Inverse Painting: Reconstructing The Painting Process

BOWEI CHEN, University of Washington, USA
YIFAN WANG, University of Washington, USA
BRIAN CURLESS, University of Washington, USA
IRA KEMELMACHER-SHLIZERMAN, University of Washington, USA
STEVEN M. SEITZ, University of Washington, USA

Input

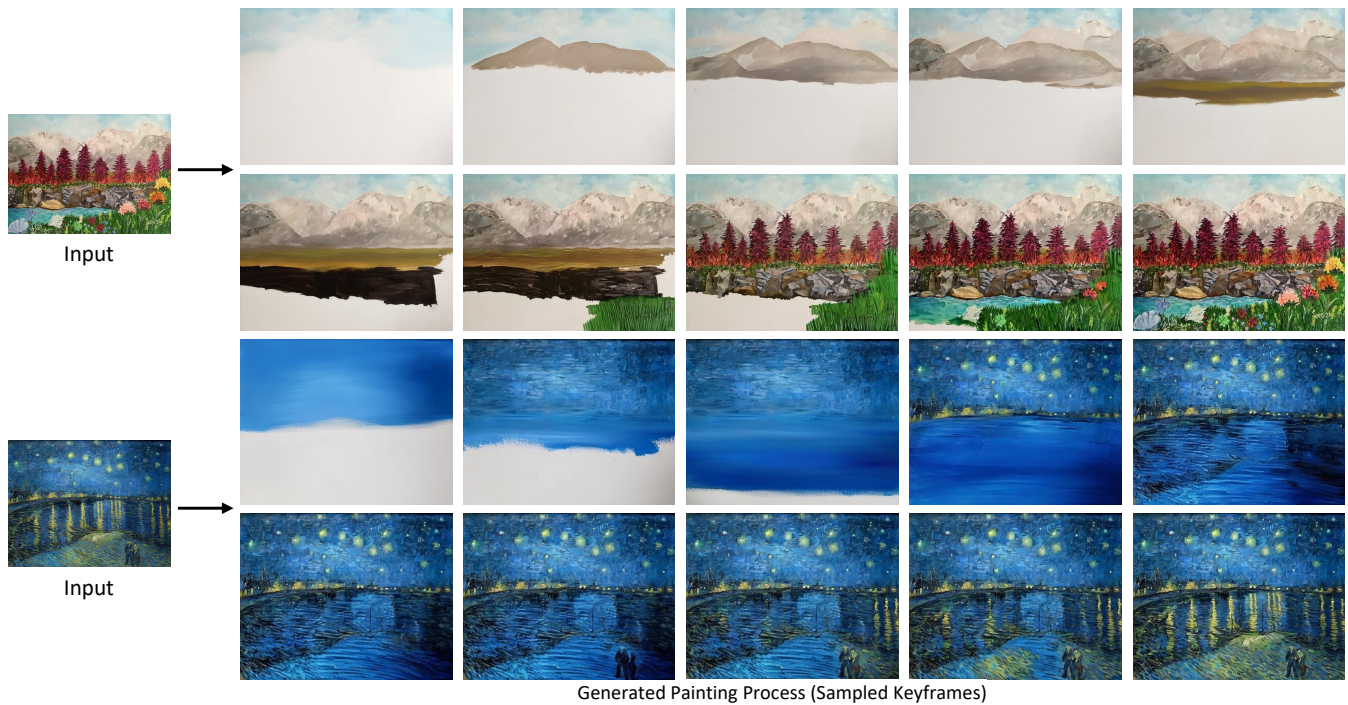Input

Generated Painting Process (Sampled Keyframes)

Fig. 1. We present Inverse Painting, a diffusion-based method to generate time-lapse videos of the painting process from a target painting. This figure shows 10 keyframes from the generated painting process for two paintings. By training on acrylic paintings with a similar artistic style to that of the first example in this figure, our method is capable of handling a diverse range of styles (e.g., Van Gogh, above bottom). The resulting videos resemble how human artists typically paint, for example, from back to front, focusing on semantic objects or regions at a time, and employing layering techniques. Images courtesy Catherine Kay Greenup and Rawpixel.

Given an input painting, we reconstruct a time-lapse video of how it may have been painted. We formulate this as an autoregressive image generation problem, in which an initially blank "canvas" is iteratively updated. The model learns from real artists by training on many painting videos. Our approach incorporates text and region understanding to define a set of painting "instructions" and updates the canvas with a novel diffusion-based renderer. The method extrapolates beyond the limited, acrylic style paintings on which it has been trained, showing plausible results for a wide range of artistic styles and genres. Our project page and code are available at: https://inversepainting.github.io/

Authors' addresses: Bowei Chen, University of Washington, 1410 NE Campus Pkwy, Seattle, WA, 98195, USA, boweiche@cs.washington.edu; Yifan Wang, University of Washington, Seattle, USA, yifan1@cs.washington.edu; Brian Curless, University of Washington, Seattle, USA, curless@cs.washington.edu; Ira Kemelmacher-Shlizerman, University of Washington, Seattle, USA, kemelmi@cs.washington.edu; Steven M. Seitz, University of Washington, Seattle, USA, seitz@cs.washington.edu.

Additional Key Words and Phrases: Painting Process Reconstruction, Inverse Painting, Diffusion Models

1 INTRODUCTION

When we look at a painting, we see only the final outcome of the artist's creative process. Leonardo da Vinci worked on the Mona Lisa for 16 years – it would be fascinating to see a time-lapse video of the Mona Lisa's creation. While no such video exists for Leonardo, there are many videos online in which artists have filmed the creation of entire paintings. These visualizations are fascinating in the way they show hidden layers, structures, and provide insight into the artistic creation process. While such visualizations currently exist for only a tiny subset of paintings in the world, we propose to train machine learning models on such data to predict how a wide variety of other paintings could have been made. While the resulting videos should not be treated as accurate reconstructions of any specific painting's

Fig. 2. **How a real artist paints.** Time lapse from a real painting video, representative of the training painting style. The artist uses a back-to-front order with layering techniques, starting with the sky, then clouds, mountains, and other elements. The artist typically focuses on one semantic region at a time.

creation, they are nonetheless intriguing and insightful as *plausible* reconstructions, in capturing rules that many painters employ, such as layering, back-to-front-ordering, and focusing on objects/regions in stages [Bob 1987; Dozier 2007]. While our approach was trained only on acrylic paintings of landscapes, as shown in Fig. 2, we believe that future models could produce plausible visualizations for almost any piece of art.

Previous approaches [de Guevara et al. 2023; Ganin et al. 2018; Hu et al. 2023b; Huang et al. 2019; Liu et al. 2021; Singh et al. 2021; Singh and Zheng 2021; Zou et al. 2021] rely on hand-crafted painting principles instead of learning them from the real painting processes. The most relevant work, Timecraft [Zhao et al. 2020], generates time-lapse videos by learning from actual painting videos. However, it operates only on low-resolution (50x50 pixels) patches and therefore lacks holistic semantic context.

We define this task as an autoregressive image generation problem. It begins with a blank canvas, which is iteratively updated based on its current state and the target painting. This sequential updating continues until the artwork is completed. To implement this, one possible approach, similar to Timecraft [Zhao et al. 2020], is to train a network that takes the current and target images as inputs and outputs the updated canvas at each step. However, we've found that a purely pixel-based approach struggles to produce reasonable results in practice, and thus we incorporate additional semantic analysis.

In particular, we draw inspiration from the techniques used by human artists. Consider the second painting in Fig. 1. Initially, an artist decides on the semantic content to depict – such as the sky – and selects appropriate areas of the canvas for this content, such as the upper portion. Subsequently, the artist applies specific paints to these regions, using blue for the sky, while leaving contents such as stars for later stages. Building on this observation, we design a two-stage method that decomposes the instruction generation and canvas rendering. In the first stage, we generate a textual instruction and a corresponding region mask from the current and target images. The textual instruction provides high-level guidance on the order of painting, and its corresponding region mask directs the focus area. In the second stage, a diffusion-based renderer is proposed to leverage the textual instruction and region mask, in conjunction with the current and target images, to effectively update the canvas.

Our method can handle in-the-wild landscape paintings across diverse artistic styles (*e.g.*, Impressionism and Realism) and color themes, ranging from dark to bright. We demonstrate that our approach surpasses current state-of-the-art methods in creating high-quality, human-like painting videos, supported by both qualitative and quantitative results, as well as human studies.

2 RELATED WORK

2.1 Painting Process Generation

Stroke-Based Rendering. This is a computer graphics technique that creates non-photorealistic images by placing discrete elements like brush strokes instead of traditional pixels [Frans and Cheng 2018; Fu et al. 2011; Ganin et al. 2018; Ha and Eck 2017; Haeberli 1990; Hertzmann 1998; Jia et al. 2019; Litwinowicz 1997; Liu et al. 2023b; Tang et al. 2017; Xie et al. 2013]. The painting process can be obtained by visualizing the sequence of element placement. However, many studies mainly focus on generating images in various artistic styles [Kotovenko et al. 2021; Litwinowicz 1997; Liu et al. 2023b; Xie et al. 2013; Zou et al. 2021]. They often overlook the order and position of brush stroke placement, resulting in a non-human-like painting process. To produce a more human-like painting process, recent work constrains the placement of brush strokes based on predefined painting principles using techniques such as reinforcement learning (RL) [de Guevara et al. 2023; Ganin et al. 2018; Hu et al. 2023b; Huang et al. 2019; Singh et al. 2021; Singh and Zheng 2021; Zou et al. 2021] or Transformers [Liu et al. 2021]. For example, [Liu et al. 2021] developed a Transformer-based [Vaswani et al. 2017] feed-forward painter that applies a coarse-to-fine painting strategy. Initially, the painter works from a blurry (downsampled) version of the target painting, progressively refining the canvas using higher-resolution versions until the painting is complete.

However, these techniques face two limitations: (1) They rely on hand-crafted painting principles that do not accurately mimic the actual human painting process, whereas our method learns from real-world painting data. (2) Their parameterized brush strokes fail to capture the full variation observed in real painting processes and result in an approximate version of the target painting. In contrast, our diffusion-based renderer effectively handles these variations and recreates the target painting more accurately.

Pixel-Based Generation. This stream of work generates the painting process by directly updating pixels on the canvas [Leiser and Schlippe 2021; Wang et al. 2024; Zhao et al. 2020]. [Wang et al. 2024] used a Vision Transformer (ViT) [Dosovitskiy et al. 2020] to map a target image to a series of 9 intermediate images via a non-autoregressive approach. However, this method is specifically tailored for traditional Chinese painting. The work most related to ours is Timecraft [Zhao et al. 2020], which generates time-lapse videos from paintings by learning from actual painting videos. However, their method is limited to low-resolution (50x50 pixels) crops, neglecting the full artwork's context. Our method handles full paintings with higher resolution.

2.2 Diffusion Models

Diffusion models have recently demonstrated their success in various tasks such as text-to-image [Rassin et al. 2022; Rombach et al. 2022; Saharia et al. 2022], image-to-image translation [Chen et al. 2023; Yang et al. 2022], and image-to-video synthesis [Blattmann et al. 2023; Hu et al. 2023a]. [Blattmann et al. 2023] developed a latent video diffusion model capable of generating videos from an initial frame. [Hu et al. 2023a] proposed a diffusion framework to synthesize character animation videos from a reference image and a sequence of target poses. This framework includes a component called ReferenceNet to inject features from the reference image directly into the denoising UNet of the diffusion model. In this paper, we adopt the ReferenceNet to inject the target image into our diffusion-based renderer.

3 OUR METHOD

In a real painting process, an artist continually applies changes to the current state of the canvas. Our method formulates this process as an autoregressive image generation problem. Given a target painting I_T as input, we reconstruct a time-lapse video consisting of T keyframes $\{\hat{I}_1, ..., \hat{I}_T\}$, starting from a blank canvas I_0 progressively towards the target I_T. Each *keyframe* transition represents a fixed time interval, a feature of time-lapse videos. We start by presenting a one-step canvas rendering approach for each canvas update and discussing its limitations in Sec. 3.1. This motivates our two-stage design, which incorporates additional semantic analysis. The details of the training process are described in Sec. 3.2 and 3.3, while the testing is covered in Sec. 3.4. Fig. 3 shows an overview of the method.

3.1 One-Stage Canvas Rendering Approach

For clarity, we will refer to step $t-1$ as the current step and step t as the next step. The goal is to render next image $\hat{I}_t$ based on $\hat{I}_{t-1}$ and I_T. We approach this as an image translation problem, and design a diffusion-based renderer based on the denoising UNet g_u from Stable Diffusion [Rombach et al. 2022]. This renderer leverages image priors to enhance image quality by initializing g_u with pretrained weights from Stable Diffusion.

We train this diffusion-based renderer with inputs $\{I_{t-1}, I_T, \Delta_t\}$, where I_{t-1} is the ground-truth (GT) current image, and Δ_t is the actual time interval between I_{t-1} and GT next image I_t. By incorporating Δ_t, we model the time spent on each update in the painting process, enabling the generation of videos with time-informed keyframes during testing. The output $\hat{I}_t$, which predicts the next image, is supervised using I_t. For supervision, we start with a clean latent $z_0 = E_I(I_t)$, where E_I is the pretrained VAE encoder in Stable Diffusion. We then apply the forward process to produce a noisy latent $z_s = \beta_s z_0 + (1 - \beta_s)\epsilon$, where s denotes a randomly sampled denoising timestep, ϵ represents Gaussian noise, and β_s is a weighting parameter dependent on s. The denoising UNet g_u is then employed to denoise z_s. We update the parameters of the renderer by minimizing the following loss function:

$$\mathcal{L} = \mathbb{E}_{s,z_0,\epsilon} ||g_u(z_s, c, s) - \epsilon||_2^2, \tag{1}$$

where c represents the conditional signals of g_u, consisting of features of $\{I_T, I_{t-1}, \Delta_t\}$. We extract these features using **ReferenceNet**

g_r, **feature encoder** g_f, **time encoder** g_t and **next CLIP generator** g_c. During the training, we jointly update g_u, g_r, g_f, g_t, and g_c using Eq.1. Please refer to the gray box "Training: Canvas Rendering" in Fig. 3 (excluding the mask and text in the purple box).

ReferenceNet g_r. The target image I_T is integrated into g_u through g_r, as introduced in [Hu et al. 2023a]. The ReferenceNet g_r utilizes the same UNet architecture and pretrained weights (for initialization) from Stable Diffusion [Rombach et al. 2022], ensuring feature extraction within the same feature space as g_u. It takes the target image I_T as input, and extracts intermediate feature maps from its self-attention layers. These feature maps are then fused into g_u by concatenating them with corresponding feature maps from the same layers in g_u. We opt for the design of ReferenceNet because it fuses features more effectively than other designs, such as ControlNet [Zhang et al. 2023], in practice.

Feature encoder g_f. It consists of a shallow convolutional neural network (CNN) to encode I_{t-1}, denoted as $g_f(I_{t-1})$. We found that this shallow network effectively learns features of I_{t-1} while saving computational time. The encoded feature map has the same spatial resolution of z_s (noisy latent of I_t), and is concatenated with z_s along channel dimension as spatial input to the g_u during training. The first layer of g_u is modified to adjust to the new channel dimension.

Time encoder g_t **and next CLIP generator** g_c. First, we design a time encoder g_t that applies positional encoding – same as that used in NeRF [Mildenhall et al. 2021] – to the Δ_t, followed by a multi-layer perceptron (MLP) that maps the positional encoding feature dimension from 21 to 768. This results in the time embeddings $g_t(\Delta_t)$. The advantage of positional encoding lies in its ability to adapt to varying Δt across different training samples. This continuous encoding enhances interpolation capabilities, useful for handling specific time intervals during testing. Moreover, to provide additional painting guidance for g_u, we introduce the next CLIP generator g_c, implemented as an MLP. This module inputs the CLIP embeddings of I_{t-1} and I_T, and outputs the predicted CLIP feature of the next image. This predicted CLIP feature and $g_t(\Delta_t)$ are concatenated and fed into the cross-attention layers of g_u.

Fig. 5 (b) illustrates the results of using this model without incorporating Δ_t. In this scenario, significant content is added in a single update, which does not effectively represent the progressive nature of the painting process. When Δ_t is included through g_t (Fig. 5 (c)), the new content volume per update is better controlled; however, this approach still results in unnatural rendering of mountains, as the yellow layers prematurely appear before the completion of the green mountain base. This indicates that a purely pixel-based network struggles to fully capture the semantics of the painting, prompting the need for more explicit semantic controls.

3.2 Training: Instruction Generation

To address the aforementioned issues of the purely pixel-based method, we draw inspiration from typical artistic practices, where artists decide what to paint and where, then apply paint accordingly. Based on this observation, we propose a two-stage method (Fig. 3) where we first generate text and mask instructions as semantic guidance. These instructions are then used to steer the diffusion-based renderer. In this section, we describe the training of a text
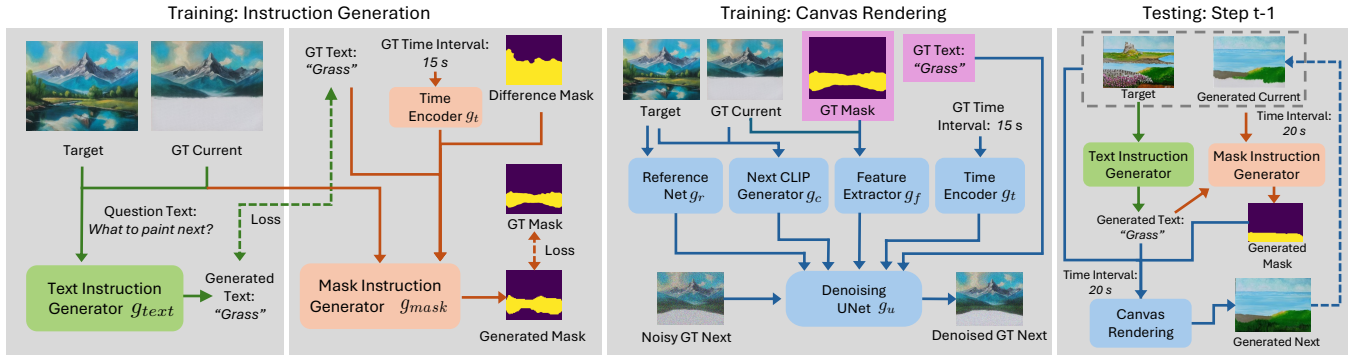
Fig. 3. **Method overview.** The training has two stages. The instruction generation stage (left two gray boxes) includes the text instruction generator (green) and the mask instruction generator (light orange). These generators produce the text and mask instructions essential for updating the canvas in the next stage. The second stage is canvas rendering (third gray box), where a diffusion-based renderer generates the next image based on multiple conditional signals, such as text and mask instructions. Omitting the text and mask (purple boxes) yields the one-stage method described in Sec. 3.1. To simplify the figure, we omit the VAE encoder, CLIP encoder, and text encoder. During testing at step $t-1$ (last gray box), we first generate a text instruction (green arrows), which is then used to create a region mask (orange arrows). Both are then provided to the canvas rendering stage to produce the next image (blue arrows). Image courtesy Catherine Kay Greenup.

and mask generator. These two generators are used to produce two types of instructions for each canvas update: a text instruction $\hat{p}_t$, describes what semantic content to paint, and a region mask $\hat{M}_t$, specifies the focus regions corresponding to the text instruction.

3.2.1 Text Instruction Generator. Like an artist who envisions a target image and compares it to the current state of the work to decide what to paint next, our text instruction generator must discern visual content and generate appropriate textual instructions. This provides high-level guidance for the painting order. We implement this generator using the architecture of a large vision-language model LLaVA [Liu et al. 2023a]. LLaVA accepts a single image combined with a question text prompt and produces a response text prompt. For training, as the generator requires a single image input, we horizontally concatenate the target image, I_T, and the ground-truth current image, I_{t-1}, to form the input. The text instruction $\hat{p}_t$ is generated as follows:

$$\hat{p}_t = g_{text}([I_T, I_{t-1}], p), \tag{2}$$

where g_{text} is the generator, and $[\cdot, \cdot]$ is horizontal concatenation of two images. p is the question text prompt (details in supplementary).

To enhance the model's performance and reduce training time, we initialize the text generator, g_{text} with pretrained weights of LLaVA 1.5 [Liu et al. 2023a]. The text generator is fine-tuned with full supervision of the ground-truth text instructions p_t. The leftmost gray box in Fig. 3 visualizes this process, where "grass" is generated as the text instruction for the next step.

Fig. 4 shows the text instructions generated during the painting process of in-the-wild paintings at test time. These instructions guide our renderer to use layering techniques, painting from back to front. This demonstrates that g_{text} not only captures the semantic essence of the target paintings but also effectively learns the painting order from the dataset.

3.2.2 Mask Instruction Generator. On top of what to paint, an artist also decides where on the canvas to apply the content. Our method
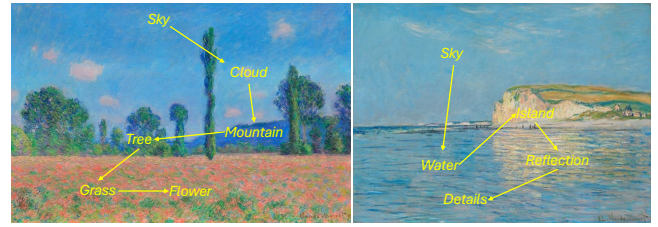

Fig. 4. **Generated text instructions.** The sequence of generated text instructions (yellow text and arrows) demonstrate a natural painting order, arranging elements from back to front such as clouds over the sky, flowers over grass, and reflections over water. The "Details" in the right image refers to water texture and small details on the island. Each text instruction may repeat over multiple frames but is displayed only once to simplify this figure. Images courtesy the Art Institute of Chicago and Cleveland Museum of Art.

formulates this as a binary region mask, $\hat{M}_t$. This mask provides instructions that specify focus regions for each step of the painting process. For training, as visualized in the second gray box in Fig. 3, our mask instruction generator considers 4 factors to determine the intended painting region: (1) The GT current image I_{t-1} and target image I_T. (2) Text instruction p_t. During training, we use the ground-truth instruction p_t to ensure that the training is guided by accurate instructional data. (3) Difference mask M_d, which represents areas of the current canvas that are still incomplete relative to the target image. This binary mask M_d is derived by computing the difference map between I_T and I_{t-1} using perceptual distance [Zhang et al. 2018]. This difference map is then binarized using a threshold $\alpha = 0.2$, setting pixels with values above α to 1 in M_d. Formally, we define the operations of computing M_d as $D(I_T, I_{t-1}, \alpha)$. (4) Time interval Δ_t, which can influence the size of the intended painting regions, as less area may be covered when less time is available.

Considering all these factors, we design our mask instruction generator based on the UNet proposed in Stable Diffusion (but

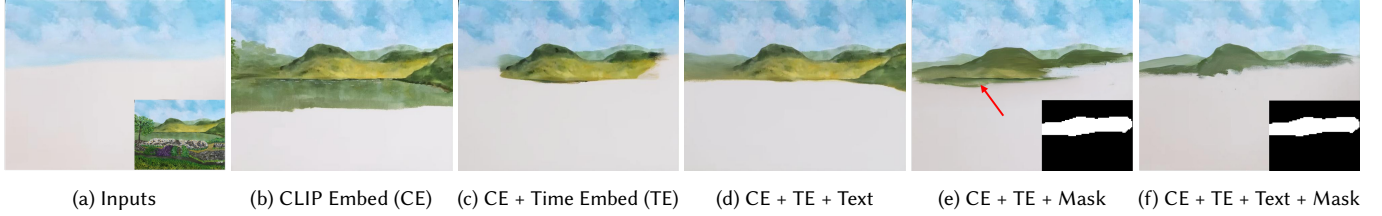|					
(a) Inputs	(b) CLIP Embed (CE)	(c) CE + Time Embed (TE)	(d) CE + TE + Text	(e) CE + TE + Mask	(f) CE + TE + Text + Mask

Fig. 5. **Effects of conditional signals.** (a) shows the current canvas and target image (inset). With only predicted CLIP embeddings of the next image (b), the model generates excessive content per update. Including time intervals (c) properly limits new content volume but results in unnatural mountain rendering. Omitting the mask instruction (d) causes the renderer to complete the mountain area in one step, relying heavily on the text instruction "mountain". Omitting the text instruction (e) results in generating some of the green lake (red arrow) before completing the mountain (mask shown in the inset). The full pipeline (f) updates the canvas at a reasonable pace, drawing the top of the mountain in green, before layering on the yellow region. Image courtesy Catherine Kay Greenup.

without noise as input). The cross-attention design of this UNet architecture allows us to seamlessly incorporate textual and time interval information as conditional signals.

The mask generator g_{mask} has two inputs. The first one is spatial input, which includes I_T, I_{t-1}, and M_d. Specifically, we begin by encoding I_T and I_{t-1} using the pretrained VAE image encoder E_I. This encoder reduces the spatial resolution by a factor of 8 and converts the channel dimension from 3 to 4. We then concatenate these encoded images with the downsampled M_d (notation remains unchanged for simplicity) along the channel axis to form the composite spatial input: $[E_I(I_T), E_I(I_{t-1}), M_d]$. This spatial input is fed into the UNet similarly to Sec. 3.1.

The second one is the conditional input, which encodes p_t and Δ_t. We first use the pretrained text encoder g_p in Stable Diffusion to encode p_t, producing a text embedding with 77 tokens, each of dimension 768. For Δ_t, similar to Sec. 3.1, we use g_t (same architecture but different weights) to compute time embeddings with dimension 1 x 768. The time embedding is treated as an additional token and concatenated with the text embeddings to form the 78-token conditional input $[g_p(p_t), g_t(\Delta_t)]$ for cross-attention layers. Formally, we denote the generation of $\hat{M}_t$ as follows:

$$\hat{M}_t = g_{mask}([E_I(I_T), E_I(I_{t-1}), M_d], [g_p(p_t), g_t(\Delta_t)]). \quad (3)$$

During the training, we keep g_p and E_I frozen, and update weights in g_{mask} and g_t. The training is supervised by the downsampled GT mask $M_t = D(I_t, I_{t-1}, \alpha)$ using the binary cross-entropy loss.

3.3 Training: Canvas Rendering

Canvas rendering aims to generate $\hat{I}_t$ using the current and target images, alongside text and mask instructions, within a specified time interval. We modify the diffusion renderer introduced in Sec. 3.1 to integrate text and mask instructions while keeping other components unchanged, as shown in Fig. 3 (third gray box). For training, we use the ground-truth text and mask. Specifically, we replace $g_f(I_{t-1})$ with $g_f([I_{t-1}, M_t])$. Here $[\cdot, \cdot]$ refers to concatenation along channel axis. We modify first layer of g_f to handle this change. The text p_t is encoded by g_p, concatenated with predicted CLIP embeddings and time embeddings, and then input into the cross-attention layers. We found these CLIP embeddings provide semantic features beyond what is captured by explicit text and mask

instructions alone. For instance, consider column 2 in row 1 of Fig. 6, where the text "mountain" and its corresponding mask serve as the instructions. These instructions do not specify whether the mountain should be painted with intricate details or in a rough, preliminary form, with finer details to be added later. Without CLIP embeddings, the model completes the mountain in full detail, deviating from the painting style of the training set. Incorporating the embeddings helps guide the model to adhere to the painting style.

Fig. 5 demonstrates the impact of excluding various conditional signals. By omitting the mask, variant (d) generates the entire semantic content (*i.e.*, mountain) at once, underscoring the importance of pixel-level mask guidance. Without text for high-level semantic guidance, variant (e) unexpectedly paints some of the green lake on the canvas. In contrast, the full pipeline (f) achieves the most natural result by integrating all these conditions.

3.4 Test-Time Generation

At inference time, we begin with a blank (white) canvas I_0, and update this canvas autoregressively using our trained pipeline to approximate a target painting I_T. We use a fixed time interval $\hat{\Delta}_t$ across all steps, following our definition of keyframes. This process is terminated when minimal updates are applied (see supplementary). We visualize an update in a specific step $t-1$ in Fig. 3 (the rightmost gray box). Given the current image $\hat{I}_{t-1}$ predicted by previous step, we first generate the text instruction $\hat{p}_t$ by employing Eq.2 and substituting I_{t-1} with $\hat{I}_{t-1}$. Then we compute the mask $\hat{M}_t$ using Eq.3 by replacing I_{t-1} with $\hat{I}_{t-1}$, M_d with $\hat{M}_d = D(I_T, \hat{I}_{t-1}, \alpha)$, p_t with $\hat{p}_t$, and Δ_t with $\hat{\Delta}_t$. Finally, to render $\hat{I}_t$, we start with random noise z_S and perform S denoising steps using $\{\hat{I}_{t-1}, I_T, \hat{p}_t, \hat{M}_t, \hat{\Delta}_t\}$ through diffusion renderer. This results in z_0 which is decoded by the VAE decoder from Stable Diffusion, producing $\hat{I}_t$. Please refer to the supplementary for details on training and test-time generation, including hyperparameters, execution time, and GT text annotations.

Fig. 6 visualizes consecutive keyframes of the painting process generated by our method, guided by the text and mask instructions. In the early stages of the process (row 1), the method typically focuses on a single semantic class per update. In latter stages of the process (row 2), once the background is completed, the focus
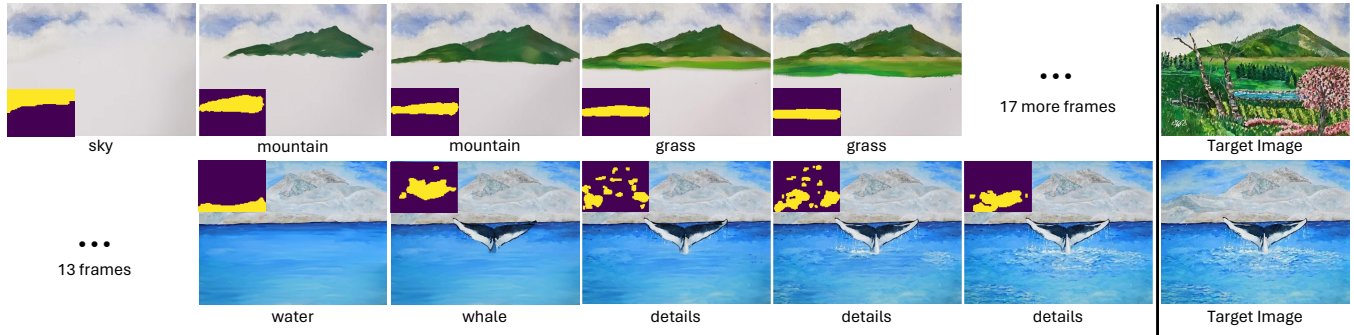
Fig. 6. **Qualitative results with generated texts and masks.** We show five consecutive generated keyframes (left) given in-the-wild target images (far-right). The text (caption) and mask (inset) for each keyframe are the generated instructions used to produce that keyframe. The first row illustrates the early stages of the process, where our method paints sequentially from back (sky) to front (grass), typically focusing on one subject per keyframe based on the provided text and mask. For instance, keyframes 4 and 5 focus exclusively on grass, disregarding other elements like trees (layered on later) that occupy the same region of the target image. The second row depicts the final stages of the painting process for a different painting. Here, our method paints the water before layering on the whale. The process concludes with the addition of final details scattered throughout the painting (as guided by text and mask), mimicking techniques used by human artists. The resulting keyframe sequence reflect human-like decisions in painting order, semantic attention, and layering techniques. Images courtesy Catherine Kay Greenup.

switches to completing foreground and refining details. Here, "details" refer to fine elements typically painted in the later stages such as the ocean spray. In the second row of Fig. 6, columns 4-6 illustrate the finishing of these elements using the same text instruction "details" but different mask instructions over three frames.

4 EXPERIMENTS

Datasets. We collected a dataset of 294 videos with typical acrylic landscape painting processes. The collection features common themes such as mountains, trees, flowers, and lakes, with paintings representing various times of day and weather conditions. Each video averages 9 minutes in length and is often sped up for more efficient viewing. The footage comprehensively captures the complete painting process, including views of the canvas and palette as well as continuous hand and paintbrush movements throughout the video.

For data preprocessing, we employed a pretrained segmentation method [Liu et al. 2023c] to segment all video frames, cropping them to focus solely on the canvas areas. Further, the cropped frames showing hands, paint strokes, or minimal changes were excluded. After preprocessing, we divided the dataset into 265 training and 29 validation paintings. Both subsets have an average of 27 frames per artwork, with time intervals averaging around 23.6 seconds in training and 22.0 seconds in validation between frames. The training and validation sets have 7261 and 783 pairs. *During testing, the time interval is set to 20 seconds unless stated otherwise.*

Our Results. Fig. 7 shows the outcomes of our pipeline on in-the-wild paintings. First, the generated painting process resembles human-like painting orderings, typically painting from back to front, saving foreground objects and fine details for the last stages. For example, in row 1, the sequence starts with the sky, moves to the water, and finishes with foreground objects and details such as the boats, the sun, its reflection, and water texture. Second, each phase of the painting process focuses on specific semantic objects or areas. For instance, transitions from columns 3 to 6 in row 1

mainly focus on the sun, its reflection on the water, and the boats, respectively. Third, the underlayering contents are reliably rendered in the intermediate images when applying the layering techniques. For example, the base of the mountain is painted in row 2, column 1, with additional details added in column 2. Similarly, clouds are layered across the sky in row 6, columns 1 to 2. Finally, our method effectively handles artworks of various aspect ratios and artistic styles such as Impressionism. It also accommodates paintings with varied color themes. More results are in supplementary.

Baselines. We consider three baselines in the main paper (more baselines in supplementary). (1)*Timecraft* [Zhao et al. 2020] employs a conditional variational autoencoder to create time-lapse videos from paintings. It trains on real painting videos to emulate the human painting process. However, it handles only low-resolution 50x50 crops from downscaled paintings (126x168) due to computational limits. We trained the model on our dataset following their training strategy. For evaluation, we resized the target painting to 50x50 as input and scale the output videos back to the original painting's resolution. See the supplementary for evaluation on cropped paintings. (2) *Paint Transformer* [Liu et al. 2021] generates a stroke-level painting process from an input painting. This self-supervised method does not rely on real painting videos, but instead employs a hand-crafted coarse-to-fine strategy to mimic human painting process. We used the pretrained model provided by the authors for our comparisons. Please see supplementary for more stroke-based rendering baselines [Hu et al. 2023b; Zou et al. 2021]. (3) *Stable Video Diffusion (SVD)* [Blattmann et al. 2023] produces a 14- or 25-frame video given the first frame as input. We fine-tuned a 14-frame model on our dataset using LoRA[Hu et al. 2021]. During this fine-tuning, we sampled one frame from our training sequences as input and its previous 13 frames as ground truth, padding with white images where necessary. For inference, the target painting was input into the fine-tuned model to generate 14 frames. The final frame from this sequence initiated the next set of 14 frames, continuing until a white
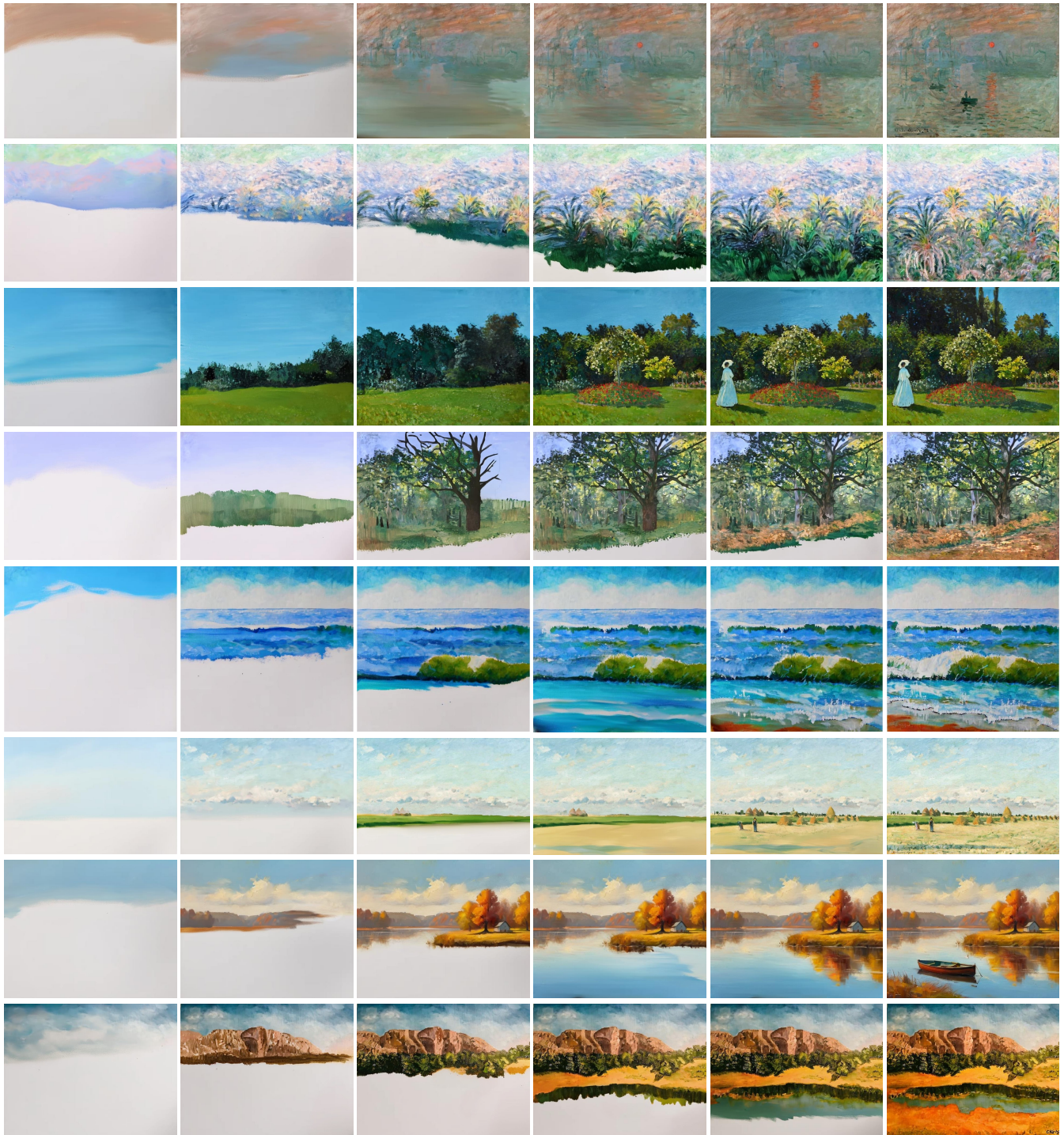
Generated Painting Process (Sampled KeyFrames)

Fig. 7. **Qualitative results.** We show results on in-the-wild paintings (right-most column), where the left five columns are sampled keyframes from the generated painting process. Our method effectively handles landscape paintings across various artistic styles, including Impressionism, Post-Impressionism, and Realism. It also adapts to different color themes, as shown. The generated videos showcase human-like layering techniques, painting orders, and semantic region attention. Images courtesy Catherine Kay Greenup, National Gallery of Art, Washington, and Rawpixel.

canvas is achieved. Please see supplementary for implementation details of all baselines.

Metrics. We aim to evaluate the methods in 5 aspects. (1) Human-likeness of the painting order. This measures whether the generated videos mimic human painting order. (2) Focus consistency in canvas updates. This checks if each update targets specific, reasonable areas. (3) Convergence speed towards the target painting. This evaluates the speed at which the generated video progresses towards the target painting. (4) Adherence to specified time intervals between keyframes. This evaluates whether the temporal progression between generated keyframes aligns with predefined intervals, reflecting the dynamics of an actual painting process. (5) Video quality. We design 5 metrics to cover these aspects.

- LPIPS. This addresses aspects (1) and (4). It calculates the perceptual distance between each frame in the real painting video and the closest generated keyframe based on the time. For example, a real frame at 1:17 is compared with the generated frame at 1:20, assuming a 20-second time interval.
- IoU (Intersection over Union). It evaluates aspect (2) without considering painting order. We first compute difference masks for consecutive frames in both real and generated videos using the function D (defined in Sec. 3.2.2). For each generated difference mask, we then calculate the IoU with all real difference masks, selecting the highest value as the final IoU.
- DDC (Difference of Distance Curve). It evaluates aspects (3) and (4) by comparing the distance curves of the generated and real sequences. The distance curve of a sequence depicts its progression towards the target painting, plotting time (x-axis, minutes) against LPIPS distance (y-axis) between target and current images. We use Dynamic Time Warping (DTW) [Müller 2007] to compute the difference between generated and real curves, accommodating time shifts.
- DTS (Difference of Time Spent). Evaluating aspect (4), DTS measures the temporal difference in minutes between the durations of the real and generated painting videos.
- FID (Fréchet Inception Distance) [Heusel et al. 2017] compares frames in generated and GT videos for aspect (5).

To compute these metrics, for *SVD* and *Timecraft* (trained on our dataset), we set the time interval to 23.6 seconds, matching our training set's average. For the self-supervised *Paint Transformer*, we set it to 2.8 seconds, based on the average training video duration (561 seconds) divided by the number of frames (200). All metrics are calculated for each painting and averaged across the validation set to derive overall performance metrics.

Comparison with Baselines. First, we present a qualitative comparison with baselines, as illustrated in Fig. 8. *SVD* generates artifacts such as the red blocks in the sky regions of column 1. This issue occurs because the target image has a tree that obscures this area, and *SVD* fails to realistically render the obscured regions. It also results in unreasonable painting orders and often stalls during the painting process, leading to extended convergence times (columns 5 and 6). *Paint Transformer* demonstrates a non-human-like painting order in which strokes are added in parallel to different grid cells of a canvas. *Timecraft* produces visible artifacts in low-resolution

Method	LPIPS ↓	IoU ↑	DDC ↓	DTS ↓	FID ↓
Paint Transformer	0.643	0.104	94.61	6.057	337.3
Timecraft	0.602	0.251	153.2	9.964	289.8
SVD	0.500	0.197	135.5	8.577	168.3
Ours-TE-TG-MG	0.468	0.128	88.13	6.204	203.3
Ours-TG-MG	0.447	0.139	62.79	4.913	187.4
Ours-TE	0.413	0.375	58.81	4.153	167.5
Ours-MG	0.435	0.175	61.09	3.972	182.5
Ours-TG	0.399	0.400	39.41	2.120	161.1
Ours-RN	0.416	0.396	46.71	1.542	174.2
Ours-CE	0.371	0.402	34.16	1.346	158.4
Ours 10	0.369	0.349	35.27	1.693	158.7
Ours 30	0.387	0.353	36.26	1.933	151.7
Ours	**0.364**	**0.418**	**32.66**	**1.273**	**150.6**

Table 1. **Comparison with baselines and our ablation variants.** Our full model (with a time interval of 20) outperforms all of them.

videos, likely because its conditional variational autoencoder does not match the image generation quality of diffusion models. Besides, this pure pixel-based method also produces unreasonable painting orders. Our model outperforms these baselines. Finally, we present the quantitative comparisons in Table. 1. Our pipeline surpasses all tested baselines across all metrics. For additional comparisons and analysis, including the visualization of distance curves and the distribution of their slopes, please refer to the supplementary.

Human Study. We conducted a human study on the generated painting processes of 8 in-the-wild paintings. The 33 participants were first presented with 3 examples of the real painting process as reference. Then for each painting, the participants were presented with 4 randomly shuffled painting processes (1 for our method and 3 for baselines), and were asked to give a rating from 1 to 5 (higher means better) for each process. We normalized the rating of each example and user to remove the user bias. Our method achieves the highest average rating, surpassing *SVD* by 1.9 times, *Paint Transformer* by 2.1 times, *Timecraft* by 3.0 times. These show that our method can produce a better painting process than baselines. Please see supplementary for more details.

Ablation Study. We perform two ablation studies. The first one evaluates the different components in the pipeline using 7 variants. (1) *Ours-TE-TG-MG*: full model excluding time embeddings, text, and mask generators. (2) *Ours-TG-MG*: full model without text and mask generators. (3) *Ours-TE*: full model without time embeddings, omitting the time interval in the pipeline. (4) *Ours-MG*: full model without the mask generator. (5) *Ours-TG*: full model without the text generator, omitting text in the pipeline. (6) *Ours-RN*: full model without ReferenceNet, where the target image is inputted to the feature encoder in a manner similar to the current image. (7) *Ours-CE*: full model without the next CLIP generator. Results shown in Table. 1 and Fig. 5 indicate that the full model surpasses all variants. In addition to the discussion in Sec. 3, we observe that: (1) *Ours* outperforms *Ours-TE*, especially on *DDC* and *DTS*, highlighting the importance of time embeddings. (2) *Ours-MG* exhibits poor performance in IoU, illustrating the effectiveness of mask guidance for
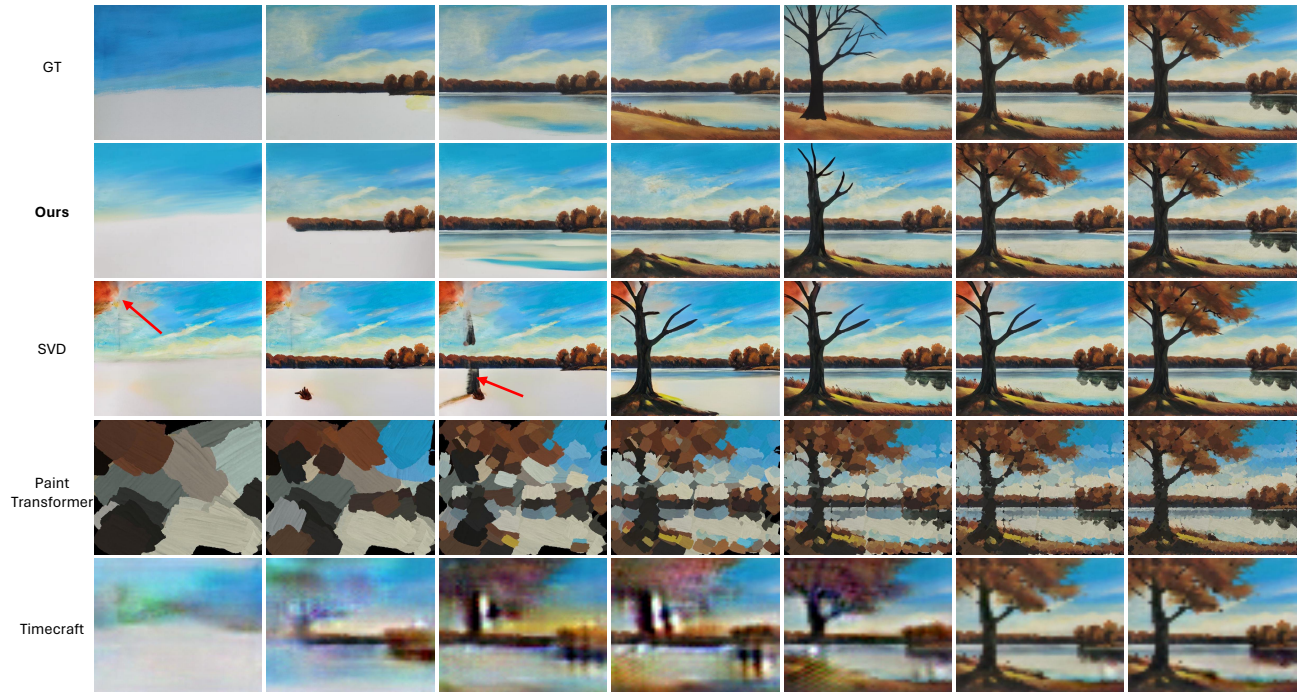
Fig. 8. **Comparison with baselines.** Displayed frames are evenly sampled from GT (top row) and generated videos (other rows). The last GT frame (top-right) is used as input. *SVD* produces noticeable artifacts such as strange color blocks and truncated tree trunks, as highlighted by the red arrows in columns 1 and 3. Additionally, it produces unreasonable painting orders and tends to get stuck during the process (e.g., minimal change between columns 5 and 6). *Paint Transformer* displays a non-human-like painting order and can only produce a "stylized" version of the target painting due to the limitations of parameterized paintstrokes. *Timecraft* results in blurry outputs with noticeable artifacts. In contrast, our method better mimics the human-like painting process and achieves higher visual quality.

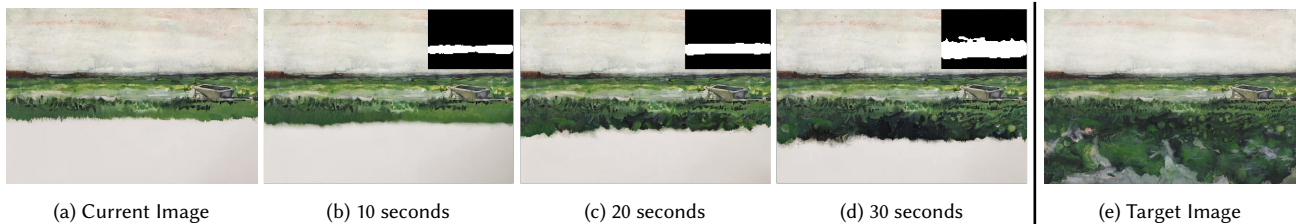| (a) Current Image | (b) 10 seconds | (c) 20 seconds | (d) 30 seconds | (e) Target Image |

Fig. 9. **Ablation of time interval.** (a) and (e) show the current and target image, respectively. (b) to (d) show the predicted next images and their masks (inset) given different time intervals. As the time interval increases, the size of the predicted mask expands, leading to more extensive updates on the canvas. Image courtesy Cleveland Museum of Art.

focus regions. (3) *Ours* outperforms *Ours-RN*, showing the effectiveness of using ReferenceNet to inject target image features into the diffusion model. (4) *Ours* works better than *Ours-CE*, indicating that the next CLIP generator offers beneficial complementary guidance.

The second ablation study evaluates the effect of different time intervals during testing. We test two variants, *Ours 10* and *Ours 30*, with time intervals set to 10 and 30 seconds respectively. Table. 1 shows that the IoU for these variants is lower than *Ours*, which might be because the average time interval in the validation set is around 22 seconds, closer to *Ours* (20 seconds). This suggests that time embeddings effectively guide the mask generator to produce

reasonable size of region masks. Additionally, Fig. 9 visualizes the impact of different time intervals on a single canvas update. A larger time interval leads to a larger region mask and a more substantial update on the canvas, demonstrating how time intervals can be used to regulate the number of frames required to complete a painting. Different time intervals can also be used at various stages of the painting process based on user needs.

Analysis of the Text and Mask Generators. First, we compare the entire sequences of our generated text instructions with the sequences of GT instructions in two ways: (a) without considering order, 91% of our instructions appear in the GT instructions, and

(a) Input   (b) Time Map

Fig. 10. **Emergent property**. The time map (b), derived from the difference masks, is similar to the (inverse) depth map of the target painting (a). This highlights our method's effectiveness in capturing the typical painting principle of layering from back to front, as learned from the training videos. Image courtesy Simon Berger.

(b) taking order into account by computing the longest common subsequence (LCS) between GT and our instructions, 78% of our instructions are included in the LCS. These show that our text generator provides reasonable instructions for the painting order.

Second, instead of evaluating entire text instruction sequences, we evaluate a single canvas update with GT current image and time interval as input. This enables the use of GT text and masks for each update during evaluation. Our text generator produces text instructions that align with GT text 72% of the time, outperforming the pretrained LLaVA model used to initialize it (31%). When providing the predicted text as input to the mask generator, the predicted mask's IoU is 0.64, slightly lower than using GT text (0.70). These demonstrate the text generator's effectiveness.

Finally, we also evaluate our mask generator based on a single canvas update. Generating masks by a pretrained segmentation method [Liu et al. 2023c] using GT text yields an IoU of 0.42, lower than our mask generator's 0.70. Removing the text condition in our mask generator yields a suboptimal IoU of 0.58. These results demonstrate the effectiveness of our mask generator design.

Error Accumulation. Our approach of training the diffusion models with GT inputs helps alleviate error accumulation. Specifically, training with GT texts and masks avoids errors propagated from the trained text and mask generators, achieving better LPIPS score of 0.364 compared to training with predicted texts and masks (0.438). Additionally, training with GT current frames enhances both stability and efficiency. Furthermore, during inference, using the target image as input helps the convergence of the painting process and further alleviates error accumulation.

5 DISCUSSIONS

Emergent Property. Our method showcases an emergent property related to the painting principle, as depicted in Fig. 10. The time map (b) is derived from difference masks $\hat{M}_t$ between consecutive keyframes in the generated videos. Each mask is weighted by its sequence position t and merged into a single image, with overlapping areas selecting the highest value to form the time map. This map is similar to a depth map, indicating that the back-to-front painting principle – reflective of the artists' styles in our training dataset – has been effectively captured by our method.

Fig. 11. **Failure cases.** Trained only on landscapes, our method produces unnatural results on portraits. Image courtesy the MET.

Limitations and Future Work. Our method has several limitations. First, it is trained on landscape paintings and struggles to generalize to other types of paintings like portraits (Fig. 11). Future work will involve extending its applicability to different genres. Second, our method currently exhibits limited painting styles due to the training data. Training on a larger and more diverse dataset could help the model learn various painting styles. Third, our method fails when encountering paintings with large objects (*e.g.*, colosseum) that are uncommon in the landscape. In such cases, the model can still paint the objects with the help of predicted CLIP embeddings, but in an incorrect painting order. Incorporating a semantic map could potentially address this issue.

ACKNOWLEDGMENTS

REFERENCES

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. 2023. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127* (2023).

Ross Bob. 1987. *Bob Ross Experience The Joy of Painting Volume X 10 Ten.* Bob Ross; First Edition.

Bowei Chen, Brian Curless, Ira Kemelmacher-Shlizerman, and Steve Seitz. 2023. Total Selfie: Generating Full-Body Selfies. *arXiv preprint arXiv:2308.14740* (2023).

Manuel Ladron de Guevara, Matthew Fisher, and Aaron Hertzmann. 2023. Segmentation-Based Parametric Painting. *arXiv preprint arXiv:2311.14271* (2023).

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929* (2020).

Annette Dozier. 2007. *Painting peaceful country landscapes.* North Light Books.

Kevin Frans and Chin-Yi Cheng. 2018. Unsupervised image to sequence translation with canvas-drawer networks. *arXiv preprint arXiv:1809.08340* (2018).

Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy J Mitra. 2011. Animated construction of line drawings. In *Proceedings of the 2011 SIGGRAPH Asia Conference.* 1–10.

Yaroslav Ganin, Tejas Kulkarni, Igor Babuschkin, SM Ali Eslami, and Oriol Vinyals. 2018. Synthesizing programs for images using reinforced adversarial learning. In *International Conference on Machine Learning.* PMLR, 1666–1675.

David Ha and Douglas Eck. 2017. A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* (2017).

Paul Haeberli. 1990. Paint by numbers: Abstract image representations. In *Proceedings of the 17th annual conference on Computer graphics and interactive techniques.* 207–214.

Aaron Hertzmann. 1998. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques.* 453–460.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems* 30 (2017).

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685* (2021).

Li Hu, Xin Gao, Peng Zhang, Ke Sun, Bang Zhang, and Liefeng Bo. 2023a. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. *arXiv preprint arXiv:2311.17117* (2023).

Teng Hu, Ran Yi, Haokun Zhu, Liang Liu, Jinlong Peng, Yabiao Wang, Chengjie Wang, and Lizhuang Ma. 2023b. Stroke-based Neural Painting and Stylization with Dynamically Predicted Painting Region. In *Proceedings of the 31st ACM International Conference on Multimedia.* 7470–7480.

Zhewei Huang, Wen Heng, and Shuchang Zhou. 2019. Learning to paint with model-based deep reinforcement learning. In *Proceedings of the IEEE/CVF international conference on computer vision.* 8709–8718.

Biao Jia, Chen Fang, Jonathan Brandt, Byungmoon Kim, and Dinesh Manocha. 2019. Paintbot: A reinforcement learning approach for natural media painting. *arXiv preprint arXiv:1904.02201* (2019).

Dmytro Kotovenko, Matthias Wright, Arthur Heimbrecht, and Bjorn Ommer. 2021. Rethinking style transfer: From pixels to parameterized brushstrokes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 12196–12205.

Alexander Leiser and Tim Schlippe. 2021. AI in art: simulating the human painting process. In *International Conference on ArtsIT, Interactivity and Game Creation.* Springer, 295–308.

Peter Litwinowicz. 1997. Processing images and video for an impressionist effect. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques.* 407–414.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual Instruction Tuning.

Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Ruifeng Deng, Xin Li, Errui Ding, and Hao Wang. 2021. Paint transformer: Feed forward neural painting with stroke prediction. In *Proceedings of the IEEE/CVF international conference on computer vision.* 6598–6607.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. 2023c. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499* (2023).

Xiao-Chang Liu, Yu-Chen Wu, and Peter Hall. 2023b. Painterly Style Transfer With Learned Brush Strokes. *IEEE Transactions on Visualization and Computer Graphics* (2023).

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Meinard Müller. 2007. Dynamic time warping. *Information retrieval for music and motion* (2007), 69–84.

Royi Rassin, Shauli Ravfogel, and Yoav Goldberg. 2022. Dalle-2 is seeing double: flaws in word-to-concept mapping in Text2Image models. *arXiv preprint arXiv:2210.10606* (2022).

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 10684–10695.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems* 35 (2022), 36479–36494.

Jaskirat Singh, Cameron Smith, Jose Echevarria, and Liang Zheng. 2021. Intelli-paint: Towards developing human-like painting agents. *arXiv preprint arXiv:2112.08930* (2021).

Jaskirat Singh and Liang Zheng. 2021. Combining semantic guidance and deep reinforcement learning for generating human level paintings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 16387–16396.

Fan Tang, Weiming Dong, Yiping Meng, Xing Mei, Feiyue Huang, Xiaopeng Zhang, and Oliver Deussen. 2017. Animated construction of Chinese brush paintings. *IEEE transactions on visualization and computer graphics* 24, 12 (2017), 3019–3031.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Zunfu Wang, Fang Liu, Zhixiong Liu, Changjuan Ran, and Mohan Zhang. 2024. Intelligent-paint: a Chinese painting process generation method based on vision transformer. *Multimedia Systems* 30, 2 (2024), 1–17.

Ning Xie, Hirotaka Hachiya, and Masashi Sugiyama. 2013. Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. *IEICE TRANSACTIONS on Information and Systems* 96, 5 (2013), 1134–1144.

Binxin Yang, Shuyang Gu, Bo Zhang, Ting Zhang, Xuejin Chen, Xiaoyan Sun, Dong Chen, and Fang Wen. 2022. Paint by Example: Exemplar-based Image Editing with Diffusion Models. *arXiv preprint arXiv:2211.13227* (2022).

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision.* 3836–3847.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR.*

Amy Zhao, Guha Balakrishnan, Kathleen M Lewis, Frédo Durand, John V Guttag, and Adrian V Dalca. 2020. Painting many pasts: Synthesizing time lapse videos of paintings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 8435–8445.

Zhengxia Zou, Tianyang Shi, Shuang Qiu, Yi Yuan, and Zhenwei Shi. 2021. Stylized neural painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 15689–15698.