LPaintB: Learning to Paint from Self-Supervision

Biao Jia [1] Jonathan Brandt [2] Radomír Měch [2] Byungmoon Kim [2] Dinesh Manocha [1]

[1] University of Maryland at College Park, U.S.A
[2] Adobe Research, U.S.A

Abstract

We present a novel reinforcement learning-based natural media painting algorithm. Our goal is to reproduce a reference image using brush strokes and we encode the objective through observations. Our formulation takes into account that the distribution of the reward in the action space is sparse and training a reinforcement learning algorithm from scratch can be difficult. We present an approach that combines self-supervised learning and reinforcement learning to effectively transfer negative samples into positive ones and change the reward distribution. We demonstrate the benefits of our painting agent to reproduce reference images with brush strokes. The training phase takes about one hour and the runtime algorithm takes about 30 seconds on a GTX1080 GPU reproducing a 1000×800 image with 20,000 strokes. More details can be found at http://gamma.umd.edu/LPaintV.

1. Introduction

Digital painting systems are increasingly used by artists and content developers for various applications. One of the main goals has been to simulate popular or widely-used painting styles. With the development of non-photorealistic rendering techniques, including stroke-based rendering and painterly rendering [Her98, WS96], specially-designed or hand-engineered methods can increasingly simulate the painting process by applying heuristics. In practice, these algorithms can generate compelling results, but it is difficult to extend them to new or unseen styles.

Over the last decade, there has been considerable interest in using machine learning methods for digital painting. These methods include image synthesis algorithms based on convolutional neural networks, including modeling the brush [XHS12], generating brushstroke paintings [XZT*15], reconstructing paintings in specific styles [TDM*18], constructing stroke-based drawings [HE17], etc. Recent developments in generative adversarial networks [GPAM*14] and variational autoencoders [KW13] have led to the development of image generation algorithms that can be applied to painting styles [ZPIE17, ZFW*18, HYW18, KALL17, SLF*17].

One of the goals is to develop an automatic or intelligent painting agent that can develop its painting skills by imitating reference paintings. In this paper, we focus on building an intelligent painting agent that can reproduce a reference image in an identical or

Figure 1: *Results Generated by Our Painting Agent: We use three paintings (top row) as the reference images to test our novel self-supervised learning algorithm. Our trained agent automatically generates the digitally painted image (bottom row) of the corresponding column in about 30 seconds without the need of a paired dataset of human artists.*

transformed style with a sequence of painting actions. Unlike methods that directly synthesize images bypassing the painting process, we focus on a more general and challenging problem of training a painting agent from scratch using reinforcement learning methods. [XZT*15, XZS13, XHS12, ZFW*18] also use reinforcement

learning to solve the problem. All the methods encode goal states, which are usually defined as reference images, to the observations. This set-up is different from classic reinforcement learning tasks because, while the problem introduces an implicit objective to the policy network of reinforcement learning, the distribution of the reward in the action space can be very sparse and it makes training a reinforcement learning algorithm from scratch very difficult. To solve the problem, [XZT*15, XZS13, XHS12, ZFW*18] pre-train the policy network with a paired dataset consisting of images and corresponding actions defined in [XHS12]. However, it is very expensive to collect such a paired dataset of human artists and we need to explore other unsupervised learning methods.

Main Results: We present a reinforcement learning-based algorithm (LPaintB) that incorporates self-supervised learning to train a painting agent on a limited number of reference images without paired datasets. Our approach is data-driven and can be generalized by expanding the image datasets. Specifically, we adopt proximal policy optimization (PPO) [SWD*17] by encoding the current and goal states as observations and the continuous action space defined based on configurations of the paintbrush like length, orientation and brush size. The training component of our method only requires the reference paintings in the desired artistic style and does not require paired datasets collected by human artists. We use a self-supervised learning method to increase sampling efficiency. By replacing the goal state of an unsuccessful episode with its final state, we automatically generate a paired dataset with positive rewards. After applying the dataset to retrain the model using reinforcement learning, our approach can efficiently learn the optimal policy. The novel contributions of our work include:

- An approach for collecting supervised data for painting tasks by self-supervised learning.
- An adapted deep reinforcement learning network that can be trained using human expert data and self-supervised data, though we mostly rely on self-supervised data.
- An efficient rendering system that can automatically generate stroke-based paintings of desired resolutions by our trained painting agent.

We evaluate our approach by comparing our painting agent with prior painting agents that are trained from scratch by reinforcement learning [JFB*19]. We collect 1000 images with different color and patterns as the benchmark and compute L2 Loss between generated images and reference images. Our results show that self-supervised learning can efficiently collect paired data and can accelerate the training process. The training phase takes about 1 hour and the runtime algorithm takes about 30 seconds on a GTX 1080 GPU for high-resolution images.

2. Related Work

In this section, we give a brief overview of prior work on non-photorealistic rendering and the use of machine learning techniques for image synthesis.

2.1. Non-Photorealistic Rendering

Non-photorealistic rendering methods render a reference image as a combination of strokes by determining many properties like position, density, size, and color. To mimic the oil-painting process, Hertzmann [Her98] renders the reference image into primitive strokes using gradient-based features. To simulate mosaic decorative tile effects, Hauser [Hau01] segments the reference image using Centroidal Voronoi diagrams. Many algorithms have been proposed for specific artistic styles, such as stipple drawings [DHVOS00], pen-and-ink sketches [SABS94] and oil paintings [ZZXZ09] [LMPD15]. The drawback of non photo-realistic rendering methods is the lack of generalizability to new or unseen styles. Moreover, they may require hand-tuning and need to be extended to other styles.

2.2. Visual Generative Algorithms

Hertzmann et al. [HJO*01] introduce image analogies, a generative method based on a non-parametric texture model. Many recent approaches are based on CNNs and use large datasets of input-output training image pairs to learn the mapping function [GEB15]. Inspired by the idea of variational autoencoders [KW13], Johnson et al. [JAFF16] introduce the concept of perceptual loss to model the style transfer between paired dataset. Zhu et al. [ZPIE17] use generative adversarial networks to learn the mappings without paired training examples. These techniques have been used to generate natural images [KALL17, SLF*17], artistic images [LFY*17], and videos [VPT16, LFY*18]. Compared to previous visual generative methods, our approach can generate results of high resolution, can be applied to different painting media and is easy to extend to different painting media and artistic styles.

2.3. Image Synthesis Using Machine Learning

Many techniques have been proposed for image synthesis using machine learning. Hu et al. [HHX*18] present a framework using reinforcement learning and generative adversarial network to learn photo post-processing. Xie et al. [XHS12, XZT*15, XZS13] present a series of algorithms that simulate strokes using reinforcement learning and inverse reinforcement learning. These approaches learn a policy from either reward functions or expert demonstrations. In contrast to our algorithm, Xie et al. [XHS12, XZT*15, XZS13] focus on designing reward functions to generate orientational painting strokes. Moreover, their approach requires expert demonstrations for supervision. Ha et al. [HE17] collect a large-scale dataset of simple sketches of common objects with corresponding recordings of painting actions. Based on this dataset, a recurrent neural network model is trained in a supervised manner to encode and re-synthesize the action sequences. Moreover, the trained model is shown to be capable of generating new sketches. Following [HE17], Zhou et al. [ZFW*18] use reinforcement learning and imitation learning to reduce the amount of supervision needed to train such a sketch generation model. In contrast to prior methods, [JFB*19] operate in a continuous action space with higher dimensions applying PPO [SWD*17] reinforcement learning algorithm to train the agent from scratch. It can handle dense images with high resolutions. We use the same painting environment as [JFB*19] to demonstrate the benefits of our proposed learning algorithm. Although both algorithms do not need imitation data from human experts, self-supervised learning helps the reinforcement learning to converge to a better policy. Compared

Symbol	Meaning
t	step index
t_q	time steps to compute accumulated rewards
s_t	current painting state of step t, canvas
s^*	target painting state, reference image
$\hat{s^*}$	reproduction of s^*
o_t	observation of step t
a_t	action of step t
r_t	reward of step t
q_t	accumulated reward of step t
γ	discount factor for computing the reward
π	painting policy, predict a by o
V_π	value function of the painting policy, predict r by o
$f(s)$	feature extraction of state s
$Render(a_t, s_t)$	render function, render action to s_t
$Obs(s^*, s_t)$	observation function, encode the current state and the target state
$Loss(s, s^*)$	loss function, measuring distance between state s and objective state s^*

Table 1: *Notation and Symbols used in our Algorithm*

with prior visual generative methods, our painting agent can automatically generate results using a limited training dataset without paired dataset.

2.4. Reinforcement Learning

Reinforcement learning (RL) has achieved promising results recently in many problems, such as playing Atari games [MKS*13], the game of Go [SSS*17] and robot control [LFDA16]. A major focus of this effort has been to achieve improved time and data efficiency of the learning algorithms. Deep Q-Learning has been shown to be effective for tasks with discrete action spaces [MKS*13], and proximal policy optimization (PPO) [SWD*17] is currently regarded as one of the most effective for continuous action space tasks. Hindsight experience replay [AWR*17] enables off-policy reinforcement learning to sample efficiently from rewards which are sparse and binary. [AWR*17] can be seen as a sampling approach for off-policy algorithms, while we treat self-supervised learning and reinforcement learning as two components. Compared with [AWR*17], we present a practical approach to handle continuous space in a sparse reward setting and enhance the sampling efficiency by the self-supervised learning.

3. Self-Supervised Painting Agent

In this section, we introduce notations, formulate the problem and present our self-supervised learning algorithm for natural media painting.

3.1. Background

Self-supervised learning methods [KZB19] are designed to enable learning without explicit supervision. The supervised signal for a pretext task is created automatically. It is a form of unsupervised learning where the data itself provides supervision. In its original formulation, this process is performed by withholding part of the
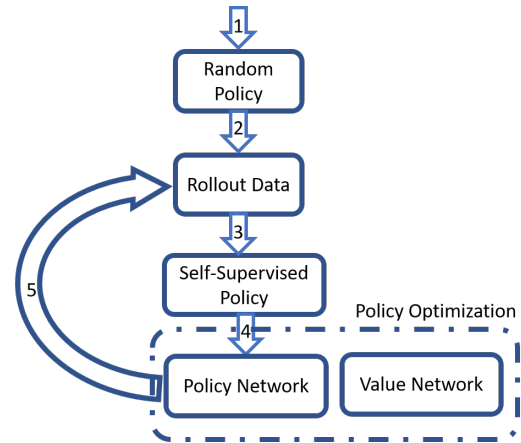

Figure 2: *Our Learning Algorithm: We use self-supervised learning to generate paired dataset using a training dataset with reference images only and initialize the model for reinforcement learning. Then we feed the trained policy network to self-supervised learning to generate the paired datasets with positive rewards. (1) We initialize the policy network with random painting actions; (2) We rollout the policy by iteratively applying to the policy network to the painting environment to get paired data, followed by assigning the goal state s^* as $\hat{s^*}$ and changing the rewards of each step accordingly; (3) We retrain the policy with the supervision data to generate the self-supervised policy, and use the behavior cloning to initialize the policy network; (4) We apply policy optimization [SWD*17] and update the policy; (5) We rollout the updated policy and continue the iterative algorithm.*

information of the data and training the classification or regression function to predict it. The required task usually has a definition of the proxy loss so that it can be solved by self-supervised learning. There are a variety of applications for self-supervised learning in different areas such as audio-visual analysis [OE18], visual representation learning [DGE15], image analysis [GSK18], robotics [JDVL18] etc. In this paper, we use the term self-supervised learning to refer to the process of generating self-supervision data and using the data to initialize the policy network of the reinforcement learning framework.

3.2. Problem Formulation

Reproducing images with brush strokes can be formalized as finding a series of actions that minimizes the distance between the reference image and the current canvas in the desired feature space. Based on notations in Table 1, this can be expressed as minimizing the loss function:

$$\pi^* = \arg\min Loss(\hat{s^*}, s^*) \tag{1}$$

After we apply reinforcement learning to solve the problem by defining $Reward()$ function, we can get:

$$\pi^* = \arg\max \sum_t^N Reward(a_t, s_t) \tag{2}$$

3.3. Painting Agent

In this section, we present the technical details of our reinforcement learning-based painting agent.

3.3.1. Observation

Our observation function is defined as follows. First, we encode the objective state (reference image) with the painting canvas. Second, we extract both the global and the egocentric view of the state. As mentioned in [ZFW*18, JFB*19], the egocentric view can encode the current position of the agent and it provides details about the state. The global view can provide overall information about the state. $o(s_i)$ is defined as Eq.(3), given the patch size (h_o, w_o) and the position of the brush position(h_p, w_p).

$$o(s_i) = \left\{ s_i \left[h_p - \frac{h_o}{2} : h_p + \frac{h_o}{2}, w_p - \frac{w_o}{2} : w_p + \frac{w_o}{2} \right], s_i \right.$$
$$\left. s_{ref} \left[h_p - \frac{h_o}{2} : h_p + \frac{h_o}{2}, w_p - \frac{w_o}{2} : w_p + \frac{w_o}{2} \right], s_{ref} \right\}. \tag{3}$$

3.3.2. Action

The action is defined as a vector in continuous space with positional information and paintbrush configurations. $a = \{dh, dw, width, color_R, color_G, color_B\} \in \mathbb{R}^6$. Each value is normalized to $[0, 1]$. The action is in a continuous space, which makes it possible to train the agent using policy gradient based reinforcement learning algorithms. The updated position of the paint brush after applying an action is computed by adding (dh, dw) to the coordinates of the paint brush $(p'_h, p'_w) = (p_h + dh, p_w + dw)$.

3.3.3. Loss Function

The loss function defines the distance between the current state and the objective state. It can guide how the agent reproduces the reference image. In practice, we test our algorithm with L_2 defined as Eq.(4), where s is the image of size $h \times w \times c$.

$$L_2(s, s^*) = \frac{\sum_{i=1}^h \sum_{j=1}^w \sum_{k=1}^c ||s_{ijk} - s^*_{ijk}||_2^2}{hwc} \tag{4}$$

For the self-supervised learning process, the loss function only affects reward computation. However, the reinforcement learning training process uses $\{\hat{s}^*\}$ as the reference images to train the model and the loss function can affect the policy network.

3.3.4. Policy Network

To define the structure of the policy network, we consider the input as a concatenated patch of the reference image and canvas $82 \times 82 \times 3$ in egocentric view and global view, given the sample size of $41 \times 41 \times 3$. The first hidden layer convolves 64 8×8 filters with stride 4, the second convolves 64 4×4 filters with stride 2 and the third layer convolves 64 3×3 filters with stride 1. After that, it connects to a fully-connected layer with 512 neurons. All layers use ReLU activation function [KSH12].For the training process, we add the criteria $r > 0$ to expedite the training process.

3.3.5. Runtime Algorithm

After we trained a model using self-supervised learning and reinforcement learning, we can apply the model to generate reference images with different resolutions. First, we randomly sample a position from the canvas and draw a patch with size (h_o, w_o) and feed it to the policy network. Second, we iteratively predict actions $a_t = \pi(o_t)$ and render them by environment until the value network V_π returns a negative reward. Then we reset the environment by sampling another position from the canvas and keep the loop until $Loss(\hat{s^*}, s^*)$ less than $Thresh_{sim}$.

3.4. Behavior Cloning

Behavior cloning uses a paired dataset with observations and corresponding actions to train the policy to imitate an expert trajectory or behaviors. In our setup, the expert trajectory is encoded in the paired dataset $\{o^{(t)}, a^{(t)}\}$ which is related to step 4 in Figure 2. We use behavior cloning to initialize the policy network of reinforcement learning with the supervised policy trained by paired data. The paired dataset can be generated by a human expert or an optimal algorithm with global knowledge, which our painting agent does not have. Once the paired dataset $\{o^{(t)}, a^{(t)}\}$ is obtained, one solution is to apply supervised learning based on regression or classification to train the policy. The trained process can be represented using an optimization formulation as:

$$\pi^* = \arg \min \sum_{t}^{N} ||\pi(o_t) - a_t||. \tag{5}$$

It is difficult to generate such an expert dataset for our painting application because of the large variation in the reference images and painting actions. However, we can generate a paired dataset by rolling out a policy defined as Eq.(6), which can be seen as iteratively applying predicted actions to the painting environment. For the painting problem, we can use the trained policy itself as the expert by introducing self-supervised learning.

3.5. Self-Supervised Learning

As we apply reinforcement learning to the painting problem, several new identities emerge as distinct from those of the classic controlling problems [SWD*17,SLA*15,MKS*13,SMSM00]. We use the reference image as the objective and encode it in the observation of the environment defined in Eq.(3). As a result, the objective of the task Eq.(5) is not explicitly defined. Hence the rollout actions on different reference images $\{s^*\}$ can vary.

Through the reinforcement learning training process, the positive rewards in the high dimensional action space can be very sparse. In other words, only a small portion of actions sampled by policy network have positive rewards. To change the reward distribution in the action space by increasing the probability of a positive reward, we propose using self-supervised learning. Our formulation uses the rollout of the policy as the paired data to train the policy network and retrains the model using reinforcement learning. Specifically, we replace the reference image s^* with the final rendering of the rollout of the policy function $\hat{s^*}$. Moreover, we use the updated observation $\{\hat{o}_t\}$ and the actions $\{a_t\}$ as the paired supervised training dataset. For the rollout process of the trained policy

Algorithm 1 Self-Supervised Learning

Require: Set of objective states $\{s^{*(i)}\}$, its size is n_s
Ensure: Painting Policy π and its value function V_π

1: **for** $i = 1, \cdots, n_s$ **do**
2:     $t = 0$
3:     $s_0 = \text{INITIALIZE}()$
4:     // Rollout the policy and collect the paired data with positive reward
5:     **while** $r \geq 0$ **do**
6:       $t = t + 1$
7:       $a_t = \pi(Obs(s_{t-1}, s^{*(i)}))$
8:       $s_t = Render(s_{t-1}, a_t)$
9:       $r = (Loss(s_{t-1}, s^{*(i)}) - Loss(s_t, s^{*(i)})))/Loss(s_0, s^{*(i)})$
10:    **end while**
11:    // Build self-supervised learning dataset
12:    **for** $j = 0, \cdots, t-1$ **do**
13:       $\hat{o}_j = Obs(s_j, s_t)$
14:       $\hat{r}_j = (Loss(s_{j+1}, s_t) - Loss(s_j, s_t))/Loss(s_0, s_t)$
15:    **end for**
16:    // Compute cumulative rewards
17:    **for** $j = 0, \cdots, t-1$ **do**
18:       $\hat{q}_j = \sum_{k=j}^{t-1} \gamma^{k-j} \hat{r}_k$
19:    **end for**
20:    $\pi = \text{UPDATE}(\pi, \{\hat{o}_{(j)}, a_{(j)}\})$ // Initialize policy network for policy optimization
21:    $V_\pi = \text{UPDATE}(V_\pi, \{\hat{o}_{(j)}, \hat{q}_{(j)}, a_{(j)}\})$ // Initialize value network for policy optimization
22: **end for**
23: **return** π, V_π

π, we have:

$$a_t = \pi(o_{t-1}), \tag{6}$$
$$s_t = Render(s_{t-1}, a_t), \tag{7}$$
$$r_t = \frac{Loss(s_{t-1}, s^*) - Loss(s_t, s^*)}{Loss(s_0, s^*)}, \tag{8}$$
$$o_t = Obs(s_t, s^*). \tag{9}$$

We can collect $\{o_{(t)}, a_{(t)}\}$ as the paired data. We denote the rendering of the final state as $\hat{s^*}$. The reward function is defined as the percentage improvement of the loss over the previous state.

Next, we modify o_t and r_t to a self-supervised representation as $\hat{o}_t$ and $\hat{r}_t$ as:

$$\hat{o}_t = Obs(s_t, \hat{s^*}), \tag{10}$$
$$\hat{r}_t = \frac{Loss(s_{t-1}, \hat{s^*}) - Loss(s_t, \hat{s^*})}{Loss(s_0, \hat{s^*})}, \tag{11}$$
$$\hat{q}_t = \sum_{k=t}^{t_s} \gamma^{k-t} r_k. \tag{12}$$

We use $\{\hat{o}_{(t)}, a_{(t)}, \hat{q}_{(t)}\}$ to train a self-supervised policy $\hat{\pi}$ and the value function $\hat{V}_\pi$. Algorithm 1 highlights the learning process for self-supervised learning.

Benchmarks	Benchmark1	Benchmark2
Reinforcement Learning Only	4.67	26.33
Self-supervised Learning Only	31.20	30.79
Our Combined Scheme	49.42	61.13

Table 2: Comparison of Different Training Schemes: *We evaluate our method by comparing the average cumulative rewards on the test dataset. Self-supervised learning only refers to a policy that is trained with rollouts of a random policy by supervised learning, which reference image s^* is replaced as the final rendering s_t.*

4. Implementation

Our painting environment is similar to that in [JFB*19], which is a simplified simulated painting environment. Our system can execute painting actions with parameters describing stroke size, color and positional information and updates the canvas accordingly. We use a vectorized environment [HRE*18] for a parallel training process, as shown in Figure 3, to train our model.

4.1. Performance

In practice, we use a 16 core CPU and a GTX 1080 GPU to train the model with a vectorized environment of dimension 16. We use SSPE [JFB*19] as $Render(a, s)$ to accelerate the training process. The learned policy can also be transferred to other simulated painting media like MyPaint or WetBrush [CKIW15] to get different visual effects and styles.

5. Results

In this section, we highlight the results and compare the performance with prior learning-based painting algorithms.

For the first experiment, we apply a critic condition to reward each step $r_t \geq 0$ for $t \geq 5$. Once the agent fails the condition, the environment will stop the rollout. We compare the cumulative reward $\sum_t r_t$ by feeding the same set of unseen images $\{s^{*(i)}\}$ to the environment. We use two benchmarks to test the generalization of the models. Benchmark1 is to reproduce an image $s^{*(i)}$ from a random image s_j. Benchmark2 is to reproduce an image $s^{*(i)}$ from a blank canvas. It can lead to a higher cumulative reward of Benchmark2 because the initial loss of Benchmark1 is less than Benchmark2. Each benchmark have 1000 $41 \times 41 \times 3$ patches. As shown in Table 2, our combined training scheme outperforms using only self-supervised learning or only reinforcement learning.

For the second experiment, we evaluate the performance on the high-resolution reference images. We compute the L_2 loss and cumulative rewards and compare our approach with [JFB*19]. We draw 1000 400×400 patches from 10 reference images to construct the benchmark. Moreover, we iteratively apply both the algorithms 1000 times to reproduce the reference images. We use the same training dataset with images to train the models. As shown in Table 3, our approach have a lower L_2, loss although both methods perform well in terms of cumulative rewards.
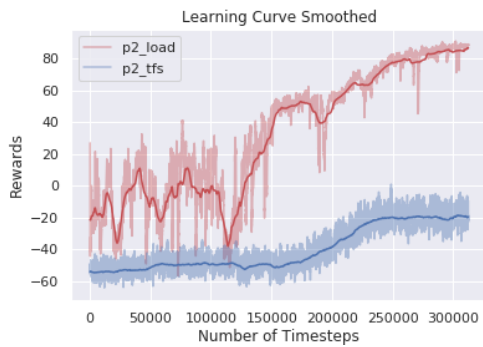
Figure 3: Learning Curve Comparison *We evaluate our algorithm by plotting the learning curve of training from scratch (blue) and training with self-supervised learning (red). As shown in the figure, the method with self-supervision have better convergence and performance.*

Approaches	Cumulative Rewards	L_2 Loss
PaintBot [JFB*19]	97.74	1920
LPaintB	98.25	1485

Table 3: Comparison with Previous Work *We evaluate our method by comparing the average cumulative reward and L_2 loss between final rendering and the reference image Eq.(4) on the test dataset.*

6. Conclusion

We present a novel approach for stroke-based image reproduction using self-supervised learning and reinforcement learning. Our approach is based on a feedback loop with reinforcement learning and self-supervised learning. We modify and reuse the rollout data of the previously trained policy network and feed it into the reinforcement learning framework. We compare our method with both the model trained with only self-supervised learning and the model trained from scratch by reinforcement learning. The result shows that our combination of self-supervised and reinforcement learning can greatly improve efficiency of sampling and performance of the policy.

One major limitation of our approach is that the generalization of the trained policy is highly dependent on the training data. Although reinforcement learning enables the policy to generalize to different states that supervised learning cannot address, the states still depend on the training data. Specifically, the distribution of generated supervision data is not close to the unseen data. Another limitation is the result generated by our method is not sharp enough, especially for the high contrast regions of the reference image. It can be improved by increasing either the total number of strokes or the resolutions of reference images but we still need a better definition of reward/loss to the problem.

For future work, we aim to enlarge the runtime steps and action space of the painting environment so that the data generated by self-supervised learning can be closer to the distribution of the unseen data. Our current setup includes most common stroke parameters like brush size, color, and position, but the painting parameters describing pen tilting, pen rotation, and pressure are not used. Moreover, we aim to develop a better definition of reward/loss to mitigate the blurry effects.
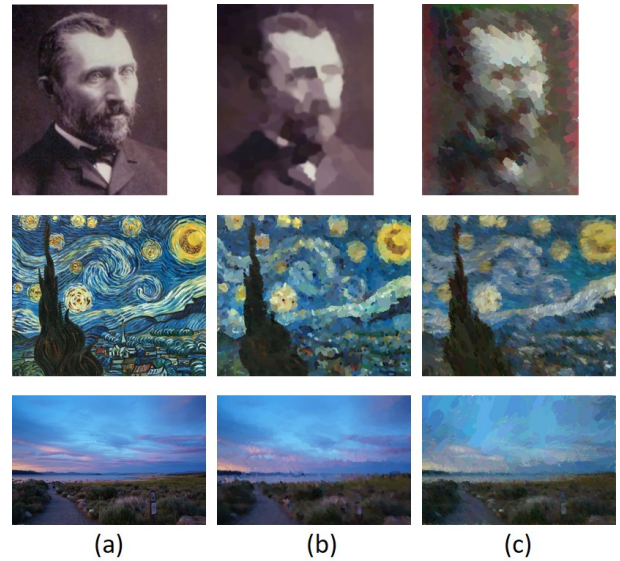

(a) (b) (c)

Figure 4: Our results compared with [JFB*19] *We compare the final rendering result using the same scale of the reference image and the same amount of painting actions. (a) are the reference images. (b) are generated by our painting agent (c) are generated by the agent [JFB*19]. We demonstrate the benefits of self-supervised learning by reference images with different resolutions. The training dataset for both algorithms consists of 374 $41 \times 41 \times 3$ patches sampling from one painting.*

References

[AWR*17] ANDRYCHOWICZ M., WOLSKI F., RAY A., SCHNEIDER J., FONG R., WELINDER P., MCGREW B., TOBIN J., ABBEEL O. P., ZAREMBA W.: Hindsight experience replay. In *Advances in Neural Information Processing Systems* (2017), pp. 5048–5058. 3

[CKIW15] CHEN Z., KIM B., ITO D., WANG H.: Wetbrush: Gpu-based 3d painting simulation at the bristle level. *ACM Transactions on Graphics (TOG) 34*, 6 (2015), 200. 5

[DGE15] DOERSCH C., GUPTA A., EFROS A. A.: Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1422–1430. 3

[DHVOS00] DEUSSEN O., HILLER S., VAN OVERVELD C., STROTHOTTE T.: Floating points: A method for computing stipple drawings. In *Computer Graphics Forum* (2000), vol. 19, Wiley Online Library, pp. 41–50. 2

[GEB15] GATYS L. A., ECKER A. S., BETHGE M.: A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576* (2015). 2

[GPAM*14] GOODFELLOW I., POUGET-ABADIE J., MIRZA M., XU B., WARDE-FARLEY D., OZAIR S., COURVILLE A., BENGIO Y.: Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680. 1

[GSK18] GIDARIS S., SINGH P., KOMODAKIS N.: Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728* (2018). 3

[Hau01] HAUSNER A.: Simulating decorative mosaics. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 573–580. 2

[HE17] HA D., ECK D.: A neural representation of sketch drawings. *CoRR abs/1704.03477* (2017). 1, 2, 3

[Her98] HERTZMANN A.: Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (1998), ACM, pp. 453–460. 1, 2

[HHX*18] HU Y., HE H., XU C., WANG B., LIN S.: Exposure: A white-box photo post-processing framework. *ACM Transactions on Graphics (TOG) 37*, 2 (2018), 26. 2

[HJO*01] HERTZMANN A., JACOBS C. E., OLIVER N., CURLESS B., SALESIN D. H.: Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (2001), ACM, pp. 327–340. 2

[HRE*18] HILL A., RAFFIN A., ERNESTUS M., GLEAVE A., TRAORE R., DHARIWAL P., HESSE C., KLIMOV O., NICHOL A., PLAPPERT M., RADFORD A., SCHULMAN J., SIDOR S., WU Y.: Stable baselines. https://github.com/hill-a/stable-baselines, 2018. 5

[HYW18] HUANG H., YU P. S., WANG C.: An introduction to image synthesis with generative adversarial nets. *CoRR abs/1803.04469* (2018). 1

[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

[JDVL18] JANG E., DEVIN C., VANHOUCKE V., LEVINE S.: Grasp2vec: Learning object representations from self-supervised grasping. *arXiv preprint arXiv:1811.06964* (2018). 3

[JFB*19] JIA B., FANG C., BRANDT J., KIM B., MANOCHA D.: Paintbot: A reinforcement learning approach for natural media painting. *arXiv preprint arXiv:1904.02201* (2019). 2, 4, 5, 6

[KALL17] KARRAS T., AILA T., LAINE S., LEHTINEN J.: Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196* (2017). 1, 2

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (2012), pp. 1097–1105. 4

[KW13] KINGMA D. P., WELLING M.: Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013). 1, 2

[KZB19] KOLESNIKOV A., ZHAI X., BEYER L.: Revisiting self-supervised visual representation learning. *arXiv preprint arXiv:1901.09005* (2019). 3

[LFDA16] LEVINE S., FINN C., DARRELL T., ABBEEL P.: End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research 17*, 1 (2016), 1334–1373. 3

[LFY*17] LI Y., FANG C., YANG J., WANG Z., LU X., YANG M.-H.: Universal style transfer via feature transforms. In *Advances in Neural Information Processing Systems* (2017), pp. 386–396. 2

[LFY*18] LI Y., FANG C., YANG J., WANG Z., LU X., YANG M.-H.: Flow-grounded spatial-temporal video prediction from still images. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 600–615. 2

[LMPD15] LINDEMEIER T., METZNER J., POLLAK L., DEUSSEN O.: Hardware-based non-photorealistic rendering using a painting robot. In *Computer graphics forum* (2015), vol. 34, Wiley Online Library, pp. 311–323. 2

[MKS*13] MNIH V., KAVUKCUOGLU K., SILVER D., GRAVES A., ANTONOGLOU I., WIERSTRA D., RIEDMILLER M.: Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013). 3, 4

[OE18] OWENS A., EFROS A. A.: Audio-visual scene analysis with self-supervised multisensory features. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 631–648. 3

[SABS94] SALISBURY M. P., ANDERSON S. E., BARZEL R., SALESIN D. H.: Interactive pen-and-ink illustration. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques* (1994), ACM, pp. 101–108. 2

[SLA*15] SCHULMAN J., LEVINE S., ABBEEL P., JORDAN M., MORITZ P.: Trust region policy optimization. In *International Conference on Machine Learning* (2015), pp. 1889–1897. 4

[SLF*17] SANGKLOY P., LU J., FANG C., YU F., HAYS J.: Scribbler: Controlling deep image synthesis with sketch and color. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), vol. 2. 1, 2

[SMSM00] SUTTON R. S., MCALLESTER D. A., SINGH S. P., MANSOUR Y.: Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems* (2000), pp. 1057–1063. 4

[SSS*17] SILVER D., SCHRITTWIESER J., SIMONYAN K., ANTONOGLOU I., HUANG A., GUEZ A., HUBERT T., BAKER L., LAI M., BOLTON A., ET AL.: Mastering the game of go without human knowledge. *Nature 550*, 7676 (2017), 354. 3

[SWD*17] SCHULMAN J., WOLSKI F., DHARIWAL P., RADFORD A., KLIMOV O.: Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017). 2, 3, 4

[TDM*18] TANG F., DONG W., MENG Y., MEI X., HUANG F., ZHANG X., DEUSSEN O.: Animated construction of chinese brush paintings. *IEEE transactions on visualization and computer graphics 24*, 12 (2018), 3019–3031. 1, 3

[VPT16] VONDRICK C., PIRSIAVASH H., TORRALBA A.: Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems* (2016), pp. 613–621. 2

[WS96] WINKENBACH G., SALESIN D. H.: Rendering parametric surfaces in pen and ink. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM, pp. 469–476. 1

[XHS12] XIE N., HACHIYA H., SUGIYAMA M.: Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting. *CoRR abs/1206.4634* (2012). 1, 2, 3

[XZS13] XIE N., ZHAO T., SUGIYAMA M.: Personal style learning in sumi-e stroke-based rendering by inverse reinforcement learning. *Information Processing Society of Japan* (2013). 1, 2

[XZT*15] XIE N., ZHAO T., TIAN F., ZHANG X. H., SUGIYAM M.: Stroke-based stylization learning and rendering with inverse reinforcement learning. *IJCAI* (2015). 1, 2, 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. *British Machine Vision Conference* (2018). 1, 2, 4

[ZJH19] ZHENG N., JIANG Y., HUANG D.: Strokenet: A neural painting environment. In *International Conference on Learning Representations* (2019). 3

[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. 1, 2

[ZZXZ09] ZENG K., ZHAO M., XIONG C., ZHU S. C.: From image parsing to painterly rendering. *ACM Trans. Graph. 29*, 1 (2009), 2–1. 2