Constrained Reverse Diffusion

Alexander Rosen

Abstract—Diffusion models have revolutionized computer vision by achieving unprecedented quality, fidelity, and diversity across various applications. To extend this success to real-world tasks, we believe the principles and heuristics of diffusion models must be adapted to incorporate physical constraints into their formulation. We make this adjustment by discarding the first-order Markov chain and synthesizing a new reverse process with stochastic linear interpolation and a differentiable physics engine. This results in a generative model that incorporates constraints into its sampling process while matching the performance of unconstrained models on numerous computer vision tasks, ranging from image denoising to 2D point cloud reconstruction.

Index Terms—Diffusion, Generative Models, Constrained Generation

1 Introduction

Over the past four years computer vision tasks such as image generation, image denoising, and point cloud generation have been taken over by Denoising Diffusion Probabilistic Models (DDPM) [1] and variants thereof. Diffusion models and DDPM were originally inspired by a random walk from non-equilibirum thermodynamics, where we define a process to gradually move from a training sample to noise we call an equilibrium state [2]. This perspective unveiled a new framework to train a generative model by learning to reverse this process moving from an equilibrium state to a high-concentration sample resembling our dataset.

It turns out that this point of view for training a generative model has many deep and intuitive properties, from an absence of joint training objectives to the reverse random walk encouraging a diverse set of samples. Still, one intriguing property remains elusive: How do we bake constraints into the reverse process or final sample? How might we sample a configuration of objects in a room with guarantees they don't overlap? In this work, we pave a path forward for diffusion-esque algorithms that facilitate sampling under geometric constraints; we construct an algorithm as general as current diffusion literature, show our algorithm is on par with DDPM for image generation and denoising tasks, and provide examples of our algorithm handling obstacles and collisions for 2D point cloud generation.

2 Related Work

2.1 Denoising Diffusion Probabilistic Models

DDPMs belong to a class of generative models whose sampling process involves moving backwards through a finite first-order Markov chain starting from an isotropic Gaussian noise equilibrium. Given the previous element x_{t-1} in our chain (starting with a sample x_0 from our training set), we define the forward process for x_t as

$$q(x_t|x_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}x_{t-1}, \beta_t\mathbb{I}) \tag{1}$$

which we can sample directly from x_0 as

$$q(x_t|x_0) = \mathcal{N}(\sqrt{\overline{\alpha}_t}x_0, (1-\overline{\alpha}_t)\mathbb{I}) \tag{2}$$

- *Alexander Rosen is an undergraduate student at the University of Toronto E-mail: a.rosen@mail.utoronto.ca*

for $\alpha_t = 1 - \beta_t$ and $\overline{\alpha}_t = \prod_{i=1}^{t} \alpha_t$. This forward process is repeated for T steps (usually 1000), and we choose $\{\beta_t\}_{t=1}^{T}$ so that equilibrium x_T is approximately standard normal.

At inference time, we seek to start with a sample x_T and repeatedly sample x_{t-1} given x_t until we reach x_0. Using (2) with the reparameterization trick, we can write

$$x_0 = \frac{1}{\sqrt{\overline{\alpha}_t}}\left(x_0 - \sqrt{1-\overline{\alpha}_t}\epsilon\right) \tag{3}$$

for $\epsilon \sim N(0,1)$, allowing us to show the mean of the reverse posterior

$$q(x_{t-1}|x_t, x_0) = \frac{q(x_t|x_{t-1}, x_0)q(x_{t-1}|x_0)}{q(x_t|x_0)} \tag{4}$$

is

$$\mu = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\overline{\alpha}_t}}\epsilon\right) \tag{5}$$

So, we train a network to predict this ϵ given x_t and t, and sample x_{t-1} as this μ with small additive Gaussian noise dependent on t.

2.2 Extensions of Diffusion

With the success of this method came many subsequent alterations, such as speeding up the reverse process [3], reasoning about a more suitable variance schedule [4], or performing the diffusion process in a latent space [5]. The relevant follow-up work we choose to focus on is cold diffusion [6], which generalizes the idea of diffusion to non-Gaussian or even deterministic equilibria. A much simpler algorithm, cold diffusion attempts to predict back x_0, learning a restoration operation r_θ given some degradation $x_t = d(x_0, t)$:

$$x_0 \approx r_\theta(d(x_0, t)) \tag{6}$$

To sample an x_0 using our model, we start with x_T, predict $\hat{x}_0 = r(x_T, T)$, run this prediction through our forward process to produce an $x_{t-1} = d(\hat{x}_0, t)$, and repeat until timestep zero.

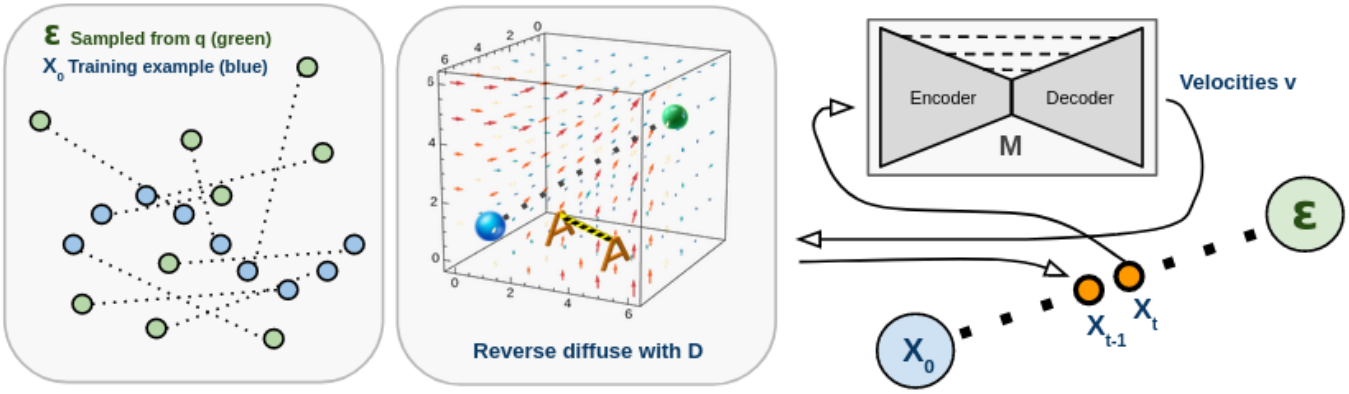

Fig. 1. Backward prediction process

2.3 Constrained Problems

Incorporating constraints into the reverse process and final generation are typically problem specific, resulting in a variety of different methods. The focus of this report is on ensuring that the generation quality and diversity of our proposed method is up to the standard set by current generative models, meaning we will not be comparing our work to methods for layout planning, object reorganization, etc. Moving forward, we hope to compare our work against diffusion policies [7] and DDPM-based methods whose samples approximately satisfy constraints in a scene [8].

3 Proposed Method

3.1 A New Generative Model

The formulations of diffusion we have seen in the related work section rely on an intermediate prediction of x_0, whether it's explicit with cold diffusion or implicitly through the mean of a posterior with DDPM. In order to account for constraints during sampling or constraints in the final result, we need some process involving these constraints during training, which we can introduce by having a direct prediction back one timestep opposed to an intermediate prediction or sampling process through x_0.

Our method begins by removing the Markov chain from the diffusion process and replacing it with linear interpolation between equilibria noise and training examples. At training time, we sample a timestep uniformly from $\{0, \dots, T\}$, a training example x_0 and an equilibrium $\epsilon \sim q$ from any deterministic or non-deterministic distribution q. Lying on the line segment between ϵ and x_0, we define

$$x_t = \left(1 - \tfrac{1}{T}\right) x_0 + \tfrac{t}{T}\epsilon \tag{7}$$

that approaches ϵ as $t \to T$ and x_0 as $t \to 0$. To devise a sampling algorithm to obtain x_{t-1} given x_t, we can compute x_{t-1} and x_t using the same source of randomness ϵ, and directly predict x_{t-1} without conditioning on x_0.

3.2 Reformulation with Velocity

Now that we have a method to directly predict x_{t-1}, we can rephrase this prediction to incorporate constraints; instead of a model learning x_{t-1}, suppose we learn an initial velocity vector to apply to points in space x_t. Then, we can feed these velocity vectors and initial positions x_t to a simulation, run this simulation, and record the final positions as our prediction for x_{t-1}. That is, for a differentiable physics engine $D(x, v)$ taking initial positions x and velocities v to final positions, a model M, and some loss $\mathcal{L}$ between $\hat{x}_{t-1}$ and x_{t-1}, we substitute the gradient descent step on

$$\nabla \mathcal{L}(x_{t-1}, M(x_t, t)) \tag{8}$$

for a step on

$$\nabla \mathcal{L}(x_{t-1}, D(x_t, M(x_t, t))) \tag{9}$$

This introduction of D allows the reverse process to move through space under constraints, such as object-to-object collisions, object-to-scene collisions, deformations, a background vector field, etc.

At inference time, our algorithm remains similar to DDPM and cold diffusion. We sample $x_t = x_T \sim q$, and repeatedly compute $x_{t-1} = D(x_t, M(x_t, t))$ down to timestep zero. When performing inference in the real-world rather than simulation, the velocity vectors $M(x_t, t)$ would be applied to the objects in positions x_t and x_{t-1} would be measured from the scene after some fixed period of time. The differentiable physics engine D may also be used to introduce small sources of randomness during training time simulations to ensure final positions are not fully determined by x_T and our learned velocities are robust against discrepancies between simulation and the real-world.

4 Experimental Results

For our experimental results, we test our algorithm's image generation and image denoising capabilities against DDPM, as well as providing qualitative results for 2D point cloud generation.

4.1 Setup

For image generation and denoising, we train on the 32 by 32 CIFAR-10 dataset. To evaluate image generation quality, we extract feature vectors for generated images and images sampled from the training set using the final convolutional layer of Inception-v3 [9] after upsampling to 224 by 224. Then, we compute FID scores [10] with these feature vectors to quantify the quality and diversity of the generated

Algorithm 1 Train(x_0)

$t \sim \text{Unif}(1, T), \epsilon \sim q$
$x_t \leftarrow (1 - \frac{t}{T})x_0 + \frac{t}{T}\epsilon$
$x_{t-1} \leftarrow (1 - \frac{t-1}{T})x_0 + \frac{t-1}{T}\epsilon$
$v \leftarrow M(x_t, t)$
$\hat{x}_{t-1} \leftarrow D(x_t, v)$ ▷ Positions from simulation
Optimize with $\mathcal{L}(x_{t-1}, \hat{x}_{t-1})$

Algorithm 2 Sampling

$x_t \sim q$
for $t \leftarrow T$ to 1 **do**
    $v \leftarrow M(x_t, t)$
    $x_t \leftarrow D(x_t, v)$
end for
return x_t ▷ Sampled x_0

Fig. 2. Training and sampling algorithms

images. To evaluate the same models on image denoising, we compute a collection of $T - 1$ denoised images where the backward denoising process starts as each possible x_t. Of these images, we record PSNR and SSIM as the maximum over the collection, ensuring our comparisons between denoising methods have optimal starting points for each respective backward process. PSNR and SSIM are computed for 30 CIFAR-10 images with additive noise sampled from a $\mathcal{N}(0, 0.1)$ Gaussian.

For 2D point cloud generation, we learn to reconstruct a single point cloud with DDPM and our method to provide intuition for the reader and motivation for future work.

4.2 Implementation Details

Each model for image generation and denoising uses the same U-Net [11]. The convolutional block is composed of convolutions, group normalization layers [12], and ReLU non-linearities, as well as a residual connection [13]. A down block is composed of a single convolutional block and max pooling, and an up block is composed of a transposed convolution, two convolutional blocks, and a concatenated skip connection from the corresponding down block. There are three up and down blocks, and the middle consists of average pooling and time siren [14] to condition the model on timesteps t, which we found to accelerate convergence for each diffusion algorithm.

For 2D point cloud generation, we found an MLP with three 128-dimensional hidden layers to be sufficient. The input to the model was a concatenation of points from the point cloud and an embedding of the current timestep, both with separate sinusoidal positional encoding [15].

Each image model was trained for 200 epochs (<1hr on an A6000 GPU) with a batch size of 100. Point cloud models were trained with various configurations in under 5 minutes. The total number of timesteps T was set to 1000 for cold diffusion and DDPM and 100 for our method since we found linear interpolation required fewer timesteps for diverse generations.

4.3 Image Generation and Denoising Results

Quantitative results for image generation quality and diversity and image denoising are shown in Table 1. Each model performs quite well given the training time and setup, with DDPM achieving the best FID and PSNR scores and our model achieving the best SSIM score. As seen in Figures 3 and 4, the generated and denoised images appear qualitatively on par with each other.

One important observation we can take from Figure 4 is that DDPM appears to denoise images by generating new details that weren't there before while our method produces blurrier representations without these new details. This may offer some explanation as to why our SSIM scores are slightly better and PSNR scores are slightly lower.

Another interesting observation is that DDPM and cold diffusion training updates seemed heavily biased towards a single colour palette per step while our algorithm was less biased but had much more difficulty with solid backgrounds (e.g. completely white backgrounds or plain skies). This could explain why FID scores for DDPM are the best since the model for our method may be struggling to produce images with very high frequencies in certain areas and low frequencies everywhere else, but this difference is near negligible towards the end of training.

Overall, these results allow us to infer that our algorithm seems to match the generation quality and diversity of DDPM.

4.4 2D Point Cloud Generation Results

Qualitative results for 2D point cloud generation are shown in Figures 5 to 7. Figures 9 and 10 show the paths of points through space with and without obstacles in the scene; these paths appear efficient in space, unlike those for DDPM $x_t \rightarrow x_{t-1}$ jumps shown in Figure 8. Compared to image generation and denoising results, the 2D point cloud generation results highlight more of the disadvantages of our model including the failure of linear interpolation to produce optimal straight paths for scenes with several obstacles.

5 Conclusion

In this work, we devised a new generative model inspired by DDPM and other diffusion models, but removed the first-order Markov chain to enable the reverse process and final sample to satisfy constraints specified by a differentiable physics engine. We showed this algorithm matched image generation and denoising standards set by DDPM and gave examples of our algorithm handling a constrained environment for 2D point cloud generation. We believe our algorithm is a promising formulation for handling constrained generation since it performs well on all our

TABLE 1
Generation and Denoising Results

Model	FID (generation)	PSNR (denoising)	SSIM (denoising)
DDPM	29.8	22.84	0.907
Ours	31.1	21.12	0.953

experiments, maintains the generality of sampling equilibria from any distribution [6], produces relatively efficient trajectories through space, and is portable to several different applications. Even so, our algorithm is hindered by training overhead from the physics engine and certain constraints like large immovable obstacles, but we expect constrained problems to come with these tradeoffs. Henceforth, we hope to continue testing our algorithm and scaling it to tackle image generation with geometric constraints in RGB-XY pixel space, voxelized 3D point cloud generation with non-overlapping points, and generative models for manipulating deformable meshes.

ACKNOWLEDGMENTS

The authors would like to thank Prof. David Lindell for his help and advice.

REFERENCES

[1] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *CoRR*, vol. abs/2006.11239, 2020. [Online]. Available: https://arxiv.org/abs/2006.11239

[2] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," *CoRR*, vol. abs/1503.03585, 2015. [Online]. Available: http://arxiv.org/abs/1503.03585

[3] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *CoRR*, vol. abs/2010.02502, 2020. [Online]. Available: https://arxiv.org/abs/2010.02502

[4] A. Nichol and P. Dhariwal, "Improved denoising diffusion probabilistic models," *CoRR*, vol. abs/2102.09672, 2021. [Online]. Available: https://arxiv.org/abs/2102.09672

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," *CoRR*, vol. abs/2112.10752, 2021. [Online]. Available: https://arxiv.org/abs/2112.10752

[6] A. Bansal, E. Borgnia, H.-M. Chu, J. S. Li, H. Kazemi, F. Huang, M. Goldblum, J. Geiping, and T. Goldstein, "Cold diffusion: Inverting arbitrary image transforms without noise," 2022. [Online]. Available: https://arxiv.org/abs/2208.09392

[7] 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," *The International Journal of Robotics Research*, 2024.

[8] S. Hosseini, M. A. Shabani, S. Irandoust, and Y. Furukawa, "Puzzlefusion: Unleashing the power of diffusion models for spatial puzzle solving," 2023.

[9] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *CoRR*, vol. abs/1512.00567, 2015. [Online]. Available: http://arxiv.org/abs/1512.00567

[10] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, G. Klambauer, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017. [Online]. Available: http://arxiv.org/abs/1706.08500

[11] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[12] Y. Wu and K. He, "Group normalization," *CoRR*, vol. abs/1803.08494, 2018. [Online]. Available: http://arxiv.org/abs/1803.08494

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[14] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *CoRR*, vol. abs/2006.09661, 2020. [Online]. Available: https://arxiv.org/abs/2006.09661

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

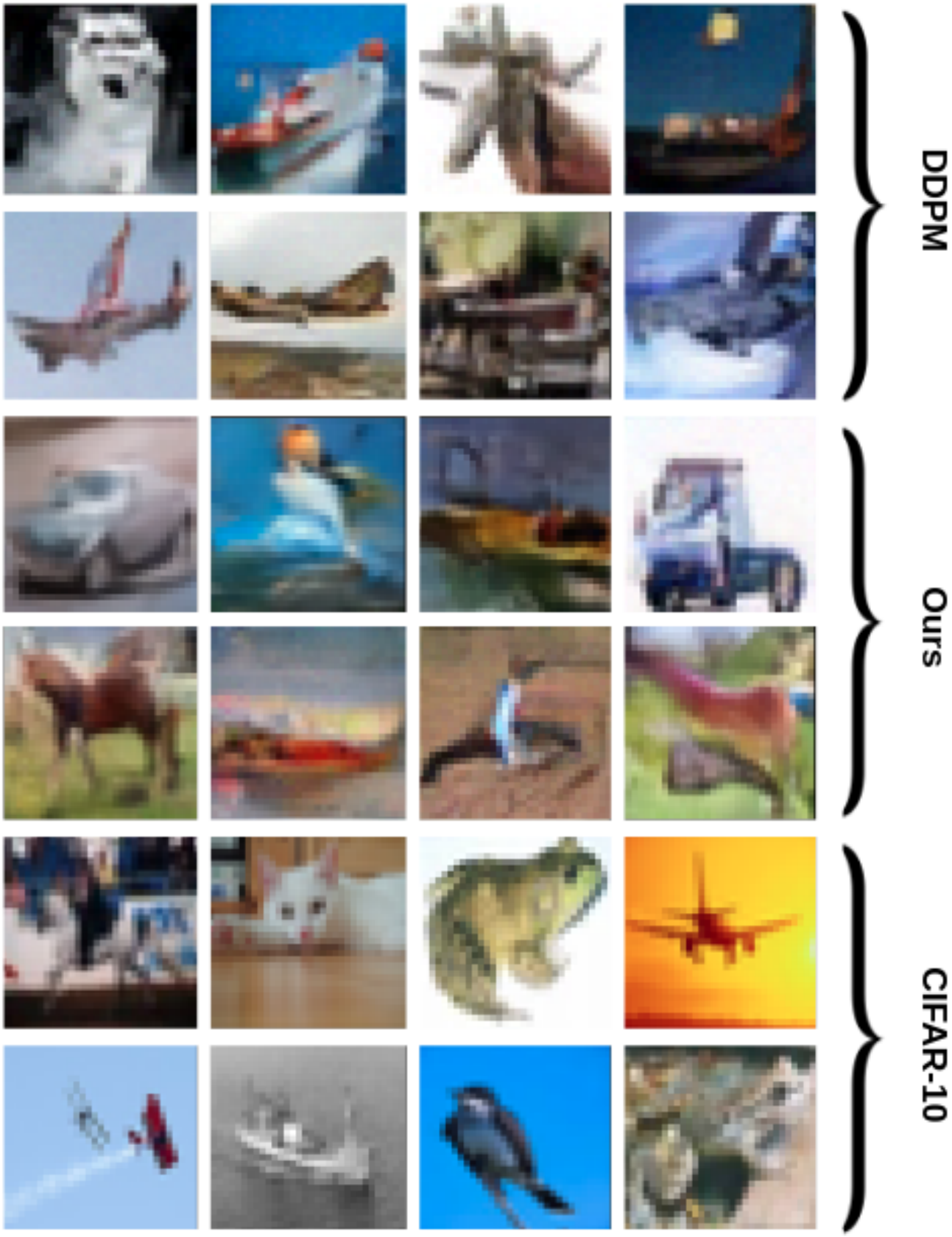

Fig. 3. Generated Images

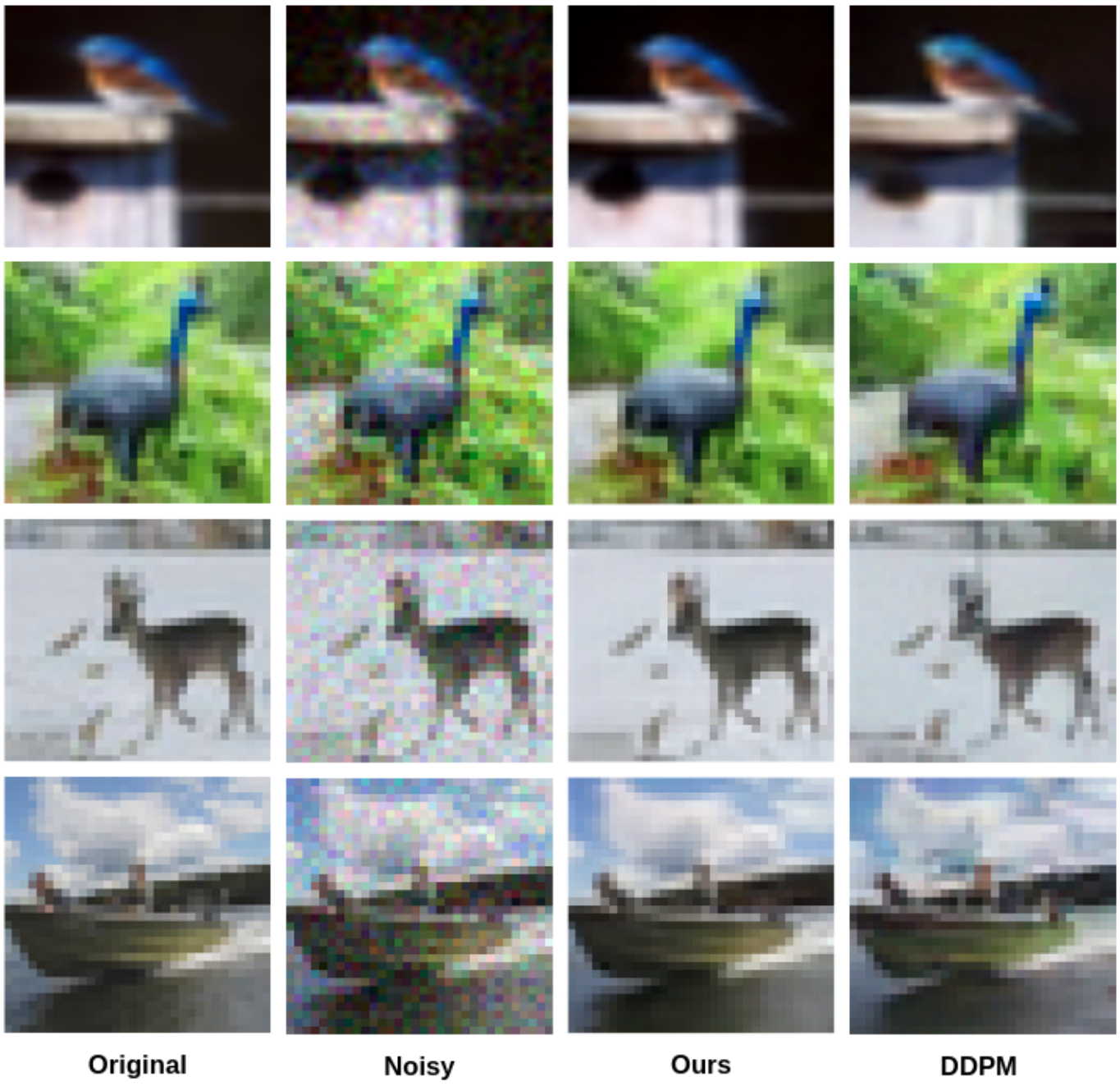

Fig. 4. Image Denoising

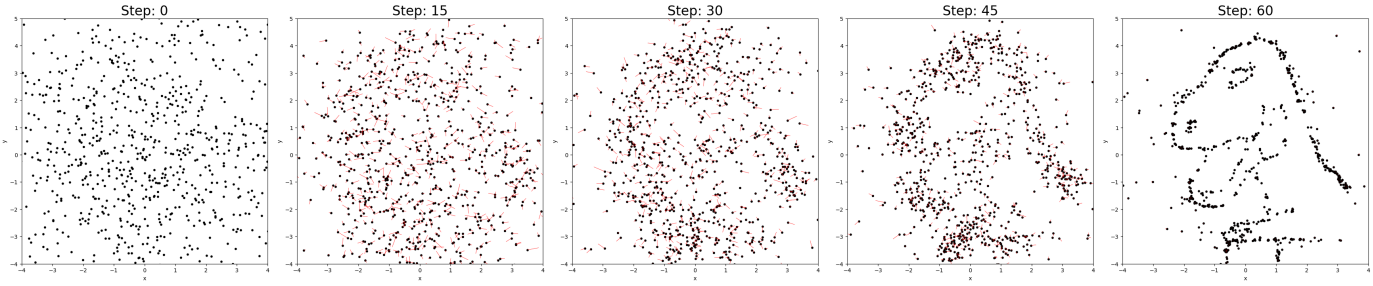

Fig. 5. DDPM 2D point cloud generation. Sizes of points are smaller in actuality.

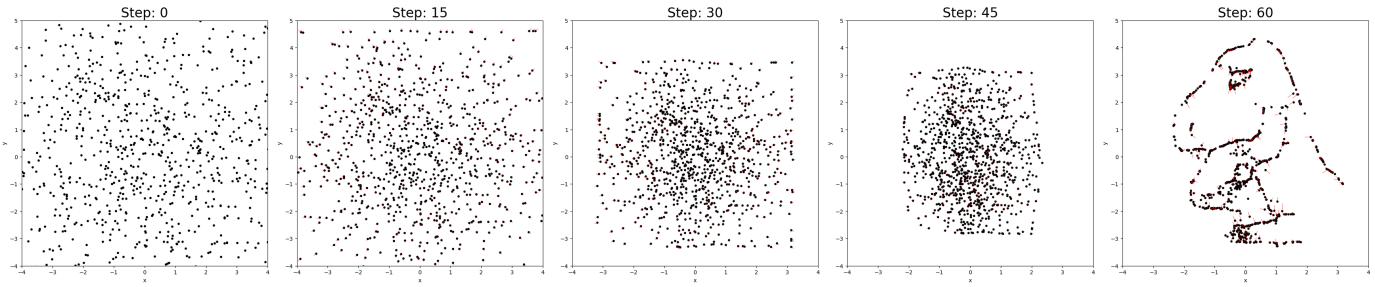

Fig. 6. Our 2D point cloud generation. Sizes of points are smaller in actuality.

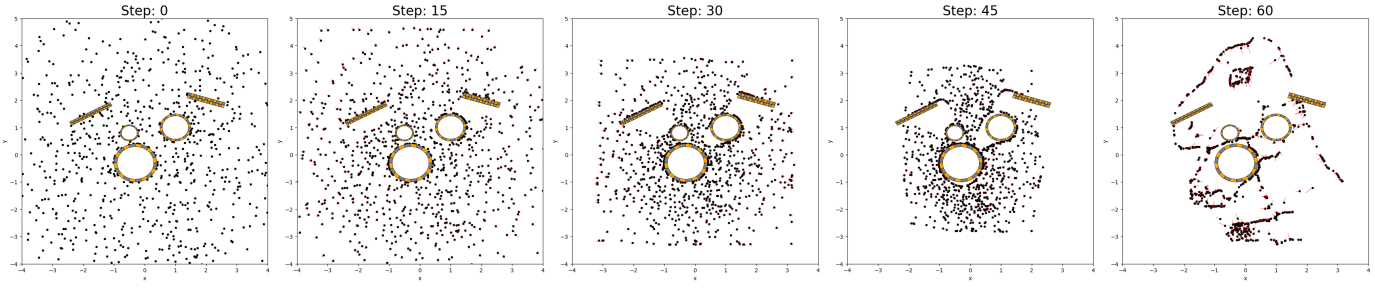

Fig. 7. Our 2D point cloud generation with obstacles. Sizes of points are smaller in actuality.

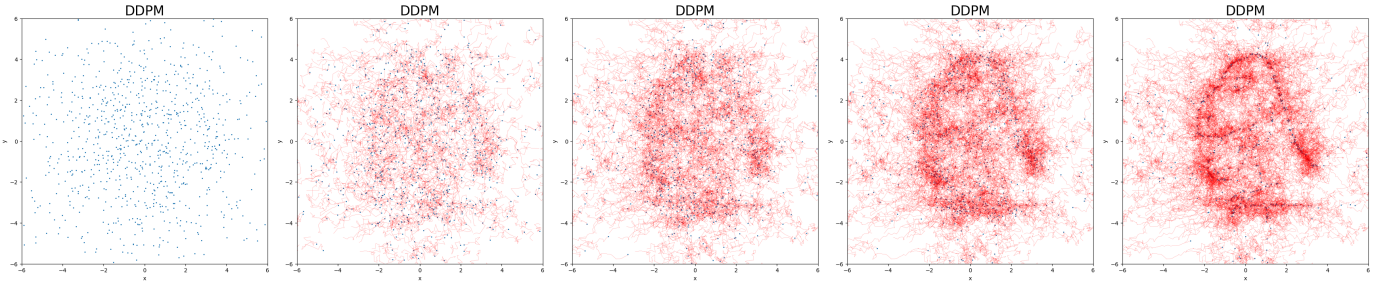

Fig. 8. DDPM 2D point cloud generation trajectories.

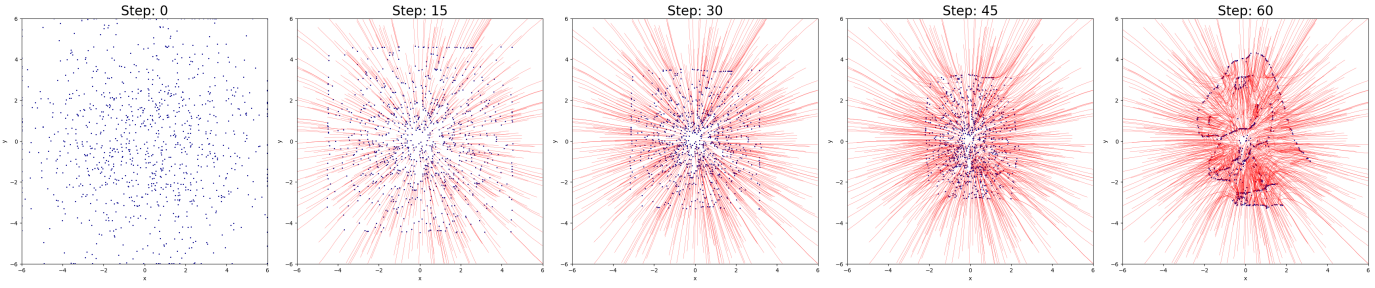

Fig. 9. Our 2D point cloud generation trajectories.

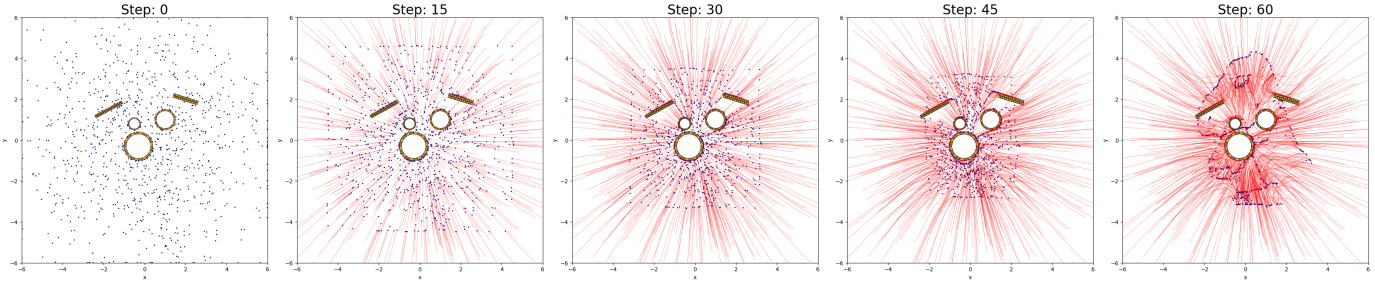

Fig. 10. Our 2D point cloud generation trajectories with obstacles.