbf{z}_t)$:
  for $m = 1$ *to* M **do**
    $\mathbf{z}_t \leftarrow \mathbf{z}_t - \gamma \bigtriangledown_x \|\boldsymbol{\mathcal{A}}_{\boldsymbol{\theta}}(\mathbf{z}_t, t) - \boldsymbol{\mathcal{E}}(\boldsymbol{I}_c)\|$
  end
  return x
Operator $\mathtt{Proj}_{\boldsymbol{\mathfrak{I}}_t}(\mathbf{z}_t)$:
  $\mathbf{z}_t \leftarrow \mathtt{Proj}_{\boldsymbol{\mathcal{D}}_{t+1}}(\mathbf{z}_t)$
  $\mathbf{z}_t \leftarrow \mathtt{Proj}_{\boldsymbol{\mathfrak{C}}_t}(\mathbf{z}_t)$
  $\mathbf{z}_t \leftarrow \mathtt{Proj}_{\boldsymbol{\mathcal{D}}_{t+1}}(\mathbf{z}_t)$
  $\mathbf{z}_t \leftarrow \mathtt{Proj}_{\boldsymbol{\mathfrak{D}}_t}(\mathbf{z}_t)$
  return x
Operator $\mathtt{Proj}_{\boldsymbol{\mathfrak{I}}_t}(\mathbf{z}_t)$:
  for $n = 1$ *to* N **do**
    $\mathbf{z}_t \leftarrow \mathtt{Proj}_{\boldsymbol{\mathfrak{I}}_t}(\mathbf{z}_t)$
  end
  return x

$\mathbf{z}_t \leftarrow \mathbf{z}_T$
for $t = T-1$ *to* 1 **do**
  $\mathbf{z}_t \leftarrow \mathrm{Proj}_{\boldsymbol{\mathfrak{D}}_t}(\mathbf{z}_{t+1})$
  if *conducting APCtrl Sampling* **then**
    $\mathbf{z}_t \leftarrow \mathrm{Proj}_{\boldsymbol{\mathfrak{I}}_t}(\mathbf{z}_t)$
  end
end
Output: x_t

B.2 Pseudocode for Implementing Algorithm 1

This pseudocode outlines the steps to implement the alternative projection method as described in Algorithm 1. It starts with an initial point and iteratively applies projections onto the given sets until the desired number of iterations is reached, resulting in a point that is close to the intersection of the two sets.

```
def project_to_cond(z_t, A_model, cond_latents):
    for n in range(N):
        optimizer = torch.optim.RMSProp([z_t], lr=preset_lr)
        optimizer.zero_grad()

        # Get the predicted conditional latents
        pred_latents = A_model(z_t, t)

        # Calculate the deviation
        # between predicted latents and conditional latents.
        loss = F.mse_loss(pred_latents, cond_latents)

        # Make the current latents better match the conditions.
        loss.backward()
        optimizer.step()

    return z_t


def alternative_projection_sampling(z_t, A_model, cond_latent):
    for m in range(N):
        z_t = add_noise(z_t)        # up projections
        z_t = project_to_cond(z_t, A_model, cond_latent)
        z_t = add_noise(z_t)        # up projections
        z_t = default_denoise(z_t)  # original denoising process

    return z_t

# Load the trained A_model
A_model = UNet2DConditionModel.from_pretrained(
    "trained_checkpoint", subfolder="unet"
)

# Encode the condition image into the latent space
cond_latents = vae.encode(condition_image)

# sampling
for t in timesteps:
    z_t = default_denoise(z_t)    # original denoising process
    if do_APCtrl:
        z_t = alternative_projection_sampling(
            z_t, A_model, cond_latents
        )
```

C Appendix: More Qualitative Evaluation

We present additional visual results, including various single-condition cases. Moreover, we also demonstrate more results under multiple conditions.

"a red telephone booth near a lantern in the street"

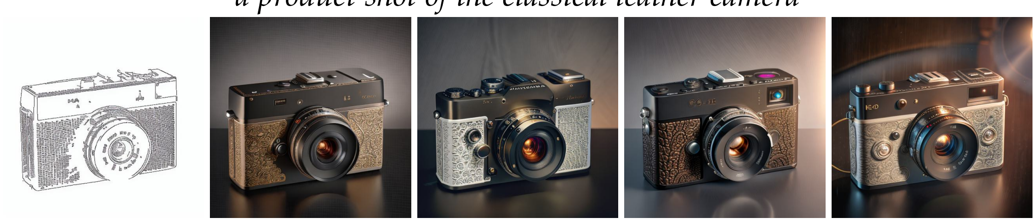
"a hammock hanged on the tree on the beach"

"a white and black drum set on the grass"

Depth Map

"slices of bread on the wooden chopping board"

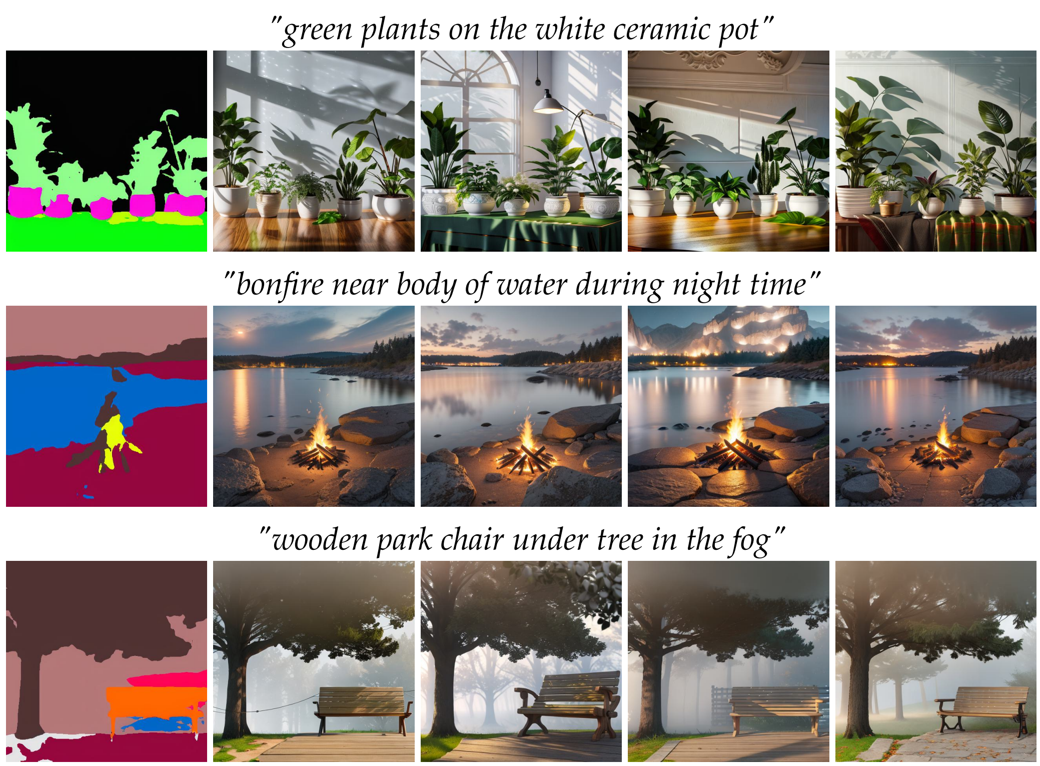
"a product shot of the classical leather camera"



"a majestic waterfall down the rock cliffside"

Canny Edge

Style Guidance

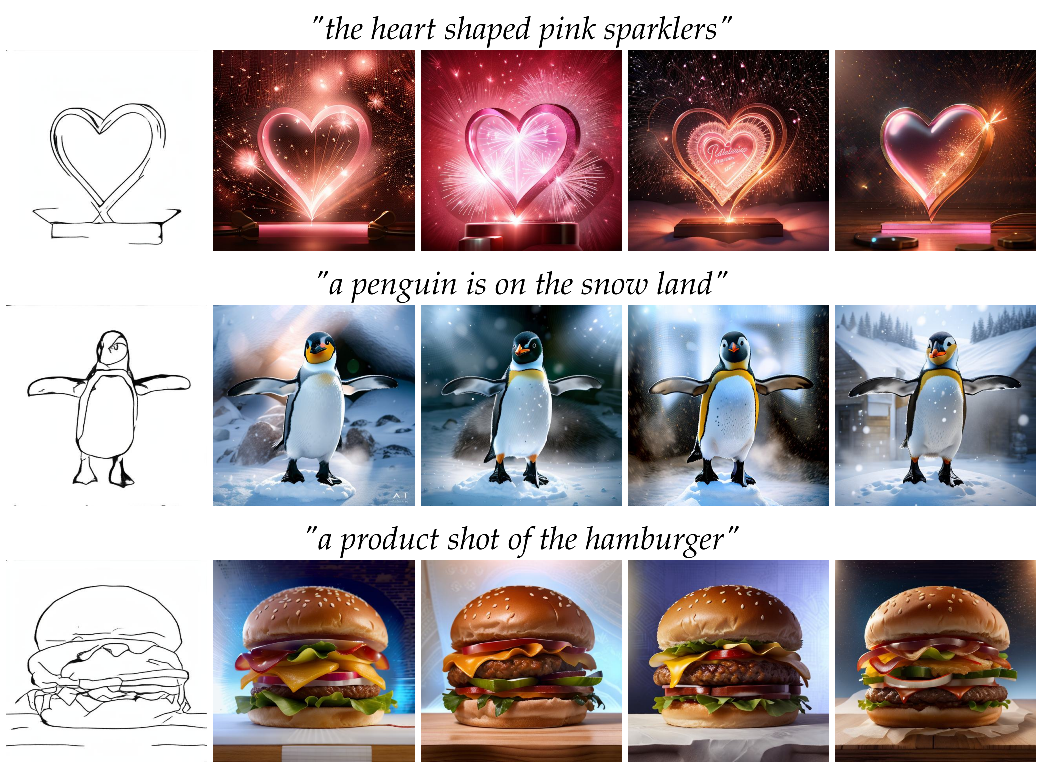


Segmentation

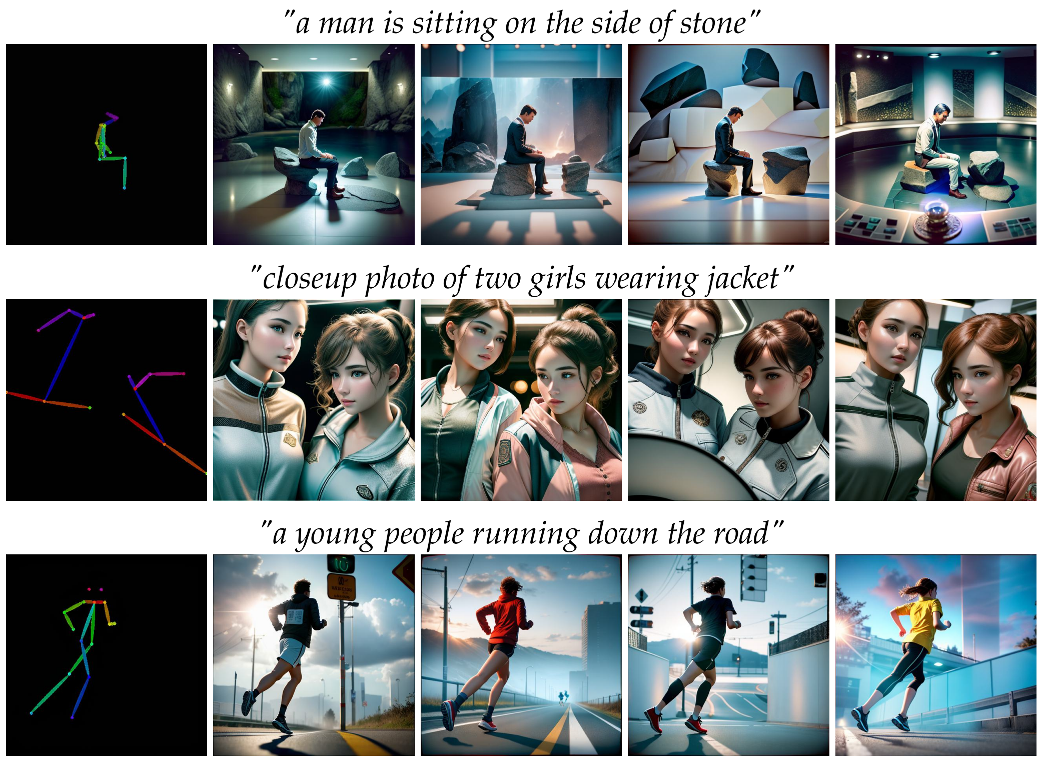
Normal Map

HED Edge

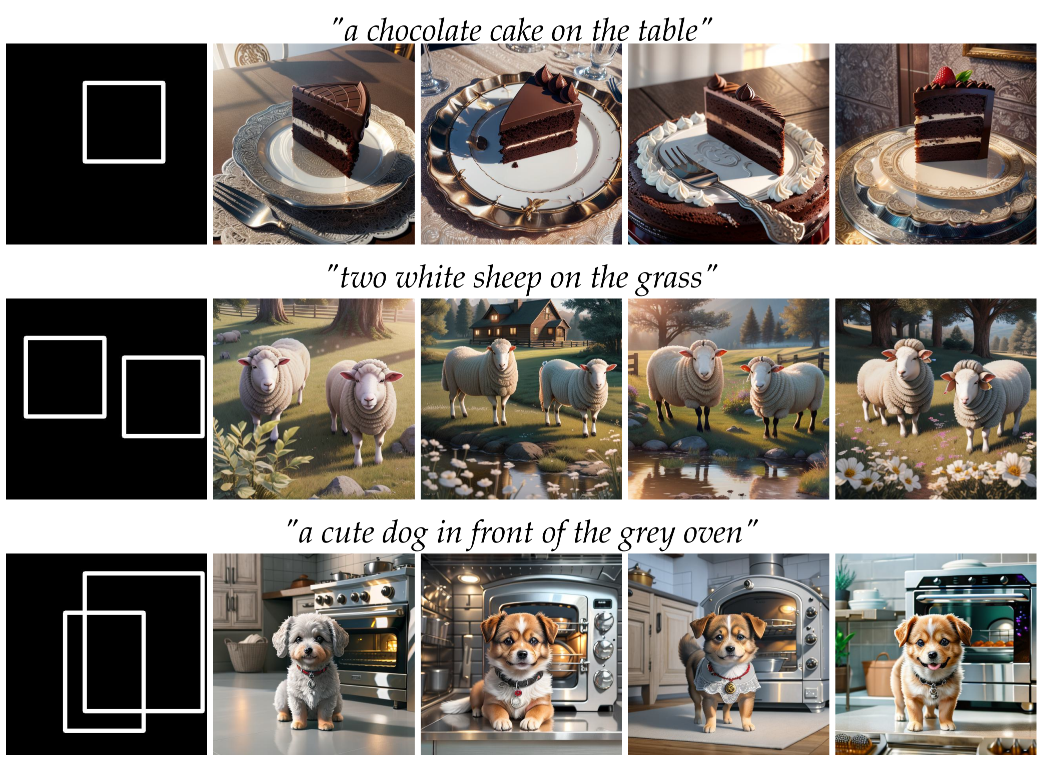
M-LSD Line



Sketch

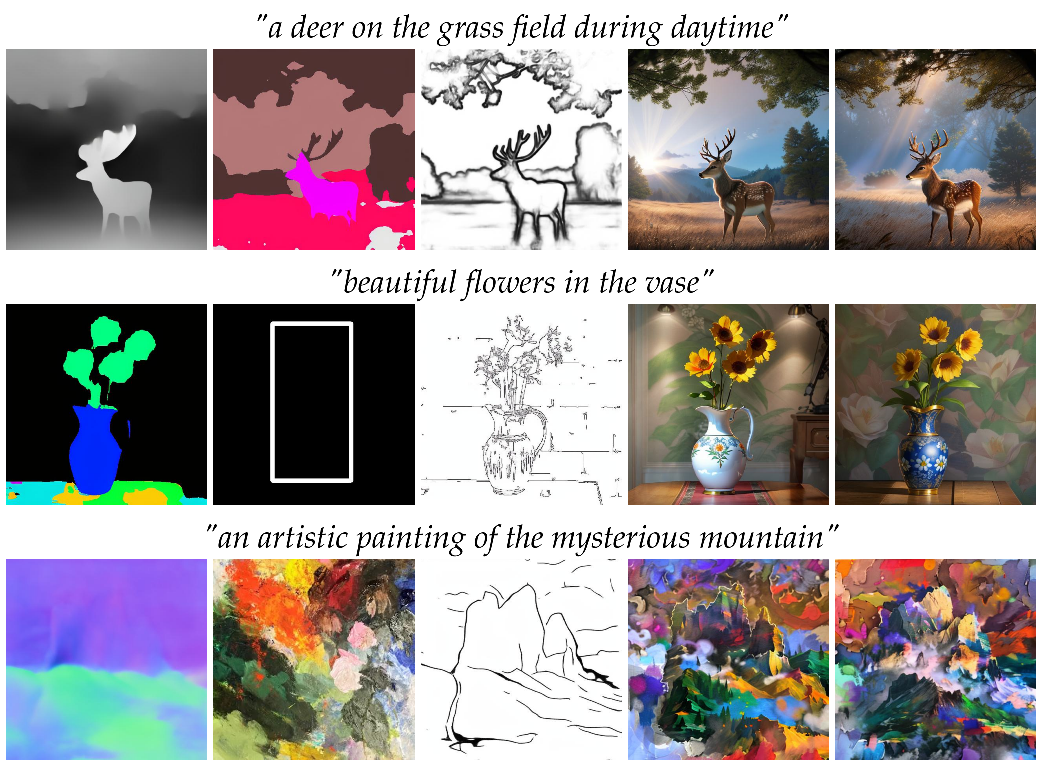


Skeleton



Object Location

Two Conditions



Three Conditions