
Latent Painter

Shih-Chieh Su
jessysu@gmail.com

Abstract

Latent diffusers revolutionized the generative AI and inspired creative art. When denoising the latent, the predicted original image at each step collectively animates the formation. However, the animation is limited by the denoising nature of the diffuser, and only renders a sharpening process. This work presents Latent Painter, which uses the latent as the canvas, and the diffuser predictions as the plan, to generate painting animation. Latent Painter also transits one generated image to another, which can happen between images from two different sets of checkpoints.

1 Introduction

Recently, denoising diffusers gain a lot of traction in generative AI, for its high quality outcome without adversarial training [1], its efficiency [2, 3], content diversity with easy text conditioning [4], and reasonable footprint [5]. Although the convenience of text-to-image largely spurs the creativity, little has been studied about the composition of its generated art. This work presents Latent Painter, which uses the existing diffuser to generate painting animation along with the output image.

2 Method

During the diffusion denoising, the latent is denoised step-by-step into the state representing an image matching the text input. The predicted original image, which is the progressive estimate \hat{x}_0 of the reverse diffusion process in [1], becomes sharper and sharper when the latent being denoised. Collecting the predicted original images forms an animation about the sharpening process, where information is updated omnipresently in the same frame. However, the update is uneven across frames, with higher total pixel value change toward earlier frames. Latent painter prioritizes the update locally to just like the brush strokes. Once the released information is close enough to match the current predicted original, the residue is accumulated for later updates. This mechanism provides update more evenly over frames.

3 Result

Sample outputs from the Latent Painter Strokes (Alg. 1, see Appendix A) are shown in Fig. 1. The samples were first generated with stable diffusion [5] using a text sentence. The incurred latent series of the predicted original images is fed into the painter to produce the strokes. From the bottom rows of each block in Fig. 1, the denoising outputs quickly converge close to the final state in the first couple iterations, each of which provides only one frame in the animation.

In contrast, the Latent Painter slows down the rapid update during early denoising iterations. This prevents frames being updated too quickly, and helps rendering the new information more evenly over frames. Here, the new information is released in the form of strokes. Each denoising iteration can be released in tens to hundreds of strokes (frames). The stroke maps in the middle rows indicate the regions with large information gap between the latest \hat{x}_0 update and current state, thus being stroked.

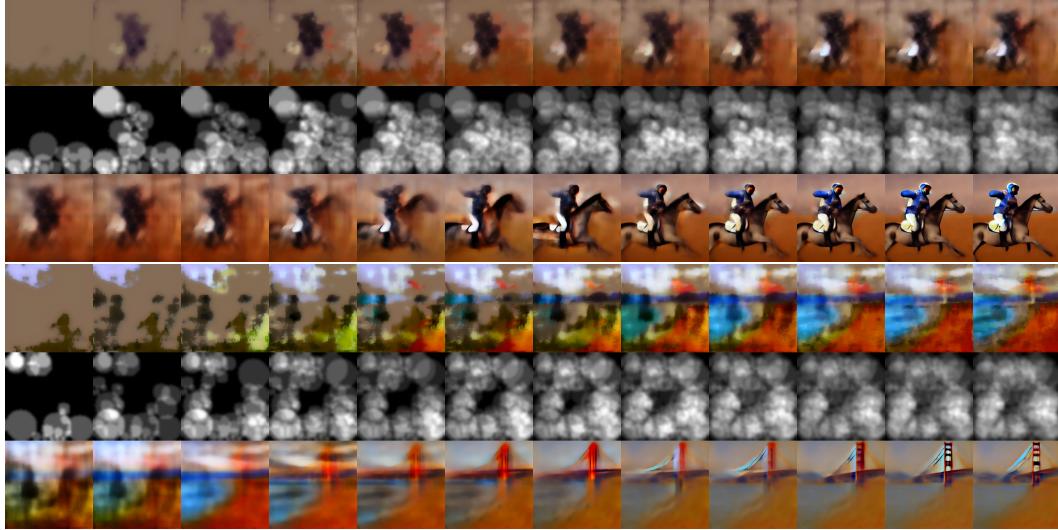


Figure 1: Stroke location of Latent Painter. Within each block, the bottom row shows the predicted original images \hat{x}_0 after each of the first 12 denoising iterations, each with only one frame. The top row shows the Latent Painter stroke snapshots, each containing 20-40 strokes (frames), during the same period. The middle row are the corresponding accumulated stroke map, stacked over channels. The animations are available at <https://latentpainter.github.io/>

4 Content Analysis

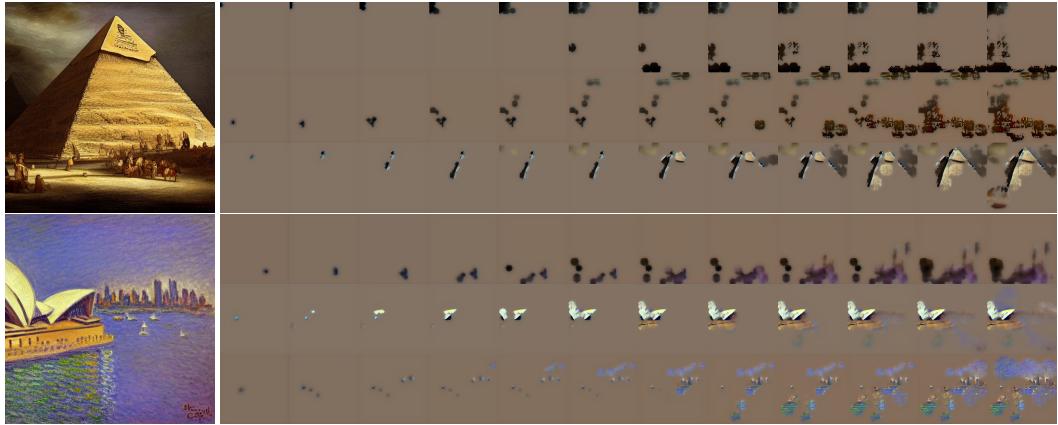


Figure 2: Stroke content of Latent Painter. Within each block, the rows show the cumulative stroke output of channel 0, when it was chosen for the first three times as the target channel to stroke.

Some stroke samples from the Latent Painter are shown in Fig. 2. The snapshots each row only accumulates the strokes from the same channel update, which starts from Eq. 2. Within the same channel, the strokes tend to be congruent in color or style, or both. The channels of the latent space are different from those at the visible layer, where the RGB channels each only account for a color.

5 Conclusion

Latent Painter is presented in this work. It turns existing latent diffuser output into painting actions via evenly releasing the information update during the denoising iterations. Several extension beyond the stroke algorithm has been covered. Since the latent space of the stable diffusion [5] has only four channels, it is possible to investigate the VAE for further painting behaviors.

6 Ethical Implications

With the depth in animation planning, the Latent Painter currently bears a unique signature to self-categorize as a type of generated art. Self regulation is advised. The user should not submit the outcome to anywhere that prohibits generated art.

References

- [1] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.
- [2] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [3] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, 35:26565–26577, 2022.
- [4] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- [5] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.
- [6] Manuel Brack, Felix Friedrich, Dominik Hintersdorf, Lukas Struppek, Patrick Schramowski, and Kristian Kersting. SegA: Instructing diffusion using semantic dimensions, 2023.

Appendix A Stroke Algorithm

Let $Z(x, y, c)$ denote the state of the latent, having shape of width by height by channels, and can be initiated as zeros. During each inference step t , the scheduler provides an updated denoised latent sample D_t . The policy \mathcal{P} decides whether the information difference between Z and D_t qualifies an update and if so, which subset (\bar{C}) of all latent channels C needs to be updated.

$$\bar{C} = \mathcal{P}(Z, D_t), \text{ where } \bar{C} \in C \quad (1)$$

During a channel update, a channel c is chosen out of \bar{C} . Then, a qualifying threshold θ picks the region R to be stroked on.

$$R = \{(x, y) : |Z(x, y, c) - D_t(x, y, c)| > \theta\} \quad (2)$$

Let $G_c(x, y)$ denote the information gain at the current channel c .

$$G_c(x, y) = |Z_c(x, y) - D_{t,c}(x, y)| \quad (3)$$

The first stroke is placed at the location whose neighborhood $\mathcal{N}(x, y)$ has the largest information gain.

$$p(\hat{x}, \hat{y}) = \operatorname{argmax}_{(x, y)} \sum_{(x', y') \in \mathcal{N}(x, y)} G_c(x', y') \quad (4)$$

The whole neighborhood $\mathcal{N}(x, y)$, presented as the stroke of the Latent Painter, is then updated with the current scheduler output.

$$Z(x, y, c) = D_t(x, y, c), \text{ for all } (x, y) \in \mathcal{N}(\hat{x}, \hat{y}) \quad (5)$$

The stroke-able region in Eq. 2 is then updated to exclude the newly stroke region. Following the same procedure, the strokes are placed one by one until R being empty, or an early stopping criteria \mathcal{E} has been reached. Continue the stroke action in other channels in \bar{C} likewise.

Upon finishing all \bar{C} channels, the painter steps through another iteration t of the scheduler to get a new D_t , then starts from Eq. 1 to get the channels to be painted this iteration. However, there could be very little difference between the new D_t and current Z . This is particularly true toward the end of the denoising process.

To ensure the animation frames having meaningful update, one idea is to release D_t more evenly across strokes, each being presented in one frame. Therefore, \mathcal{P} requires the total difference between Z and D_t being larger than a portion of the largest total difference over all previous iterations. With D_t sufficiently diverse from Z , \mathcal{P} then provides the channels needing update and to be painted, as in Eq. 1.

Note the current scheduler output D_t may not fully pass onto the latent Z at time t , when the early stopping condition \mathcal{E} exists. The un-updated residue is carried over to next scheduler output D_{t+1} . The accumulated residue becomes the momentum for the next strokes.

Algorithm 1 Latent Painter - Strokes

Given policy \mathcal{P} , stroke qualifier θ , stroke neighborhood \mathcal{N}

```

Initialize painter state  $Z$  as zeros of the latent shape  $w \times h \times |C|$ 
for  $t$  in diffuser schedule do
    Compute  $D_t$ , the diffuser latent space prediction of  $x_0$  at current time  $t$ 
    Based on policy  $\mathcal{P}$ , decides the channels to be updated  $\bar{C} = \mathcal{P}(Z, D_t)$ , where  $\bar{C} \in C$ 
    for  $c \in \bar{C}$  do
        Compute the stroke region  $R = \{(x, y) : |Z(x, y, c) - D_t(x, y, c)| > \theta\}$ 
        Initialize move cost  $M(x, y)$  as  $w \times h$  of ones
        while  $R$  not empty, as step  $s$ , do
            Compute gain  $G_c(x, y) = |Z_c(x, y) - D_{t,c}(x, y)|$  at channel  $c$ 
            Compute motivation  $V(x, y) = G_c(x, y) \cdot M_c(x, y)$ 
            Pick the stroke point  $p(\hat{x}, \hat{y}) = \operatorname{argmax}_{(x, y)} \sum_{(x', y') \in \mathcal{N}(x, y)} V(x', y')$ 
            Stroke to make  $Z(x, y, c) = D_t(x, y, c)$  for all  $(x, y) \in \mathcal{N}(\hat{x}, \hat{y})$ 
            Update  $R \leftarrow R \setminus \{(x, y) : (x, y) \in \mathcal{N}(\hat{x}, \hat{y})\}$ 
            Compute move cost  $M(x, y)$  as an inverse Gaussian filter centered  $(\hat{x}, \hat{y})$ 
        end while
    end for
end for

```

Appendix B Move Cost

The human painter typically considers optimizing the effort in painting, such as to stroke the nearby area first, and keep using the same brush and the same color as much as possible, before changing or cleaning the brush. On the machine side, while the convolutional layers are trained to capture congruent patterns, placing the strokes within the same channel allows more congruent patterns being stroked.

The moving cost of the brush, denoted by $M_c(x, y)$, is modeled as the inverse of a Gaussian kernel centered at the current stroke location. Rather than the location with largest information gain, the stroke is placed where having the largest motivation V , which is defined as the information gain modulated by the moving cost,

$$V(x, y) = G_c(x, y) \cdot M_c(x, y) \quad (6)$$

The motivation-based stroke method is presented in Alg. 1.

Appendix C Beyond Strokes

Besides the painting action in Alg. 1, there are other ways to leverage the differential response between updates. The glow effect collects updates of the differential latent into the mass center, where the update radiates concentrically.

In addition to the content-driven fashion, Latent Painter can also paint regardless of the differential latent response. Some use cases including the flip effect that mimics page flipping and the fade effect that release the update uniformly. The dissolve effect, however, can be either content-driven or random. Some sample trails of the mentioned effects are visualized in Fig. 3 in the form of update heatmap.

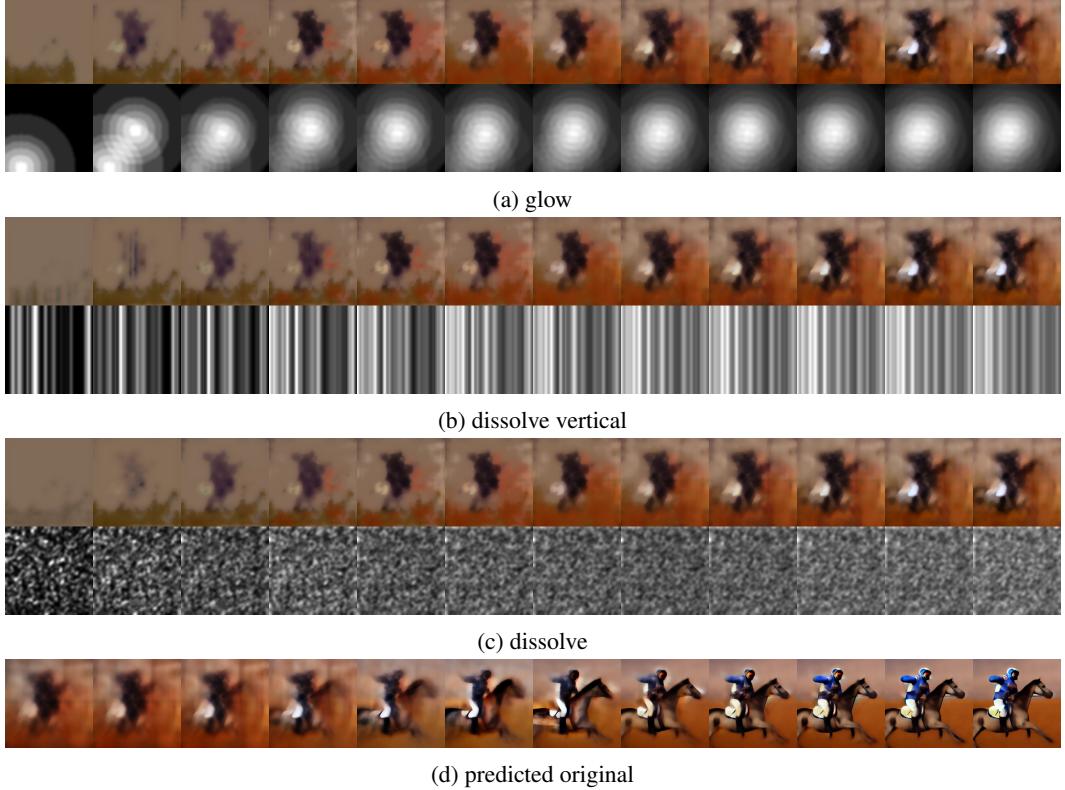


Figure 3: Extensions from strokes. The glow effect is content-driven, while the dissolve effect here isn’t. See <https://latentpainter.github.io/> for animations and more examples.

Appendix D Image Transition

Existing image transiting animation is based on interpolating the seeded embeddings between the source and the destination, or interpolating the latents, or both. However, it takes either more memory or more time to denoise the interpolated embeddings. Latent Painter animates based on the predicted original images from only two denoising trails, the source and the destination. With the chosen painting effect, the update runs the source image denoising schedule backward, and then the destination schedule forward. Through the constraints of the update release, the transition time can be traded with the detail.

When the source and destination images share a certain part of background, such as in image editing, the interpolated latents between the source latent and the destination latent can be used as the prediction guidance. This avoids using the denoising schedules to guide, while it still requires the denoised latents. Fig 4 illustrates the painting progress to transit the semantically edited images from [6].

As an extra credit, Latent Painter can transit the generated images from two different sets of denoising checkpoints, given the same VAE is used to decode the latent.

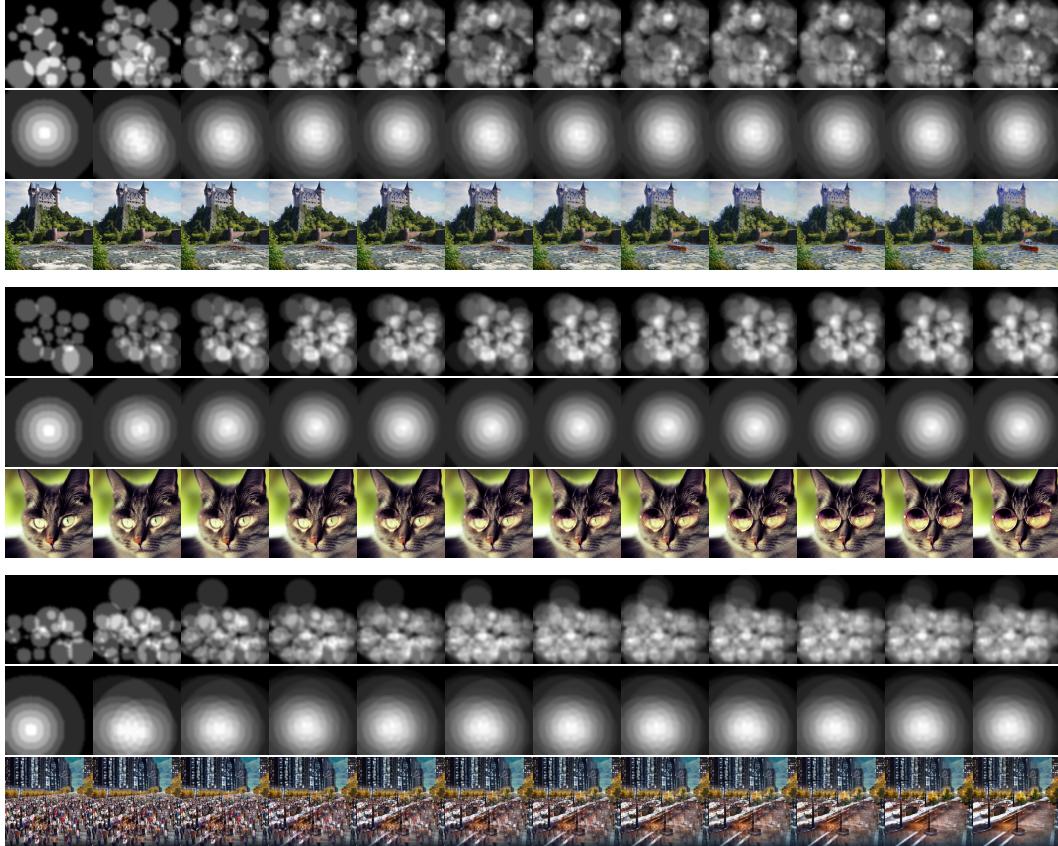


Figure 4: Image to image transiting animation guided by the interpolated latents from [6]. Within each block, the top row presents the stroke heatmap transiting one image to its semantically edited variation, while the middle row presents the update heatmap of glow effect, both cumulative. The bottom row shows the output status at the corresponding time points.