DOI: 10.1111/cgf.14688
Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

Abstract Painting Synthesis via Decremental optimization

Ming Yan[1], Yuanyuan Pu[†1], Pengzheng Zhao[1], Dan Xu[1], Hao Wu[1], Qiuxia Yang[1] and Ruxin Wang[2]

[1]Yunnan University, China [2]Alibaba Group, China

Abstract

Existing stroke-based painting synthesis methods usually fail to achieve good results with limited strokes because these methods use semantically irrelevant metrics to calculate the similarity between the painting and photo domains. Hence, it is hard to see meaningful semantical information from the painting. This paper proposes a painting synthesis method that uses a CLIP (Contrastive-Language-Image-Pretraining) model to build a semantically-aware metric so that the cross-domain semantic similarity is explicitly involved. To ensure the convergence of the objective function, we design a new strategy called decremental optimization. Specifically, we define painting as a set of strokes and use a neural renderer to obtain a rasterized painting by optimizing the stroke control parameters through a CLIP-based loss. The optimization process is initialized with an excessive number of brush strokes, and the number of strokes is then gradually reduced to generate paintings of varying levels of abstraction. Experiments show that our method can obtain vivid paintings, and the results are better than the comparison stroke-based painting synthesis methods when the number of strokes is limited.

CCS Concepts

• ***Computer Graphics*** → *Non-photorealistic rendering; Stroke based rendering;*

1. Introduction

Abstraction is an essential technique in painting, requiring the painter to use a few strokes to convey crucial information about the target scene, fully reflecting the painter's wisdom and creativity. For example, the watercolor painting depicts the target scene with a few expressive strokes, and each stroke may need to simultaneously contain meaningful semantic, geometric, color, and illumination information. This kind of creation requires a comprehensive understanding of the target scene and adequate knowledge of the properties of the brushstroke. People usually need a large amount of practice and professional training to master such skills. Even a skilled painter considers carefully to create an excellent abstract painting. This paper will take the watercolor painting as an example to explore how to use algorithms to create abstract paintings.

Today, machines can generate paintings by various techniques, e.g., non-photorealistic rendering [GG01], generative adversarial networks (GANs) [GPAM*14], style transfer [GEB15], and diffusion models [DN21]. Existing methods can produce a wide variety of high-quality paintings. However, the literature dedicated to the generation of abstract paintings is rare. A primary goal in non-photorealistic rendering is automatically generating a corresponding painting based on a given photo. One technique, called stroke-based rendering (SBR) [Her03], defines a painting as a set of strokes controlled by parameters and then rasterizes the painting by a stroke renderer, which is similar to human painting. It is hard to define the abstract level in other techniques. Nevertheless, for SBR, we can use the number of strokes to define the abstraction level: fewer strokes refer to a higher abstraction level. Therefore, as shown in Figure 1, in this paper, we will develop our method based on the SBR technique.

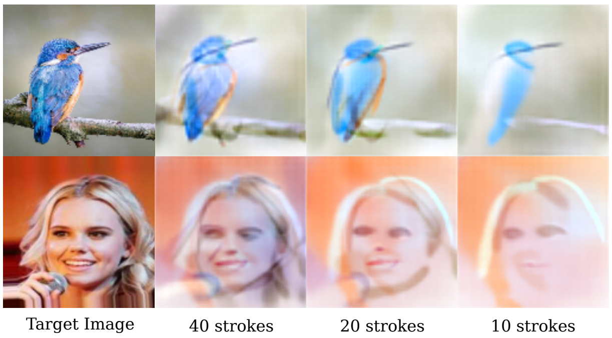


Figure 1: *Given a target image, our method can generate abstract paintings of varying levels of abstraction.*

Existing SBR methods usually limit the number of strokes because the metrics guide these methods in the pixel domain, when the number of strokes is unlimited, the result will be close to the given image, which is meaningless. For the same reason, these

[†] Corresponding author

methods usually fail when we need to create abstract paintings with a minimal number of strokes. Covering the whole image with vague strokes is the optimal solution for minimizing the pixel distance between the input photo and the painting. However, it may lose crucial semantic information, which will lead to pointless results. This problem is particularly acute for the methods that use plain lines to generate sketches. To solve this problem, CLIPasso [VPB*22] uses a pre-trained CLIP [RKH*21] model to construct the loss function that focuses on the semantic similarity between photos and sketches to bridge the gap between the photo domain and the sketches domain, and enables the generation of sketches with different levels of abstraction by using different numbers of strokes.

Inspired by CLIPasso, we will use the CLIP-based loss to guide the generation of abstract watercolor paintings. However, it is not trivial because the objective function based on the CLIP model is highly nonconvex, so the initial values and the optimization strategy may affect the results profoundly. In CLIPasso, the authors use a saliency-guided initialization strategy to guarantee convergence. This paper uses a neural renderer to render watercolor strokes, which differs from the plain lines used in CLIPasso. The complex watercolor strokes are more difficult to optimize, so we cannot use the same strategy directly. Therefore, we propose a new decremental optimization strategy to provide convergence guarantees. For the SBR methods, when the number of strokes is large, the generated painting will be close to the target image. These strokes represent almost the complete information of the target image, so we consider the optimization process of obtaining an abstract painting as a process of keeping the strokes that provide the most information and eliminating irrelevant strokes. Specifically, this strategy starts with an excess of strokes to generate a painting that approximates the target photo and takes the corresponding stroke control parameters as initialization values. Next, at each step of the iterative optimization process, we first take the strokes generated in the last step and evaluate the contribution of each stroke to the semantic similarity to eliminate strokes that do not contribute much to generating paintings of higher abstraction level. The step ends with fine-tuning the remaining strokes using the CLIP-based loss to reduce distortion caused by removing the strokes directly. Our strategy does not need extra models to get the saliency map or image segmentation compared to the saliency-guided initialization strategy. Only a reinforcement learning-based paint agent is used to accelerate the acquisition of initialized strokes. With this strategy, our method can build vivid paintings in different abstraction levels and outperform comparison methods in a limited number of strokes.

Our works are summarized as follows:

- We propose a new decremental optimization strategy combined with the CLIP-based semantic loss for a stroke-based painting synthesis method to create paintings at different levels of abstraction.
- The proposed method performs surprisingly well, especially in a limited number of strokes, demonstrating its robustness and generalization ability, and also validating our motivation.
- Extensive experiments and comparisons show that our method is very competitive with the recent state-of-the-art methods.

2. Related Work

Style transfer and image translation Style transfer is widely used for painting synthesis tasks. It is a general image processing technique for transferring the style of one image to another. When the style is artistic painting, painting synthesis is possible. Texture transfer techniques achieve style transfer initially. A representative method is [EF01], based on texture synthesis and image quilting. Besides that, most of the classic works are developed on texture synthesis, morphology, or other image processing techniques [YO12, MCO11, MBO14]. More recently, Gatys et al. [GEB15] proposed a method that extracts content features and style features of different images by a deep neural network and fuses them to complete image style transfer. Follow by this work, [JAFF16] proposed feed-forward convolutional neural networks to solve the optimization problem approximate. [HB17] implements arbitrary style transfer by manipulating the statistics of the feature map. Another line of work is image translation [PLQC21], which aims at translating images from one domain to another. For the task of painting synthesis, the target domain is artworks. Most of the image translation method is a kind of image-conditional generative model, such as CycleGANs [ZPIE17], Pix2Pix [IZZE17], BicycleGAN [ZZP*17], and their variants [LZH*21]. Generally, these methods use a network that simultaneously learns stroke and content to synthesize painting by directly manipulating pixels, which is different from the human painting process and can not use the number of strokes to control the abstraction level.

Stroke-based rendering Stroke-based rendering (SBR) converts a given image with a limited number of strokes. As a non-photorealistic technique, the goal of SBR is not to reconstruct the image but to represent the image with some artistic style [Her03]. This technique has a long history of research, and earlier work focused on obtaining stroke parameters by greedy search [Lit97] or image segmentation [GCS02] methods. Recent work can be divided into two categories. The basis of optimization approaches is the differentiable renderer. These methods obtain stroke control parameters by optimizing the loss of the target image and rasterized painting [ZSQ*21, MH21, KWHO21]. The feed-forward approach directly predicts the control parameters of the strokes through deep neural networks. Some approach model this as a sequential process [HE17], Some methods use reinforcement learning [HHZ19, GKB*18, ZJH18], and some borrow ideas from object detection [LLH*21]. Our approach is in the optimization category. Besides, because the feed-forward method has a faster speed, this paper will adopt a feed-forward method to initialize the stroke control parameters.

Sketch/doodle generation The purpose of sketch generation is to produce abstract sketches in terms of geometric structure and semantic features. Require depicting the most critical information in a minimal number of strokes, which is very difficult for computers, so sketch generation has attracted many researchers to study it. [BSM*13] using a data-driven approach to learn different artists' styles and abstraction processes to synthesize the portrait. [LSHG17] use the model to learn how to combine strokes for a given category and then transform the edge map of the input image into a sketch by deformation. [ZFW*18] use stroke demonstration and deep Q-learning to train a paint agent that paints in a simulation

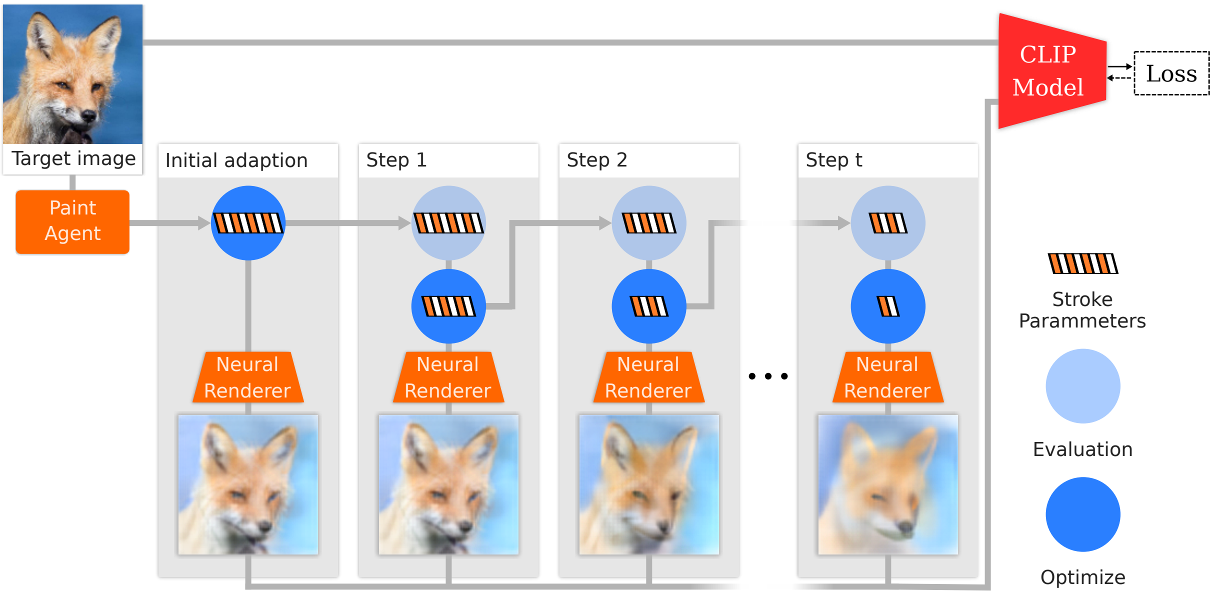


Figure 2: *Overview of our method. Given a target image, we first decompose it into a group of strokes using a pre-trained paint agent and keep the parameters of these strokes. Next, we will use a Decremental optimization strategy with several steps to produce paintings of different abstraction levels. This strategy starts with the initial adaption, optimizing the parameters generated by the agent using a CLIP-based loss. In each step that follows, we take the parameters generated in the last step, evaluate every stroke, and eliminate those strokes that do not contribute much. At last, fine-tune the remaining strokes with the CLIP-based loss to get a higher level of abstraction.*

environment. Spiral++ [MPG*19] is also a reinforcement learning approach to implement a generative agent in an unsupervised manner. [TH21] uses colored triangles as brush strokes and utilizes a modern evolutionary algorithm combined with CLIP-based loss to achieve abstract painting generation. [CDI22] uses CLIP loss as a semantic constraint and incorporates style loss and geometric loss to train the GAN model and improve line drawings. Recent work, CLIPasso, takes abstraction as a central problem in sketch generation and gives a discussion with insight. Our work is based on CLIPasso. While CLIPasso uses plain lines as strokes, our method aims to use complex strokes with color and texture features to generate abstract watercolor paintings, which are closer to Spiral++. It should be emphasized that ensuring the convergence is difficult due to the highly non-convex nature of the CLIP-based objective function. Therefore, the saliency-guided initialization used in CLIPasso cannot be applied directly to our problem.

CLIP-guided image synthesis/manipulation CLIP is a neural network model trained on 400 million image-text pairs collected from the internet based on Contrastive Language-Image Pretraining technique. The model maps a given text and an image to the same embedding space, and the semantic similarity could be evaluated by calculating the cosine distance of the embedding vectors. Since the data used to train CLIP are collected from various sources on the internet and cover various image domains, the models are found to be powerful for a wide range of image synthesis/manipulation tasks. StyleCLIP [PWS*21] is based on stylegan [KLA19] architecture to manipulate the image style by the text prompt through the CLIP model. Similar to this idea, [ZAFW21] delves into the problem of domain gap and presents a method combined with the CLIP model for one-shot domain adaptation. The proposed algorithm can translate any output of the trained GAN from one domain to another. Other works [NDR*21, KY21] combined the CLIP model with the modern diffusion generative model to push text-guided image synthesis to a new stages. [FSW21] and [SLO21] use the CLIP model to guide an optimization process of a stroke-based painting synthesis, which is similar to our method. However, our method is specifically designed to generate abstract paintings and is guided by the given target image rather than a text prompt.

3. Methodology

Figure 2 provides an overview of our method. In general, Our approach consists of four components: 1) a neural renderer for rendering the given stroke parameters into a rasterized image; 2) a reinforcement learning agent for generating initialized stroke parameters; 3) a CLIP-based loss to measure the distance between the target image and the rendered canvas, which is aware of semantic similarity; and 4) a decremental optimization method combined with the CLIP-based loss to gradually reduce the number of strokes to obtain the paintings at different levels of abstraction.

We will introduce each component in the following.

3.1. Stroke Renderer

A differentiable stroke renderer is essential to optimize the parameters of the strokes. Many studies have utilized diffvg [LLGRK20] to render strokes. In this paper, we define discrete strokes based on an open-source paint library called libmypaint [lc22] and train a differentiable neural renderer to imitate it. A major advantage of libmypaint is that it provides more realistic watercolor strokes than diffvg. Additionally, an approximate blending method is chosen to merge the strokes for a fast speed in the optimization process.

Stroke define The trajectory of a single stroke is modelled as a quadratic Bézier curve:

$$B(t) = (1-t)^2 P_0 + 2(1-t)tP_1 + t^2 P_2, \tag{1}$$

where $0 \leq t \leq 1$, and $P_0 = (x_0, y_0)$,$P_1 = (x_1, y_1)$,$P_2 = (x_2, y_2)$ is the three control points. Then, the control parameter of a single stroke is defined as the following tuple:

$$a = (x_0, y_0, x_1, y_1, x_2, y_2, R, G, B, p_1, p_2, s), \tag{2}$$

where (R, G, B) is the color of the stroke, and follows the API of libmypaint, p_1, p_2 is the start and end pressure, and s is the stroke size.

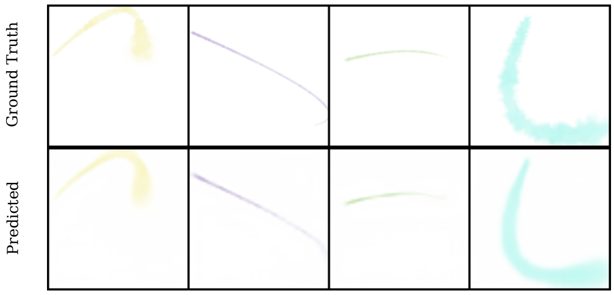


Figure 3: *Samples of the stroke.*

Neural renderer The neural renderer is a deep convolutional network R trained to imitate libmypaint engine to generate discrete stroke, it takes the stroke control parameter a as input, and output is the stroke foreground $\mathbf{S}$ and the alpha map α:

$$\mathbf{S}, \alpha = R(a). \tag{3}$$

We train our neural renderer with the ℓ_2 loss on both the stroke blended on a white and a black canvas. The objective function used in training is:

$$\mathcal{L}_R(a) = \mathbb{E}_{a \sim \mathrm{U}(a)} \left[||C_W - \hat{C}_W||^2 + ||C_B - \hat{C}_B||^2 \right], \tag{4}$$

where C_W and C_B are the predicted strokes generated by the neural renderer blended on the white and the black canvas, $\hat{C}_W$ and $\hat{C}_B$ are the ground-truth strokes generated by libmypaint blended on the white and the black canvas. $\mathrm{U}(a)$ is a uniform distribution defined on the parameter space. We use the Adam [KB14] optimizer to minimize $\mathcal{L}_R$ in practice. The stroke samples generated by the trained neural renderer are shown in Figure 3.

Blending As illustrated in Figure 4, given an empty canvas C_0 and a sequential of N strokes controlled by the parameters $\{a_t\}_{t=1}^N$, the soft blending is defined as:

$$C_t = \alpha_t \mathbf{S}_t + (1 - \alpha_t) C_{t-1}, \tag{5}$$

where $\mathbf{S}_t, \alpha_t = R(a_t)$, and after N steps of soft blending, we get the final canvas C_N as the output painting. Since the recursive process of soft blending is time-consuming, we use an approximate blending method to merge the strokes:

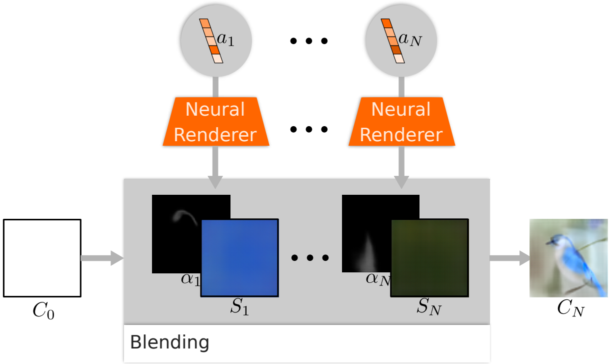


Figure 4: *Given an empty canvas C_0 and a sequential of N strokes controlled by the parameters $\{a_t\}_{t=1}^N$, we first use neural to predict the alpha map α and foreground S of each stroke, then merge the strokes to get the output canvas C_N through a blending method.*

$$C_N \triangleq B(\{a_t\}_{t=1}^N) = \sum_{t=1}^{N} \alpha_t \mathbf{S}_t + C_0 (1 - \sum_{t=1}^{N} \alpha_t), \tag{6}$$

which can be calculated in parallel, and speeded-up by GPU easily. Due to the characteristic of our watercolor stroke, the α_t will usually be very small, so the results of approximate blending and soft blending will be similar, and the implicit conditions $\sum_{t=1}^{N} \alpha_t \leq 1$ will be fulfilled in most cases.

3.2. Stroke Initialize

In our method, we utilized a paint agent from [HHZ19] to get a better initialization of the strokes quickly. Many previous methods have based their optimized process on randomly initialized strokes. Since the objective function in our method is highly non-convex, the convergence results are susceptible to the initialization, especially when the number of strokes is small. A saliency-guided initialization strategy is applied in CLIPasso to tackle this problem, which places the initial strokes based on the salient regions of the target image. Our strokes are different from the simple line used in CLIPasso, so the same strategy may not be suitable. In this paper, we consider using excessive strokes as the initial, then remove the strokes gradually to get abstract paintings. A straightforward solution is to initialize the stroke randomly and then utilize our decremental optimization strategy. However, this requires additional optimization steps, which are time-consuming, and there is no guarantee of getting the desired results. So we consider other methods to get the initial strokes.

The results of feed-forward [LLH*21, HHZ19] methods usually lack artistic abstraction when the number of strokes is large or may fail to convey critical information when the number of strokes is limited. However, their inference time is fast. To this end, we consider using a feed-forward method to get excessive strokes as the initialization. There are certain drawbacks associated with [LLH*21]: the stroke loss is only available for specialized oil paint brush strokes. Considering the strokes generalization, we adopt [HHZ19] to get the initial strokes.

We utilized our stroke renderer in the framework of [HHZ19] to train the paint agent. According to [HHZ19], given a target image I, the agent aims to find a stroke sequence $A_{\text{RL}} = \{a_i\}_{i=1}^{N}$, that after rendering these stokes via soft blending on empty canvas C_0, we can get the final canvas C_N which is similar to I visually. Since the stroke number N is large enough, the ℓ_2 distance between the final canvas C_N and I will be very close, so the visual information on I will be preserved almost intact. We keep A_{RL} as the initial strokes.

3.3. CLIP-based Loss

As we discussed before, pixel-wise metrics are not sufficient to measure the similarity between abstract paint and realistic photos because the same information expressed in the two domains may be very different at the pixel level. Compared to pixel-wise metrics, the perceptual losses such as LPIPS [ZIE*18] based on pre-trained deep neural networks are proved to be close to human perceptual similarity judgments. Nevertheless, it fails to encode the abstract paint to the same semantic embedding as the photo at such a high-level abstraction. The CLIP model maps the image and text to a shared encoding space of a vector, and the cosine distance will be close if two inputs have similar semantical information. The model is trained with enormous image data, including various domains, to robustly measure the cross-domain semantic similarity. To this end, we use the visual branch of a pre-trained CLIP model to build our metric.

We first define the high-level semantic loss based on the original CLIP vector embedding:

$$\mathcal{L}_{\text{semantic}} = dist(\text{CLIP}(I), \text{CLIP}(C)), \tag{7}$$

where CLIP(I) and CLIP(C) is the CLIP embedding of the target image I and the rendered canvas C, $dist(x,y) = 1 - \frac{x \cdot y}{\|x\| \cdot \|y\|}$ is the cosine distance. The augmentation scheme used in CLIPasso is not adopted to save computational resources, and it has been proven that the augmentation does not significantly impact our approach. Next, as a complementary to the semantic loss, we define the geometric loss based on the intermediate level activation of CLIP to measure the similarity of the color, texture, and spatial features:

$$\mathcal{L}_{\text{geometric}} = \sum_{l} \|\text{CLIP}_l(I) - \text{CLIP}_l(C)\|_2^2, \tag{8}$$

where CLIP_l is the CLIP encoder activation at layer l. From above, the objective of the optimization is defined as:

$$\mathcal{L}_{\text{CLIP}} = \mathcal{L}_{\text{geometric}} + w \cdot \mathcal{L}_{\text{semantic}}, \tag{9}$$

where w is the weight of the semantic loss.

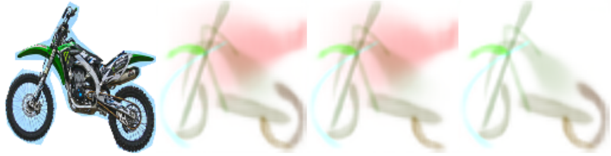
3.4. Decremental Optimization

Since we have the initial strokes A_{RL}, we use the metric described above to evaluate every single stroke and remove insignificant strokes gradually to get a different level of abstract paint, referred to as the decremental optimization strategy. The details of the strategy are specified next.

Stroke evaluation To start, the fundamental of our strategy is the evaluation of the single stroke. For each stroke, we use the loss value after removing the stroke minus the loss value before removing the stroke as the score of this stroke. Suppose we have N strokes $\{a_t\}_{t=1}^{N}$, the corresponding canvas is $C_N = B(\{a_t\}_{t=1}^{N})$, if we remove a single stroke a_i, the corresponding canvas is $C_N^i = B(\{a_t\}_{t=1}^{N(t \neq i)})$. Then, the score of stroke a_i is defined as:

$$s_i \triangleq \mathcal{S}(i, \{a_i\}_{i=1}^{N}, I) = \mathcal{L}_{\text{CLIP}}(I, C_N^i) - \mathcal{L}_{\text{CLIP}}(I, C_N), \tag{10}$$

Refer to Figure 5, a higher score means the more important the stroke is. When the score is zero, it means that this stroke does not contribute any, and when the score is negative, it means that the loss may be reduced by removing the stroke.



Target Image | Before removing a stroke | After removing a stroke (positive-scored) | After removing a stroke (negative-scored)

Figure 5: *Example of the stroke evaluation. Removing the stroke on the rear wheel will make the canvas far from the target image, so it has a positive score; removing the red stroke will make the canvas close to the target image, so it has a negative score.*

Initial adaption Next, we fine-tune the initial parameters to adapt approximate blending. We use soft blending in the case of an agent that generates initialized strokes and approximate blending in the optimization process. The same stroke control parameters may produce different results in the two blending methods, so we fine-tune the parameters to adapt the approximate blending. Given the initial strokes A_{RL}, we optimize the objective function below to fine-tune the parameters:

$$\min_{A_{\text{RL}}} \mathcal{L}_{\text{CLIP}}(I, B(A_{\text{RL}})), \tag{11}$$

In practice, we use the Adam [KB14] optimizer with a few steps T to get the approximate solution of this objective, denote as

$$\text{Adam}(\min_{A_t} \mathcal{L}_{\text{CLIP}}(I, B(A_t)), T). \tag{12}$$

Algorithm 1 Regular step of decremental optimization

Input: A target image I; strokes generate in last step A_{t-1}.
Output: Strokes of the current step A_t.
Default: Constant k; Fine-tune steps T.

```
function RegularStep($A_{t-1}, I$)
    $S_t = \{s_i | s_i = \mathcal{S}(i, \{a_i\}_{i=1}^{N}, I)\}_{i=1}^{N}$;
    $A_{t-1}^{\text{sort}} = \text{Sort}(A_t; S_t)$;
    $A_t = \{a_i | a_i \in A_{t-1}^{\text{sort}}\}_{i=1}^{N-k}$;
    $A_t \leftarrow \text{Adam}(\min_{A_t} \mathcal{L}_{\text{CLIP}}(I, B(A_t)), T)$;
    return $A_t$.
end function
```

Optimization process For simplicity, we use the number of strokes to define the abstraction level. The optimization process is divided into several regular steps associated with a different number of strokes: if we have N input strokes in the first step, the number of the input strokes in step t is $N-k(t-1)$, where k is a constant, and here we set $k=10$. At each regular step of the optimization process, we start by taking the strokes determined in the last step, then scoring each stroke using the evaluation method described above and ranking it by the score. Next, we remove the last strokes, reducing the number of strokes to a higher abstraction level. Finally, the canvas may be distorted after removing the strokes crudely, so we use the same operation in the initial adaption to fine-tune the remaining strokes. Specifically, at step t, if the N strokes are generated in the last step $t-1$ is $A_{t-1} = \{a_i\}_{i=1}^N$, we first evaluate every stroke to get their scores:

$$S_t = \{s_i | s_i = \mathcal{S}(i, \{a_i\}_{i=1}^N, I)\}_{i=1}^N. \tag{13}$$

Then, we sort the strokes by their corresponding scores:

$$A_{t-1}^{\text{sort}} = \text{Sort}(A_t; S_t). \tag{14}$$

After eliminating the last strokes ranked in, the remaining strokes at step t is:

$$A_t = \{a_i | a_i \in A_{t-1}^{\text{sort}}\}_{i=1}^{N-k}, \tag{15}$$

and the final output stroke is fine-tuned by optimizing the objective function:

$$\min_{A_t} \mathcal{L}_{\text{CLIP}}(I, B(A_t)). \tag{16}$$

In practice, supposing we have the strokes $A_{\text{ada}} = \{a_i\}_{i=1}^N$ after initial adaption and the sorted strokes are $A_{\text{ada}}^{\text{sort}}$, we will remove the strokes whose scores are not positive. If the number of the removed strokes is r, for simplicity, we will remove n more strokes ranked in the last, where $n < k$ and $N-r-n = \lfloor N-r \rfloor_k$ (which means $(N-r-n)\text{Mod}k = 0$). So the stroke number at step $t=1$ is $N' = N-r-n$. The regular steps are summarized in the form of pseudo-code in Algorithm 1, and the overall pipeline of decremental optimization is summarized in Algorithm 2.

Algorithm 2 Decremental optimization

Input: A target image I; Initial strokes A_{RL}.

Output: Paintings in different abstraction levels $\{B(A_t)\}_{t=1}^{N'/k}$.

Default: Constant k; Fine-tune steps T; $t=0$.

```
$A_{\text{ada}} = \text{Adam}(\min_{A_{\text{RL}}} \mathcal{L}_{\text{CLIP}}(I, B(A_{\text{RL}})), T)$;
$N = \mathbf{n}(A_{RL})$;
$S_{\text{ada}} = \{s_i | s_i = \mathcal{S}(i, A_{\text{ada}}, I)\}_{i=1}^N$;
$A_{\text{ada}}^{\text{sort}} = \text{Sort}(A_{\text{ada}}; S_{\text{ada}})$;
$r = \mathbf{n}(\{s_i | s_i \leq 0, s_i \in S_{\text{ada}}\})$;
$N' = \lfloor N-r \rfloor_k$;
$A_0 = \{a_i | a_i \in A_{\text{ada}}^{\text{sort}}\}_{i=1}^{N'}$;
for $t < N'/k$ do
    $t \leftarrow t+1$;
    $A_t = \text{REGULARSTEP}(A_{t-1}, I)$;
end for
return $\{B(A_t)\}_{t=1}^{N'/k}$.
```

4. Experimental Analysis

4.1. Experiment Environment and Settings

Enviroment The stroke engine used in our work is libmypaint 1.6.1. The neural renderer and the optimization method are coded with PyTorch 1.9, running on an NVIDIA GTX Titan X GPU.

Settings of neural renderer For the neural renderer, we use the same architecture as [ZSQ*21], set the rendering output size to 128×128. In training, we use the Adam optimizer with the learning rate of $2e-4$, and the betas of $(0.9, 0.999)$. We reduce the learning rate by $1/10$ in every 100 epochs and stop training after 200 epochs since the model is converged. There are 50000 randomly generated ground truth strokes in each epoch, and the batch size we use is 64.

Settings of paint agent For the paint agent, we use the same training framework as [HHZ19] and replace the original renderer with ours. The agent is trained with 2×10^5 mini-batch with the CelebA [LLWT15] datasets, the action bundle size is five, and the number of strokes is 200.

Settings of CLIP-based loss We use the official published ResNet101 model to build our CLIP-based loss and select Layers 3 and 4 of the model to calculate the geometric loss. We set the weight of semantic loss w to 0.1.

Settings of decremental optimization The target image we used to test our method is from various datasets, including CelebA [LLWT15], CUB_Bird [WBW*11], AFHQ [CUYH20], Object_Sketch [GLX*20], VGG_Flowers [NZ08] and ImageNet [DDS*09]. We set $k=10$ and $T=100$. The parameters of Adam optimizer is that learningrate=0.01 and betas=$(0.9, 0.999)$.

4.2. Abstract Painting Synthesis

Figure 10 show the paintings generated by our method. We can see no significant difference between the results of more than 40 brush strokes. That is because high-frequency information is difficult to represent through watercolor brush strokes. After depicting the key low-frequency features through a small number of strokes, additional strokes usually do not continue to refine the details of the painting, so there are a large number of redundant strokes. Our approach exhibits powerful abstraction capabilities between 40 to 10 strokes. Every stroke is elaborate, and it conveys rich information. Especially in the case of 10 strokes, our method uses a limited number of strokes to depict the dominant features of the object, which is very similar to human painting and demonstrates distinctive effects.

4.3. Comparison with other methods

4.3.1. Qualitative comparison

In Figure 11. we compare the painting results generated by different methods. Paint-Transformer [LLH*21] and Huang et al [HHZ19] predict the strokes in a feed-forward manner, Zou et al [ZSQ*21], CLIPasso [VPB*22] and our method is based on optimization. The figure shows that our method outperforms other methods with a limited number of strokes. In the case of a small number of strokes, most results of the comparison methods are blurred and lose crucial information. CLIPasso can depict the semantic features of objects, but only in line sketches.

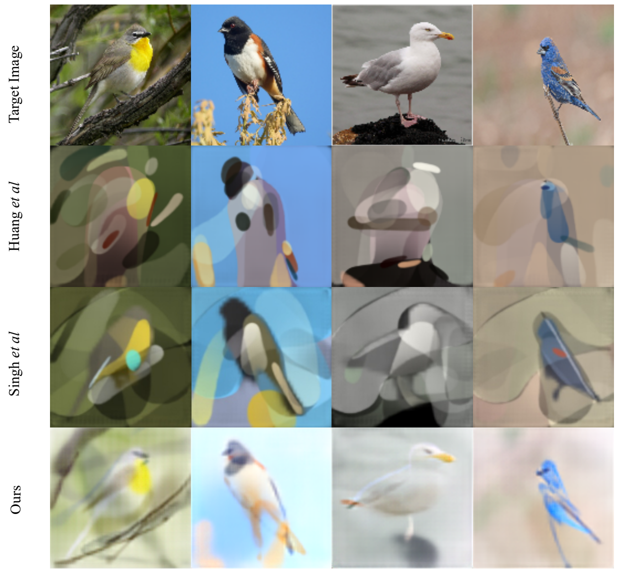


Figure 6: *Compare with Singh et al (20 strokes). From top to bottom: target image, Huang et al [HHZ19], Singh et al [SZ21], our method.*

We also compare our method with [SZ21], which is developed on [HHZ19] and guides the paint agent based on semantic segmentation techniques. Since the official implementation of [SZ21] is only available on the CUB_bird dataset, we select some samples from the CUB_bird dataset for comparison in Figure 6. We can see that although [SZ21] spends more strokes on the foreground objects compare to [HHZ19], the painting results are still inferior to our method with a limited number of strokes.

4.3.2. Quantitative comparison

User study To our best knowledge, there is no objective quantitative criterion for abstract painting synthesis tasks in the literature. To provide a statistical evaluation of the performance of the methods, we conduct a user study with 103 participants to judge the clarity of the synthesized results. We randomly selected 50 images from each dataset to construct a small test set. Each participant was presented with 50 target images from the test set, and each target image was attached with five abstract paintings synthesized by different methods. Participants were asked to choose the most clarity painting for each target image. Table 1 shows the selected rates of different methods on different datasets. The results demonstrate that our method significantly outperforms other approaches.

Computational time Table 2 compares the computational time of the different methods. Our method does not have an advantage over the feed-forward methods [LLH*21, HHZ19] but is still better than CLIPasso [VPB*22] as a baseline.

4.4. Ablation Studies

We study how the components affect the performance of our methods in this section.

Method	A	B	C	D	E
CelebA	0.12%	0.25%	0.37%	14.75%	84.5%
CUB_Bird	0%	1.44%	0.11%	17.55%	80.88%
AFHQ	0.17%	4.08%	1.82%	13.04%	80.86%
Object_Sketch	0.13%	3.73%	0.4%	13.86%	81.86%
VGG_Flowers	0%	8.85%	0.28%	0%	90.85%
ImageNet	0.11%	6.82%	1.88%	9.88%	81.29%
Average	0.08%	4.19%	0.81%	11.51%	83.37%

Table 1: *Selected rates of different methods (A: [LLH*21], B: [ZSQ*21], C: [HHZ19], D: [VPB*22], E:Ours).*

Method	A	B	C	D	E
Computational time(s)	1	14	1	294	174

Table 2: *Computational time of different methods (A: [LLH*21], B: [ZSQ*21], C: [HHZ19], D: [VPB*22], E:Ours).*

4.4.1. Objective Function

To explore the impacts of the objective function, we visually compare the abstract paintings generated by our method with the pixel-wise ℓ_1 loss, the transportation loss, the LPIPS [ZIE*18] loss, and the semantically-aware CLIP-based loss. Figure 12 shows the comparison result on several target images from different datasets. We can see that the painting generated with the pixel-wise loss is blurred, and the objects may not even be recognized for some target images. The transportation loss complements pixel-wise loss, which helps to solve the zero gradient problem and is proved effective in [ZSQ*21]. However, there is no significant improvement in our setting. The LPIPS loss helps to recover more details but still loses some crucial information in portraits. The results of the CLIP-based loss are most recognizable, and we can see more semantic information and "abstraction". Particularly, the portrait of the puppy dog generated by LPIPS is closer to the target image, but it conveys "cute-looking" information in the case of the CLIP-based loss. Moreover, the background flower is removed automatically in the paint of the sunflower.

4.4.2. Stroke Initialization and Optimization

In this experiment, we fix the objective function to the CLIP-based loss and investigate the impact of different initialization and optimization methods under the following conditions:

1. Random initialize 20 strokes and optimize with regular Adam for 1000 iteration steps;
2. Direct predict 20 strokes using paint agent and optimize with regular Adam for 1000 iteration steps;
3. Random initialize 200 strokes and optimize with decremental strategy;
4. Predict 200 strokes using paint agent and optimize with decremental strategy.

We provide a visual comparison between different conditions in Figure 13. Random initialization without the decremental optimize strategy produces the worst results, where most of the objects are

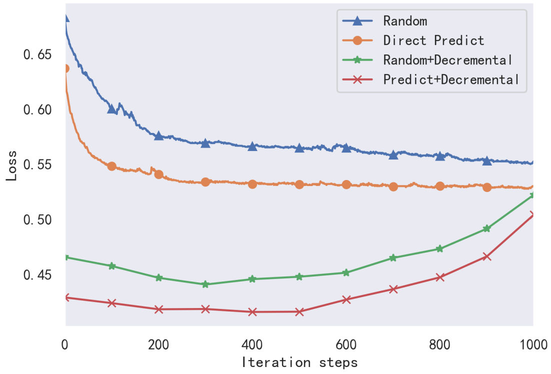


Figure 7: *Convergence comparison with different initialization and optimization strategy (The joker in Figure 13, last 1000 iteration steps of the Adam optimizer).*

unrecognizable. Direct predicting 20 strokes is slightly improved compared with Random initialization, but most of the paintings are still collapsed. The results of the Decremental optimize strategy with random initialization and agent prediction is very close, and far better than the first two conditions. But it must be stated that the number of iterations in initial adaption is set to 500 in the case of random initialization, which is time consuming compared to predicting the stroke by the paint agent directly. Besides, we take the joker in Figure 13 as an example and compare the convergence in Figure 7. (Note that since the number of strokes is decreasing, the loss curve of the decremental optimize strategy is ascending.)

4.4.3. Alternative Stroke Representations

Besides the watercolor, our method can also generalize to alternative strokes. Here we train our neural renderer to imitate the strokes defined in [ZSQ*21] and [HHZ19] rendered by their custom renderer. Similar to our stroke, these strokes are discrete, and the control parameter of a single stroke consists of several real numbers, so we only need to make minor changes to the input layer of our neural renderer. Then, we train the paint agent and utilize the decremental strategy to synthesize abstract paintings. The results are shown in Figure 8. We can see that our method produces abstract paintings robustness with different strokes. Furthermore, the high-resolution result can be produced with the stroke has vectorized characteristics as shown in Figure 9. We first use the decremental strategy with the neural renderer to find the stroke parameter in low resolution (128 × 128), then feed these parameters to the custom renderer provided by [HHZ19] and set the output of the renderer to a high resolution (512 × 512) to get the final result.

5. Limitations

Although our method can obtain good results under restricted conditions, the work in this paper still has some limitations. The first limitation of our approach is that the resolution of the synthesized painting is currently too low, which can be addressed using a practical stroke rendering pipeline. The second limitation of our method is that we use fixed stroke numbers to define abstract levels. However, the same abstract level may need a different number of strokes for different content. This correlation can be studied in future work. Finally, we use the clarity of the synthesized result as the criterion to evaluate the methods' performance, which is insufficient. An improved criterion can be further explored for the abstract painting synthesis.

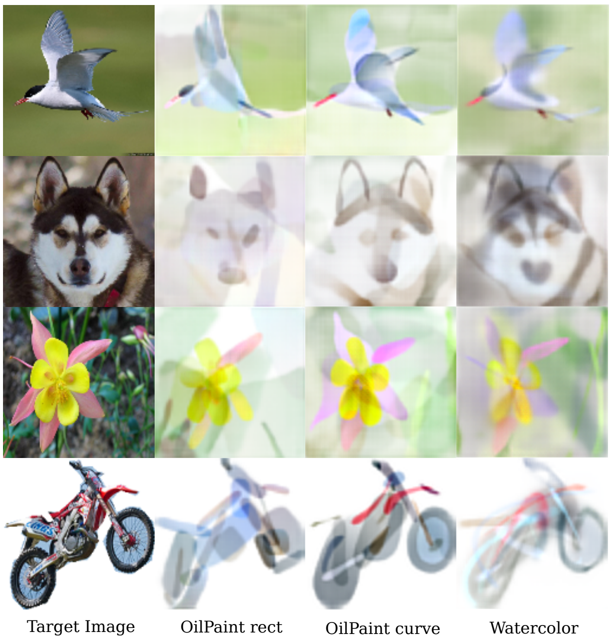


Figure 8: *Abstract paintings using different stroke types. From left to right: target image, rectangle oil paint stroke from [ZSQ*21], curve oil paint stroke from [HHZ19], watercolor.*

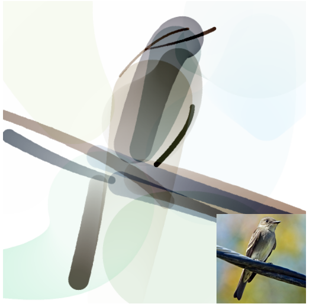


Figure 9: *High-resolution result with the stroke from [HHZ19].*

6. Conclusions

This paper presents a new method for stroke-based abstract painting synthesis. The method can generate the corresponding paintings of the given target image in different levels of abstraction. In our method, we use the number of strokes to define the abstract level. The synthesis process started with an excessive number of strokes which gradually reduced them to obtain a higher and higher level of

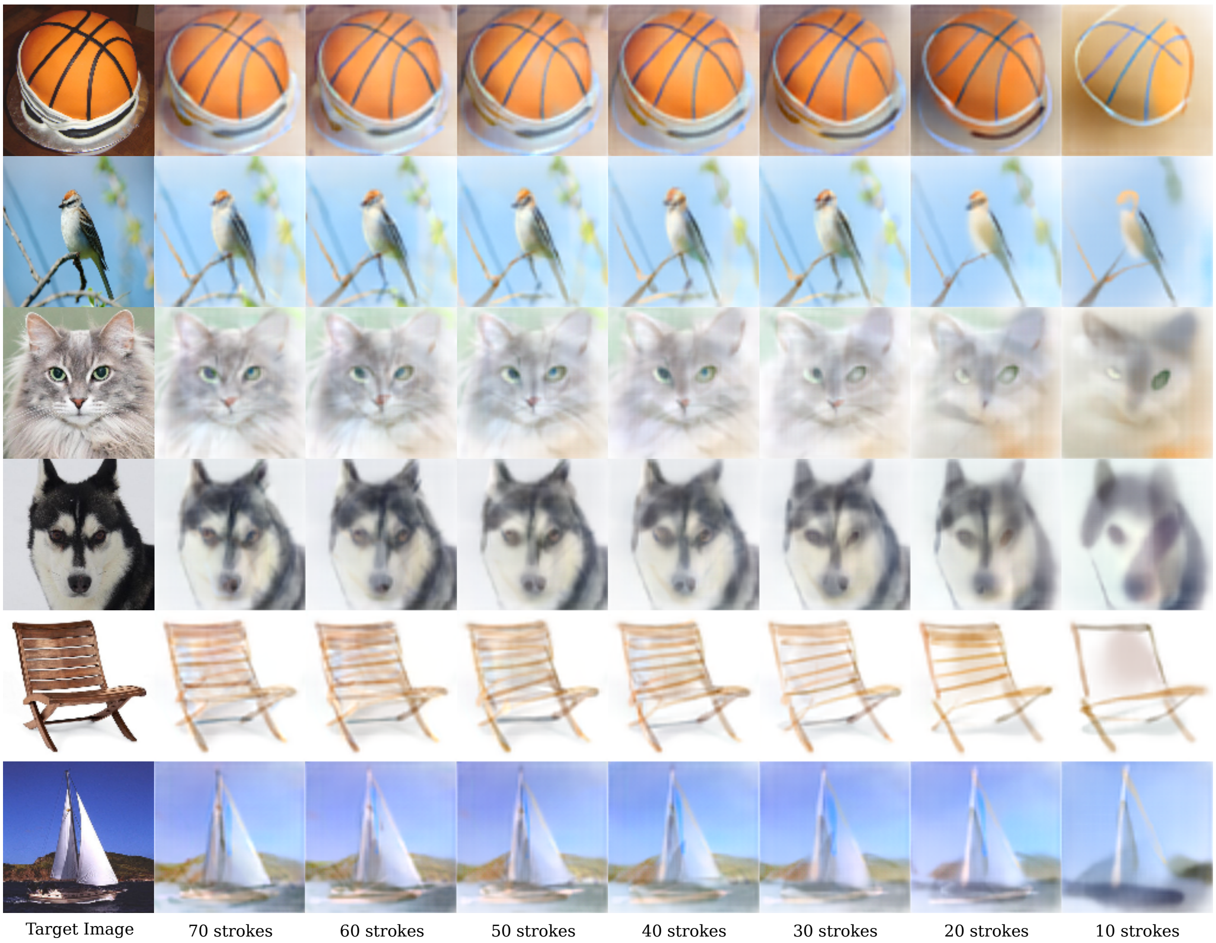


Figure 10: *The paintings in different levels of abstraction created by our method.*

abstraction. Our approach consists of four parts. First, a neural renderer is utilized to make the rendering process differentiable. Second, a reinforcement learning paint agent is used to generate the initialized strokes efficiently. Third, we use a loss function based on the CLIP model to measure the similarity between the target image and the rendered canvas. Fourth, a decremental optimization strategy that gradually reduces the number of strokes is adopted to generate painting in different levels of abstraction. Experimental results demonstrate that our method can be adapted to various target images and get better abstract paintings with limited strokes compared to other methods. Ablation studies confirm the validity of our design.

Acknowledgements

This work is supported by National Science and Technology Founda-tion of China No. 61163019, 61271361, 61662087, 61761046, U1802271 and 62061049; Yunnan Science and Technology Depart-ment Project No. 2014FA021 and 2018FB100; Key Project of Applied Basic Research Program of Yunnan Provincial Department of Science and Technology No. 202001BB050043,2019FA044; Major Science and Technology Special Project in Yunnan Province No. 202002AD080001; Reserve talents of young and middle-aged academic and technical leaders in Yunnan Province No. 2019HB121

References

[BSM*13] Berger I., Shamir A., Mahler M., Carter E., Hodgins J.: Style and abstraction in portrait sketching. *ACM Transactions on Graphics (TOG) 32*, 4 (2013), 1–12. 2

[CDI22] Chan C., Durand F., Isola P.: Learning to generate line drawings that convey geometry and semantics. *arXiv preprint arXiv:2203.12691* (2022). 3

[CUYH20] Choi Y., Uh Y., Yoo J., Ha J.-W.: Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 8188–8197. 6

[DDS*09] Deng J., Dong W., Socher R., Li L.-J., Li K., Fei-Fei L.: Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255. doi:10.1109/CVPR.2009.5206848. 6

[DN21] Dhariwal P., Nichol A.: Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems 34* (2021). 1

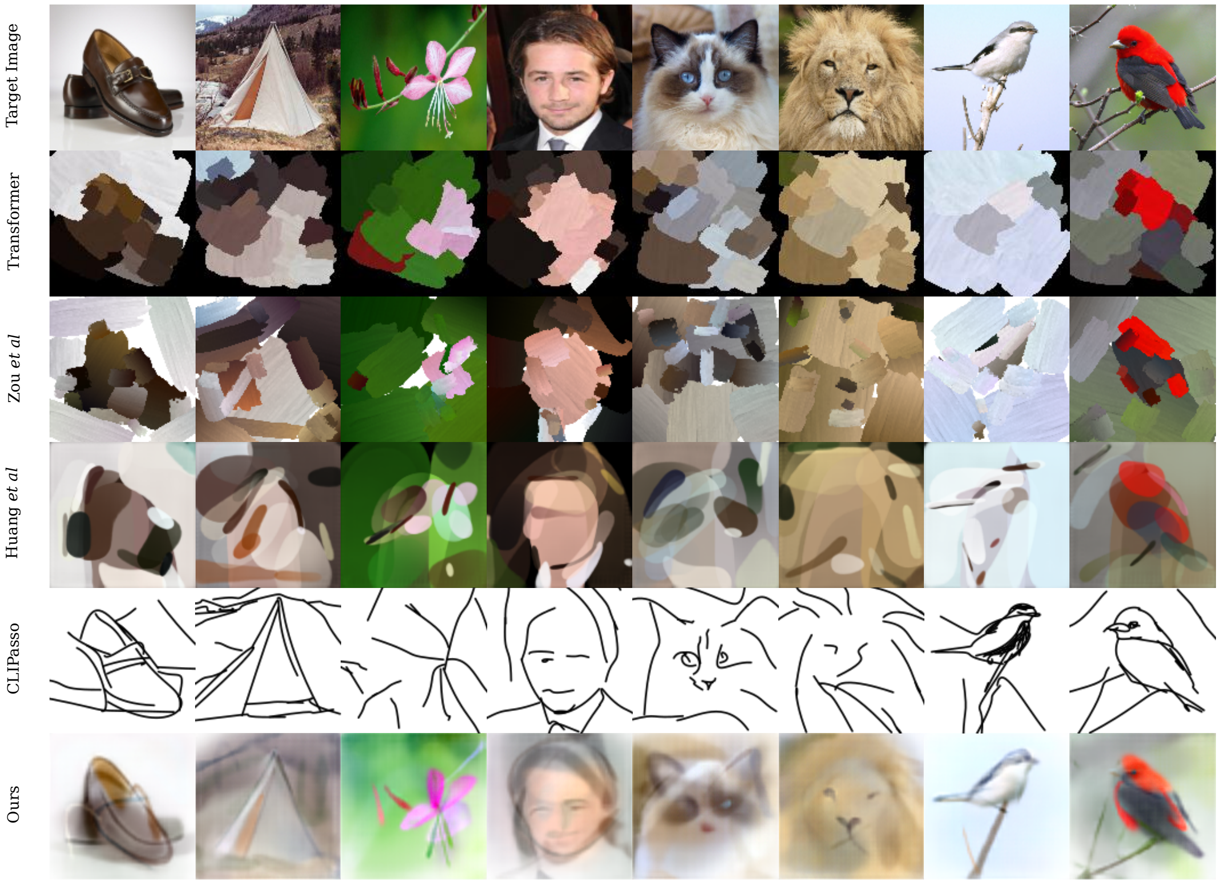


Figure 11: *Compare with other methods (20 strokes). From top to bottom: target image, Paint-Transformer [LLH*21], Zou et al [ZSQ*21], Huang et al [HHZ19], CLIPasso [VPB*22], our method.*

[EF01] Efros A. A., Freeman W. T.: Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), pp. 341–346. 2

[FSW21] Frans K., Soros L., Witkowski O.: Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *arXiv preprint arXiv:2106.14843* (2021). 3

[GCS02] Gooch B., Coombe G., Shirley P.: Artistic vision: painterly rendering using computer vision techniques. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering* (2002), pp. 83–ff. 2

[GEB15] Gatys L. A., Ecker A. S., Bethge M.: A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015). 1, 2

[GG01] Gooch B., Gooch A.: *Non-photorealistic rendering*. AK Peters/CRC Press, 2001. 1

[GKB*18] Ganin Y., Kulkarni T., Babuschkin I., Eslami S. A., Vinyals O.: Synthesizing programs for images using reinforced adversarial learning. In *International Conference on Machine Learning* (2018), PMLR, pp. 1666–1675. 2

[GLX*20] Gao C., Liu Q., Xu Q., Wang L., Liu J., Zou C.: Sketchycoco: Image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 5174–5183. 6

[GPAM*14] Goodfellow I., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y.: Generative adversarial nets. *Advances in neural information processing systems 27* (2014). 1

[HB17] Huang X., Belongie S.: Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 1501–1510. 2

[HE17] Ha D., Eck D.: A neural representation of sketch drawings. *arXiv preprint arXiv:1704.03477* (2017). 2

[Her03] Hertzmann A.: A survey of stroke-based rendering. Institute of Electrical and Electronics Engineers. 1, 2

[HHZ19] Huang Z., Heng W., Zhou S.: Learning to paint with model-based deep reinforcement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 8709–8718. 2, 4, 5, 6, 7, 8, 10

[IZZE17] Isola P., Zhu J.-Y., Zhou T., Efros A. A.: Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 1125–1134. 2

[JAFF16] Johnson J., Alahi A., Fei-Fei L.: Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision* (2016), Springer, pp. 694–711. 2

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 4, 5

[KLA19] Karras T., Laine S., Aila T.: A style-based generator architecture for generative adversarial networks. In *Proceedings of*

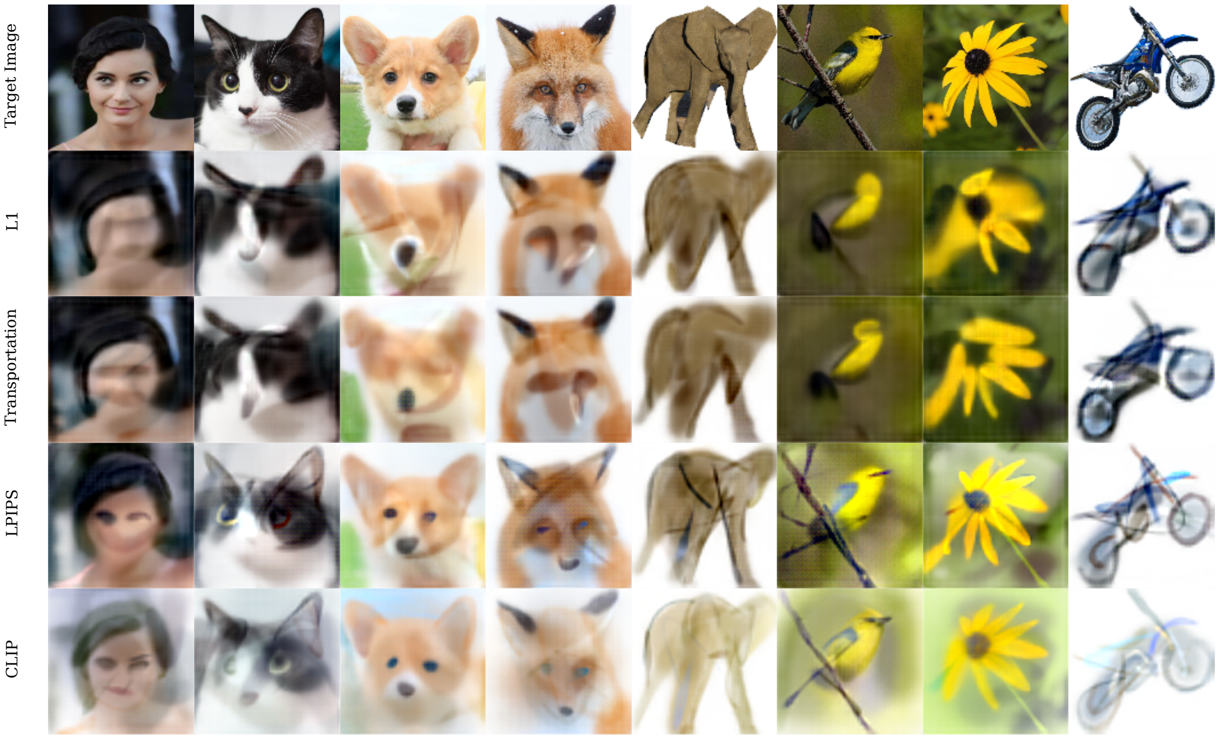


Figure 12: *Compare with different objective function (20 strokes). From top to bottom: target image, our method with pixel-wise loss (L1), our method with transportation loss (Transportation). our method with conventional perceptual loss (LPIPS [ZIE*18]), our method with the CLIP-based loss (CLIP).*

the IEEE/CVF conference on computer vision and pattern recognition (2019), pp. 4401–4410. 3

[KWHO21] KOTOVENKO D., WRIGHT M., HEIMBRECHT A., OMMER B.: Rethinking style transfer: From pixels to parameterized brushstrokes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 12196–12205. 2

[KY21] KIM G., YE J. C.: Diffusionclip: Text-guided image manipulation using diffusion models. *arXiv preprint arXiv:2110.02711* (2021). 3

[lc22] LIBMYPAINT CONTRIBUTORS.: libmypaint. https://github.com/mypaint/libmypaint, 2022. 4

[Lit97] LITWINOWICZ P.: Processing images and video for an impressionist effect. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 407–414. 2

[LLGRK20] LI T.-M., LUKÁČ M., GHARBI M., RAGAN-KELLEY J.: Differentiable vector graphics rasterization for editing and learning. *ACM Transactions on Graphics (TOG) 39*, 6 (2020), 1–15. 3

[LLH*21] LIU S., LIN T., HE D., LI F., DENG R., LI X., DING E., WANG H.: Paint transformer: Feed forward neural painting with stroke prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 6598–6607. 2, 4, 5, 6, 7, 10

[LLWT15] LIU Z., LUO P., WANG X., TANG X.: Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)* (December 2015). 6

[LSHG17] LI Y., SONG Y.-Z., HOSPEDALES T. M., GONG S.: Free-hand sketch synthesis with deformable stroke models. *International Journal of Computer Vision 122*, 1 (2017), 169–190. 2

[LZH*21] LI X., ZHANG S., HU J., CAO L., HONG X., MAO X., HUANG F., WU Y., JI R.: Image-to-image translation via hierarchical style disentanglement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 8639–8648. 2

[MBO14] MONROY A., BELL P., OMMER B.: Morphological analysis for investigating artistic images. *Image and Vision Computing 32*, 6-7 (2014), 414–423. 2

[MCO11] MONROY A., CARQUÉ B., OMMER B.: Reconstructing the drawing process of reproductions from medieval images. In *2011 18th IEEE International Conference on Image Processing* (2011), IEEE, pp. 2917–2920. 2

[MH21] MIHAI D., HARE J.: Differentiable drawing and sketching. *arXiv preprint arXiv:2103.16194* (2021). 2

[MPG*19] MELLOR J. F., PARK E., GANIN Y., BABUSCHKIN I., KULKARNI T., ROSENBAUM D., BALLARD A., WEBER T., VINYALS O., ESLAMI S.: Unsupervised doodling and painting with improved spiral. *arXiv preprint arXiv:1910.01007* (2019). 3

[NDR*21] NICHOL A., DHARIWAL P., RAMESH A., SHYAM P., MISHKIN P., MCGREW B., SUTSKEVER I., CHEN M.: Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741* (2021). 3

[NZ08] NILSBACK M.-E., ZISSERMAN A.: Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing* (Dec 2008). 6

[PLQC21] PANG Y., LIN J., QIN T., CHEN Z.: Image-to-image translation: Methods and applications. *IEEE Transactions on Multimedia* (2021). 2

[PWS*21] PATASHNIK O., WU Z., SHECHTMAN E., COHEN-OR D., LISCHINSKI D.: Styleclip: Text-driven manipulation of stylegan im-

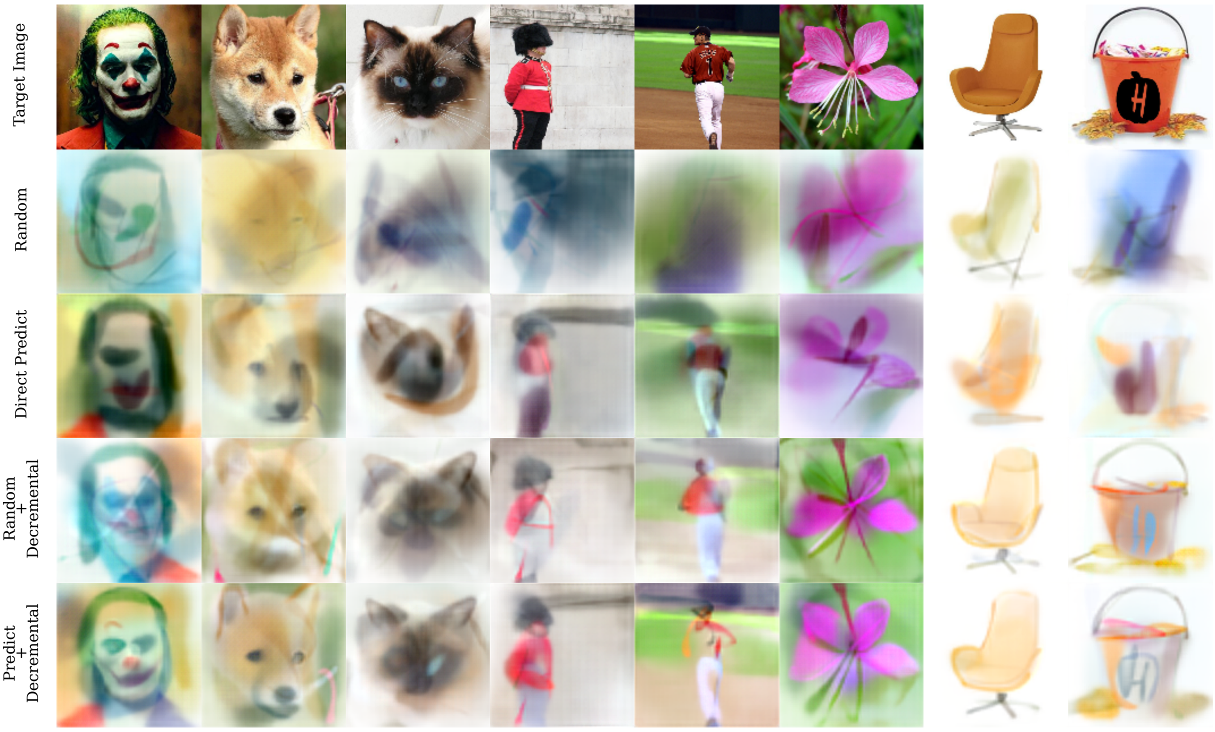


Figure 13: *Compare with different initialization and optimization strategy (20 strokes with CLIP-based loss). From top to bottom: target image, random initialize 20 strokes and optimize with regular Adam, direct predict 20 strokes and and optimize with regular Adam, random initialize 20 strokes and optimize with decremental strategy, predict 200 strokes and optimize with decremental strategy.*

agery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 2085–2094. 3

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., ET AL.: Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning* (2021), PMLR, pp. 8748–8763. 2

[SLO21] SCHALDENBRAND P., LIU Z., OH J.: Styleclipdraw: Coupling content and style in text-to-drawing synthesis. *arXiv preprint arXiv:2111.03133* (2021). 3

[SZ21] SINGH J., ZHENG L.: 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* (2021), pp. 16387–16396. 7

[TH21] TIAN Y., HA D.: Modern evolution strategies for creativity: Fitting concrete images and abstract concepts. *arXiv preprint arXiv:2109.08857* (2021). 3

[VPB*22] VINKER Y., PAJOUHESHGAR E., BO J. Y., BACHMANN R. C., HAIM BERMANO A., COHEN-OR D., ZAMIR A., SHAMIR A.: Clipasso: Semantically-aware object sketching. *arXiv e-prints* (2022), arXiv–2202. 2, 6, 7, 10

[WBW*11] WAH C., BRANSON S., WELINDER P., PERONA P., BELONGIE S.: *The Caltech-UCSD Birds-200-2011 Dataset*. Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011. 6

[YO12] YARLAGADDA P., OMMER B.: From meaningful contours to discriminative object shape. In *European Conference on Computer Vision* (2012), Springer, pp. 766–779. 2

[ZAFW21] ZHU P., ABDAL R., FEMIANI J., WONKA P.: Mind the gap: Domain gap control for single shot domain adaptation for generative adversarial networks. *arXiv preprint arXiv:2110.08398* (2021). 3

[ZFW*18] ZHOU T., FANG C., WANG Z., YANG J., KIM B., CHEN Z., BRANDT J., TERZOPOULOS D.: Learning to doodle with deep q networks and demonstrated strokes. In *British Machine Vision Conference* (2018), vol. 1, p. 4. 2

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 586–595. 5, 7, 11

[ZJH18] ZHENG N., JIANG Y., HUANG D.: Strokenet: A neural painting environment. In *International Conference on Learning Representations* (2018). 2

[ZPIE17] ZHU J.-Y., PARK T., ISOLA P., EFROS A. A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2223–2232. 2

[ZSQ*21] ZOU Z., SHI T., QIU S., YUAN Y., SHI Z.: Stylized neural painting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 15689–15698. 2, 6, 7, 8, 10

[ZZP*17] ZHU J.-Y., ZHANG R., PATHAK D., DARRELL T., EFROS A. A., WANG O., SHECHTMAN E.: Toward multimodal image-to-image translation. *Advances in neural information processing systems 30* (2017). 2