Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

Data-Driven Ink Painting Brushstroke Rendering

Koki Madono[1], Edgar Simo-Serra[1]

[1]Waseda University, Japana

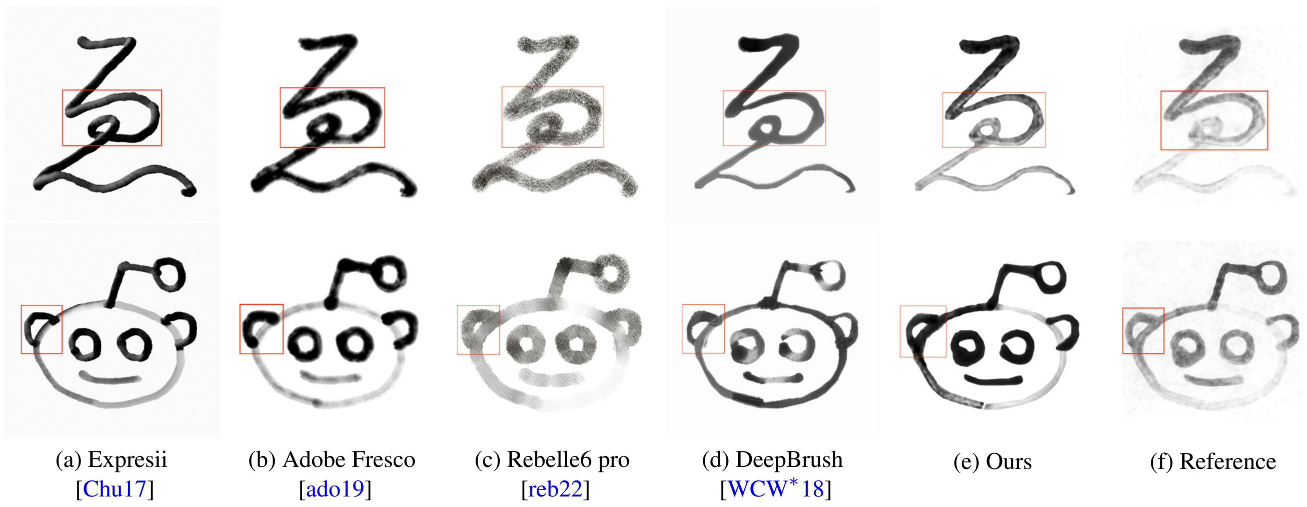

Figure 1: ***Example of ink painting rendering results.*** *We compare our approach with existing methods and a reference real painting. Our approach is able to reproduce real fading and blotting effects similar to the reference ink paintings while capturing texture variations. Important differences are highlighted in red.*

Abstract

Although digital painting has advanced much in recent years, there is still a significant divide between physically drawn paintings and purely digitally drawn paintings. These differences arise due to the physical interactions between the brush, ink, and paper, which are hard to emulate in the digital domain. Most ink painting approaches have focused on either using heuristics or physical simulation to attempt to bridge the gap between digital and analog, however, these approaches are still unable to capture the diversity of painting effects, such as ink fading or blotting, found in the real world. In this work, we propose a data-driven approach to generate ink paintings based on a semi-automatically collected high-quality real-world ink painting dataset. We use a multi-camera robot-based setup to automatically create a diversity of ink paintings, which allows for capturing the entire process in high resolution, including capturing detailed brush motions and drawing results. To ensure high-quality capture of the painting process, we calibrate the setup and perform occlusion-aware blending to capture all the strokes in high resolution in a robust and efficient way. Using our new dataset, we propose a recursive deep learning-based model to reproduce the ink paintings stroke by stroke while capturing complex ink painting effects such as bleeding and mixing. Our results corroborate the fidelity of the proposed approach to real hand-drawn ink paintings in comparison with existing approaches. We hope the availability of our dataset will encourage new research on digital realistic ink painting techniques.

CCS Concepts
• ***Computing methodologies*** $\rightarrow$ *Non-photorealistic rendering; Neural networks;*

1. Introduction

Ink painting is a traditional artwork with many different forms of expression. Although ink painting only uses black ink on white paper, the paintings can exhibit a surprising amount of complex hand motions, bleeding effects, and subtle control of tone gradients. Through careful use of whitespace and the nuances arising from the interaction between ink, water, and paper, experienced painters are able to create masterpieces that evoke the imagination of the

© 2023 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

viewer. Present works [Xue20, BDGH*20] combine heuristics and physical simulation to express real-world effects. To fill the gap between digital and real-world painting, such approaches provide computational parameters such as ink color, water amount, and brush type. However, these approaches rely on the mathematical formulation to render the ink painting, which in many cases can not capture the full variability of interactions found in the real world, such as the mixing of strokes, fading of the ink, or blotting. Instead of focusing on an increasingly complex mathematical expression of ink, water, and paper interactions, we take a data-driven approach with the proposed dataset and model to render realistic ink paintings.

To provide viable ink painting data amenable for training data-driven approaches, we employ a multi-camera robot-based setup to automatically create a diversity of ink paintings while capturing the entire process in high resolution. In order to capture the diversity of ink paintings, we use different brushes, papers, and ink tones by adjusting the ratio of water to ink. The dataset is split by strokes and contains a rich amount of meta-data relating to the brush, ink, and strokes.

We also propose a model to render realistic ink painting strokes, consisting of a recursive convolutional network, which takes stroke embeddings as an input generated from stroke parameters, and outputs realistic ink painting strokes. Using an explicit canvas, our model is able to draw complex drawings recursively while showing complex ink painting effects such as blotting or ink fading. We use our new dataset to train a model to reproduce the real strokes and our results corroborate the fidelity of our approach in comparison with existing approaches. Figure 1 shows an example of our proposed approach in comparison with existing methods.

In summary, we present:

- An ink painting dataset providing real ink paintings with rich metadata for stroke.
- A recursive convolutional network framework to generate ink paintings.
- Significant improvements over the state of the art in ink painting quality.

We plan to make the dataset available and hope the availability will encourage new advances in neural rendering for realistic ink painting.

2. Related Work

2.1. Ink Painting Frameworks

Given the proliferation of digital software and the cost associated with analog painting materials, the popularity of digital ink painting frameworks has been increasing in recent years. However, the reproduction of different physical effects in the digital medium is a non-trivial task and has led to significant research in the field.

In the early stages of ink painting research, many approaches focused on mathematical approximation techniques. One such family of approaches is the mathematical modeling of brushstrokes. Curtis [CAS*97] use the Kubelka-Munk compositing model for simulating ink painting effects, and Xu *et al.* [XYW12] employs a simple brushstroke model and renders the stroke movements via a shader for real-time applications. Xie *et al.* [XHS13] also modeled the brushstrokes and their movement while resorting to a reinforcement learning paradigm for the stroke generation. While the previous approaches are based on heuristics, other approaches focus on trying to model the fluid dynamics when ink painting. In this direction, the lattice Boltzmann equations [Suc01] have been proposed as a simple fluid dynamics simulation method to apply fluid dynamics to other tasks. In the ink painting community, Yu *et al.* [YMLS03] proposed a viscous flow using the lattice Boltzmann equations, which was later extended by MoXi [CT05] to be more physically accurate. While these approaches significantly improved the quality of rendering at the time, they were unable to model more complicated bleeding effects that can be seen in real paintings.

More recent paint rendering systems consolidate the best parts of the earlier research to create more user-friendly and higher-quality software. WetBrush [CKIW15] is a famous painting simulation framework that simulates brush movements and renders the ink particles. Although this software is mainly focused on oil painting, their new Eulerian-Lagrangian approach enables the simulation of more complex ink paintings simulations using GPUs. Expresii [Chu17] is an example of a popular research-based commercial watercolor rendering software. This software mainly extends MoXi [CT05] to use 3D brush information [CT04] while ensuring the simulation is applicable to real-time interaction in a user-friendly package. Adobe Fresco [ado19] is a similar framework to Expresii that can make a wide variety of picture effects. Unfortunately, unlike Expresii, the details of the rendering techniques used are not made open to the public. However, experimental results show that Adobe Fresco can also simulate essential ink painting effects such as bleeding effect, change at the color crossing, and water flow. Rebelle [reb22] is also a painting framework that can make an ink painting expression with the effects. Rebelle has a similar rendering quality to Adobe Fresco, and some technical artists prefer using this framework to draw the ink painting. Compared with Expresii, Adobe Fresco and Rebelle can adjust the size of brushes and layers in more detail. However, all aforementioned approaches are based on heuristics and physical simulation like older approaches and have trouble rendering some advanced interactions that arise in ink painting, which are not trivial to model mathematically.

While these mathematical and physically-based approaches are able to achieve good rendering quality, they are unable to capture the full range of effects that arise in ink painting. Instead of explicitly modeling physical interactions, we adopt a data-driven approach in which we train a machine learning model to reproduce real ink painting effects, allowing for much higher quality and natural painting effects.

Another line of research has focused on learning painting effects from data. Wu *et al.* [WCW*18] proposes a pix2pix-based image rendering framework that renders the oil painting using a simulation dataset. To train the model, they use the simulator's parameters such as pressure, ink height, time trajectories, and corresponding color information. Lu *et al.* [LBDF13] also proposes an algorithm that is based on data to simulate the natural painting media. This approach handles various paintings such as oil paintings, watercolor paintings, pencils, pastels, etc. More recently, Shugrina *et al.* [SLF22] proposed a Neural Brushstroke Engine that combines deep generative models with interactive drawing tools in a data-driven setting. In

Subset name	Paper Type	Brush Type	Strokes	Unique Vector Images
Brush-paired	Practice	Small / Medium / Large	462	8
Paper-paired	Practice / Fine	Tiny / Small / Medium / Large	1210	42
Backlight-paired	Practice	-	92	46
Unpaired	Fine	Medium	146	13

Table 1: ***Overview of our Ink Painting dataset.*** *The paired subsets repeat the same drawing varying a single parameter,* e.g.*, the brush-paired dataset will use the same paper and vector image but vary the brush to generate 3 variations of the image for each of the small, medium, and large brushes.*

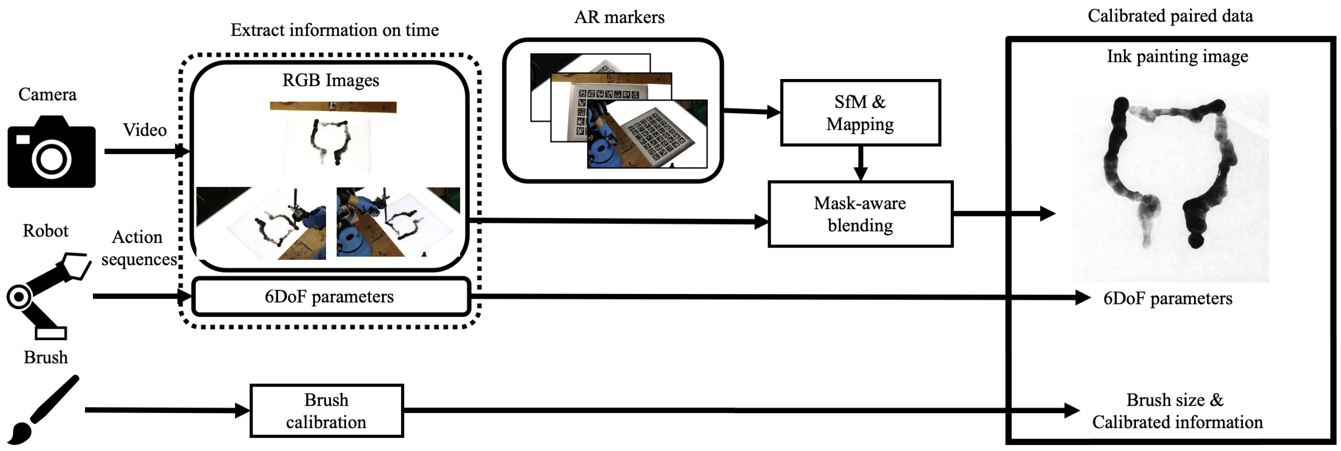

Figure 2: ***Overview of our dataset creation approach.*** *We use a 3-camera system with Structure from Motion (SfM) techniques to reconstruct the drawn ink painting. AR markers and offline calibration are used to get an accurate position information on the brush and paper, while occlusion-aware blending with a backlight removes the effect of the robot arm and shadows during the capturing process.*

particular, they adopt a patch-based rendering system in order to draw continuous strokes. Although they use real-world data in the training phase, their focus is on developing interactive drawing tools rather than accurate reproduction of complex painting effects. Our approach follows previous data-driven approaches and we manually collect a dataset of real-world ink painting strokes that we use to train a neural network to reproduce. We focus on the ink painting task and design a framework that is able to reproduce complex ink painting effects such as fading or blotting, which existing approaches have difficulty reproducing.

2.2. Image-to-Image Translation

Image-to-Image (I2I) translation aims at converting an input image from a source domain to a target domain by preserving the important details of the the input image and adopting the style to the target domain. I2I can be applied to the broad tasks such as style transfer [ZPIE17, KCK*17, KLA19], image inpainting [PKD*16, ZZP*17, SYL*18, ISSI17, LHM*19], and image colorization [ISSI16, ZZI*17, SSV17, HCL*18, ZHL*19, XWF*20].

Stroke stylization is a popular example of interactive I2I for drawing. Liu *et al.* [LFHK21] propose a 3D line drawing approach based on the surface and path geometry to render stroke thickness and displacement accurately. Lijie *et al.* [YXDW19] rendered strokes from the user's rough strokes and selected the optimal pattern for texture synthesis. Ming *et al.* [ZMGS17] focused on rendering the 3D painting stroke with stroke coherence.

More recently, I2I has been applied to ink painting tasks as well. Sketch-And-Paint GAN [Xue20] was proposed to make an ink painting from generated edge maps, and ChipGAN [HGM*18] can convert real images to the style of ink paintings. While these approaches already have significant rendering quality, they focus on the conditional generation of images and can not be used easily in an interactive setting. On the other hand, our approach focuses on reproducing individual strokes, which is amenable to interactive usage scenarios.

2.3. Robot Painting

Robot painting has been developed to automatically draw certain pictures such as facial portraits [SKT*20], avatar images [WTZ*20], and sketches [GZY*20]. Furthermore, accurate robot drawing techniques [LWKH20, VKN20] are also being developed.

Recently, robot painting has been applied to data collection by simulating a realistic painting. Wang *et al.* [WCD*20] proposed to simulate calligraphy by collecting data with a robotic arm. Bidgoli *et al.* [BDGH*20] collect the human painting with motion capture, then simulate the human drawing using a robot arm. Wang *et al.* [WCD*20] only capture the final result of calligraphy whereas Bidgoli *et al.* [BDGH*20] only captured single brush strokes.

Unlike the previous works, our proposed ink painting dataset contains sequential information, occlusion masks, and brush stroke metadata to allow for more diverse applications of ink painting.

3. RealBrush Ink Painting Dataset

We create a new ink painting drawing dataset for high-quality stroke painting based on a robot arm. Multiple cameras are used to obtain high-resolution images of the painting without occlusions, which are amenable for training high-quality data-driven ink painting models.

3.1. Dataset Overview

Our dataset consists of 1,818 strokes with associated rich metadata collected with a diversity of brushes and paper quality , which are extracted from video recordings of robot drawings. We summarize the dataset details in Table 1. In particular, four types of brushes (tiny, small, medium, and large) and two different types of paper (practice and fine) are used. The dataset is split into several subsets depending on where all aspects are kept fixed except one used to create variations of the same data. In particular, three paired subsets are created for paper, brush, and illumination techniques used.

Data is collected using a multi-view system with high-resolution video cameras and is drawn with a robotic arm. The multiple cameras allow for an occlusion-free accurate reconstruction of the drawing canvas, and by using a robotic arm we are able to have high reproducibility to collect paired data and have accurate correspondences with the reference vector data. While the robot-drawn images are less fluid and natural looking than human drawings, we find that they can already be used to train high-performance ink-painting models as explained in the next section.

In addition to the captured data, we manually annotated all the paths and segmentation masks for each stroke. The paths allow computing accurate distance calculations, that jointly with the masks, allow our proposed model to learn to generate more natural strokes.

3.2. Data Collection

The focus of our data collection approach is to fuse the multiple views to generate the resulting ink painting without occlusions, shadows, or other artifacts. We do this by first calibrating the setup offline, then we predict the occlusions from discrepancies in the views, before filtering noise with morphological operators, and finally blending the images together to obtain the final fused ink painting image. Figure 2 shows an overview of our proposed data collection approach.

3.2.1. Offline Calibration

We first employ incremental Structure-from-Motion (SfM) [Wu13, Ull79] to estimate the mapping from the camera image to world coordinates. We optimize for both extrinsic and intrinsic camera parameters by minimizing the mean squared projection error of AR markers.

3.2.2. Image Fusion

With the pre-computed calibration, we perform mask-aware blending to obtain an occlusion-free ink painting image. First, we project the camera images using the calibration parameters to obtain $\{\boldsymbol{I}_1, \boldsymbol{I}_2, \boldsymbol{I}_3, \}$ Then, we calculate the binary difference map $\boldsymbol{A}$ with:

$$\boldsymbol{A}_{i,j} = \boldsymbol{A}_{j,i} = \left[|\boldsymbol{I}_i - \boldsymbol{I}_j| < \tau \right] , \tag{1}$$

where $\boldsymbol{A}_{i,j}$ denotes a difference map between i-th image and j-th image, $|\cdot|$ denotes the absolute value, and τ is a threshold to determine the pixel difference. In this work, we set $\tau = 50$ when using 8-bit RGB values considering the light and camera noise.

We hypothesize that the occluded area should be overlapped with another difference map, and using the difference maps, we compute the occlusion maps $\boldsymbol{M}_i$ with:

$$\boldsymbol{M}_i = \left(1 - \boldsymbol{A}_{i,j}\right) \odot \left(1 - \boldsymbol{A}_{i,k}\right), \tag{2}$$

where $\boldsymbol{M}_i$ denotes i-th image's occlusion map, $\odot$ indicates element-wise multiplication, and both $\boldsymbol{A}_{i,j}$ and $\boldsymbol{A}_{i,k}$ denote a difference map against j-th, k-th image. We find our approach is more robust than training a segmentation model or other data-driven approach. For more accurate occlusion mask generation, we also employ erosion and dilation morphological operators [Dou18] to $\boldsymbol{M}$. In particular, we first apply a 60-pixel erosion followed by a 60-pixel dilation.

We finally blend the visible areas of the projected images $\{\boldsymbol{I}_1, \boldsymbol{I}_2, \boldsymbol{I}_3\}$ using refined masks $\{\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3\}$ as follows:

$$\boldsymbol{I}'_{i+1} = \boldsymbol{I}_i \odot (1 - \boldsymbol{M}_i) \odot \hat{\boldsymbol{M}}_i \tag{3}$$

$$\hat{\boldsymbol{M}}_{i+1} = \hat{\boldsymbol{M}}_i \odot \boldsymbol{M}_i \tag{4}$$

$\hat{\boldsymbol{M}}_i$ denotes an accumulated mask that is used for filling the occluded area and we initialize $\hat{\boldsymbol{M}}_0 = \mathbf{1}$.

Finally, $\boldsymbol{I}'_4$ is used as the final reconstructed image of the ink painting. This resulting image is the composition of the visible areas of all the cameras which are blended in the order front image $\boldsymbol{I}_1$, left image $\boldsymbol{I}_2$, and right image $\boldsymbol{I}_3$.

4. Data-Driven Ink Painting

In this section, we propose a data-driven model that is capable of ink painting, closely mimicking real-world ink painting effects. Our model consists of three components: stroke embedding to capture stroke direction and fading, ink painter to draw the strokes, and lighting correction to improve visual quality.

4.1. Stroke Embedding

Our painter model consists of an image-to-image translation model, that converts rasterized vector stroke paths into realistic ink brush strokes. One of the issues with the rasterization of vector strokes is that the stroke direction information is lost, which limits the quality of the results. We overcome this by converting the vector strokes into a stroke embedding, where the stroke direction information is preserved, allowing for richer and nuances painting effects.

In order to capture the drawing direction and fading effect, we embed the distance from the starting position of a stroke with:

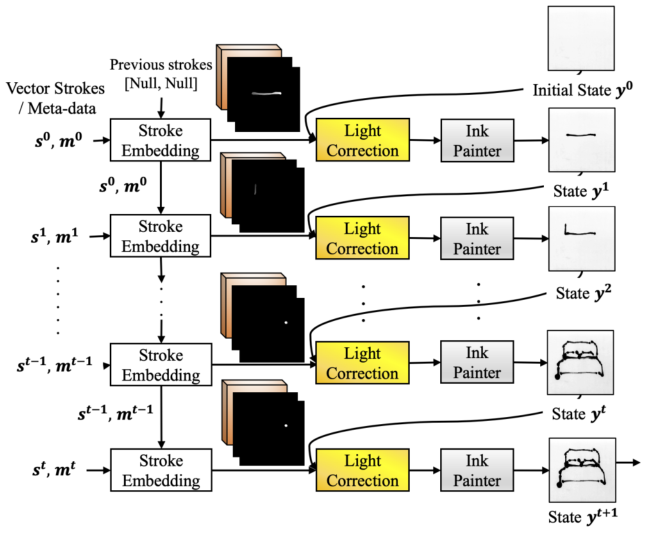

Figure 3: ***An overview of our proposed data-driven ink painting framework:*** *We begin with an empty canvas, then for each stroke, we use a stroke embedding to rasterize it and feed it to an ink painting model, that combines it with the previous canvas state. Finally, a lighting correction model can be used to tweak the final look of the ink painting.*

$$s(u,v) = f(u,v)\, g(u,v)\ , \tag{5}$$

where s denotes the stroke embedding, f denotes the stroke mask function, g denotes the fading distance function, and (u,v) indicate the 2D coordinates of a pixel.

The stroke mask function f is defined as follows:

$$f(u,v) = \begin{cases} 1, & \text{if } (u,v) \in \boldsymbol{S} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where $\boldsymbol{S}$ denotes an stroke region.

The fading distance function g is defined as follows:

$$g(u,v) = \max\left(1 - \frac{1}{\alpha} h(u,v) - \beta_i,\ \theta\right) \tag{7}$$

where the α is a fading ratio, β_i s the accumulated distance of the i-th brush stroke, h is a distance function, and θ denotes the minimum value to ensure that the embedding is completely disappear.

As a distance function h we use the Euclidean distance between the current position (u_N, v_N) and the stroke starting point (u_0, v_0). Specifically, the distance of a point is calculated by N points sampled at equal intervals from the start point to the current point, and the sum of the distances is used as the distance. We define the distance function h as follows:

$$h(u,v) = \sum_{i=0}^{N} \sqrt{(u_{i+1} - u_i)^2 + (v_{i+1} - v_i)^2} \tag{8}$$

Given that the amount of liquid contained in a brush monotonically decreases, we compute the accumulated distance of i-th brush stroke as:

$$\beta_{i+1} = \begin{cases} \beta_i + \max\left(1 - g(u,v)\right), & \text{if } N > 1 \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

where N denotes the number of times the brush has been used since it was updated. Each time the brush is dipped in ink, we set $\beta_0 = 0$.

4.2. Lighting Correction

In order to minimize the effects of shadows when constructing the dataset, we employ a backlight, which has the unfortunate side effect of changing the the overall lighting of the output. In order to correct this, we use a small training dataset of backlit and non-backlit images to train an image-to-image translation model that can create a natural ink painting look.

We first reshape the image to the pixel sequences $\boldsymbol{P}$ and correct the pixel value by a nonlinear mapping model. We can then define the lighting correction process as:

$$\boldsymbol{p}' = MLP(\boldsymbol{p}) \tag{10}$$

where $\boldsymbol{p}$ is the original pixel, $\boldsymbol{p}'$ is the light corrected pixel, MLP is a multi-perceptron model. We model with MLP by two linear layers. The first layer maps the one-dimensional pixel to the latent space. Then, we reverse the latent space to the one-dimensional pixel at the next linear layer. For the layer, we use a hyperbolic tangent activation function (tanh). Furthermore, we do not use both batch normalization and activation functions on the output of the first layer since these degrade the light-corrected results and the network itself is a shallow network. When the optimizing phase, we use Adam optimizer with a learning rate of $1e-3$. For the data, the user can use backlight-paired images for the training.

We train the light correction model by minimizing the gap between the light-corrected image to the non-backlight images. In particular, we employ the weighted L1 loss as the training objective function:

$$L(\boldsymbol{p}', \boldsymbol{p}^\star) = \left|(\boldsymbol{p}' - \boldsymbol{p}^\star) \odot \left(1 + \gamma(1 - \boldsymbol{p}^\star)\right)\right| \tag{11}$$

where γ is a weighting hyper-parameter that controls how much importance is given shadow over the ink painting paper. Specifically, a value of $\gamma = 0$ makes the loss behave like an L1 loss, while a value of $\gamma = 1$ makes the dark regions have twice the influence of the light regions. The reweighting compensates for the bias toward white pixels in the training data. $\boldsymbol{p}^\star$ is the light-corrected image which is from our backlight-paired dataset. After the training of the light correction model, we freeze the parameter and use the pre-processing of the ink state.

4.3. Ink Painter Architecture

Our ink painter model consists of an image-to-image translation model that converts the stroke embedding and a canvas state to an ink painting stroke. Our model consists of an encoder-decoder fully convolutional architecture that is applied recursively stroke-by-stroke to update an initially blank canvas state, which consists of a 2D grayscale image representation of the paper on which is being drawn. An adversarial training scheme that makes use of an auxiliary discriminator is used to improve the quality of the results.

Advanced ink painting techniques include wet-on-wet painting, where water or diluted ink is first applied to the paper, before drawing on top to create complex bleeding effects. To be able to represent these effects, we recursively draw to a canvas, allowing the model to learn to render complex bleeding. We note that our stroke embedding consists of 0 on non-drawn areas and close-to-1 values on drawn areas which allows us to use zero padding in the convolutional layers without introducing border artifacts.

Our stroke embedding consists of a 2D grayscale image with additional metadata that encodes the stroke shape and direction. In particular, as metadata, currently, the type of paper and brush hair size are currently used. We update the painting canvas in the inference phase recursively with:

$$\boldsymbol{y}^{t+1} = \max\left(P\left(\boldsymbol{y}^{t}, \boldsymbol{s}^{t}, \boldsymbol{m}^{t}, \boldsymbol{s}^{t-1}, \boldsymbol{m}^{t-1}\right), \boldsymbol{y}^{t}\right) \tag{12}$$

where $\boldsymbol{y}^{t+1}$ is the new canvas state, P is our ink painter model, $\boldsymbol{y}^{t}$ is the canvas state at time step t, $\boldsymbol{s}^{t}$ is the stroke embedding of the t-th stroke, and $\boldsymbol{m}^{t}$ is additional meta-data regarding the paper and brush. The max operator is used under the assumption drawing is purely additive, *i.e.*, ink is not able to disappear. The previous stroke and metadata are aimed at enabling ink bleeding effects by providing detailed information about paper, brush, and the current and previously drawn strokes. In the case of consecutive strokes, information from the previous stroke is used, which helps improve blotting and blending effects. For more fine-grained control, the user can also choose to edit the stroke embedding $\boldsymbol{s}$. Furthermore, the model uses blank input in place of the two parameters $\boldsymbol{s}^{t-1}$, $\boldsymbol{m}^{t-1}$ if the user ignores the past rendering information on the present time step.

We model P with an encoder-decoder convolutional network that consists of 21 layers. First, the input is processed by a 7×7 convolutional layer with reflection padding, then it is reduced in size four times with downsampling layers, then self-attention is applied, followed by 9 convolutional layers, another layer of self-attention, and finally, four upsampling layers and a 7×7 convolutional output a brush stroke at the same resolution as the input image. All convolutional layers, except the first and last, use 3×3 pixel kernels with 1 pixel 0-padding. Every layer uses batch normalization and ReLU activation function, except the last layer which uses a hyperbolic tangent activation function (tanh). In the case of downsampling, a stride of 2 is used, while upsampling is implemented by bilinear upsampling followed by a convolutional layer. Please see the supplemental materials for full details of the model architecture.

4.3.1. Training Details

We train our ink painter model using an L_1 loss and adversarial loss L_{adv} for single-time steps or individual strokes. In particular, we can write the training loss as:

$$L_{adv} + \lambda \left| y^{t+1} - P_i(\boldsymbol{s}^{t}, \boldsymbol{m}^{t}, \boldsymbol{s}^{t-1}, \boldsymbol{m}^{t-1}) \right|_1 \tag{13}$$

where L_{adv} is the standard adversarial loss and λ is a weighting hyper-parameter on L_1 loss which we set to $\lambda = 10$ in all experiments.

For L_1 loss, we also use identity mapping constraint that if the input stroke $\boldsymbol{m}^{t}$ is none, the model will produce the present time step's state $\boldsymbol{s}^{t}$. When training, we prepare original $\boldsymbol{m}^{t}$ and blank version of $\boldsymbol{m}_b^{t}$ and compute the loss with same λ as $\lambda \left| y^{t} - P_i(\boldsymbol{s}^{t}, \boldsymbol{m}_b^{t}, \boldsymbol{s}^{t-1}, \boldsymbol{m}^{t-1}) \right|_1$.

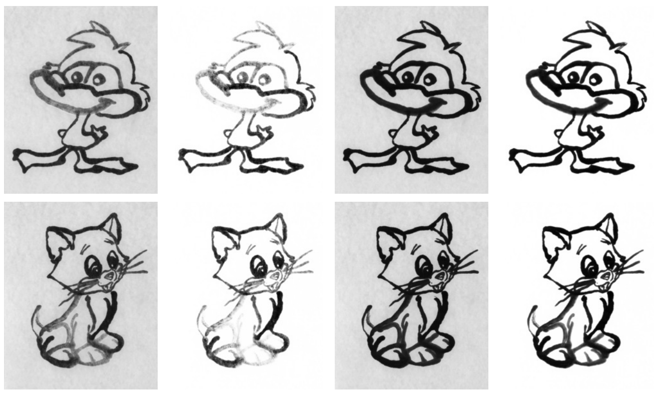

Figure 4: ***Visualization of the light correction.*** *The light correction compensates for the fact that the paper of training data is backlit, allowing for the rendered ink paintings to have a more natural look.*

5. Experiments

5.1. Experimental Setting

We train our model with the ADAM optimizer [KB14] for 200 epochs with a learning rate of 0.0002, weight parameter $\lambda = 5$, and momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$ and select the model with best quality. During inference, we initialize an initial state $\boldsymbol{y}^0 = 0.075$ based on the paper color statics.

We compare with the state-of-the-art ink painting commercial software Expresii [CT05, Chu17], Adobe Fresco [ado19], Rebelle6 pro [reb22], and DeepBrush [WCW*18]. Expressi is based on real-time fluid simulation of the interaction between the ink and the paper, while the exact details of the implementation of Adobe Fresco is unknown. Rebelle6 pro is a similar framework to Adobe Fresco that expresses complex textures with a variety of tools. DeepBrush [WCW*18] is a data-driven rendering model that is trained on simulated data. For a fair comparison, we perform an in-depth analysis of both comparison approaches to choose the best combination of brush and parameters to represent the reference images, and adjust the brush color and size manually for each stroke. In particular, we use brush parameters of the Wash Soft Live watercolor brush (Adobe Fresco), Large-short brush (Expresii), and round brush (Rebelle6 pro), which our analysis shows give the best results on our dataset. Full details on the brush and parameter selection are shown in the supplementary materials.

Each reference image used in the evaluation is traced to obtain vector lines which are used to generate the input for all approaches while making sure the stroke order is the same as the reference image.

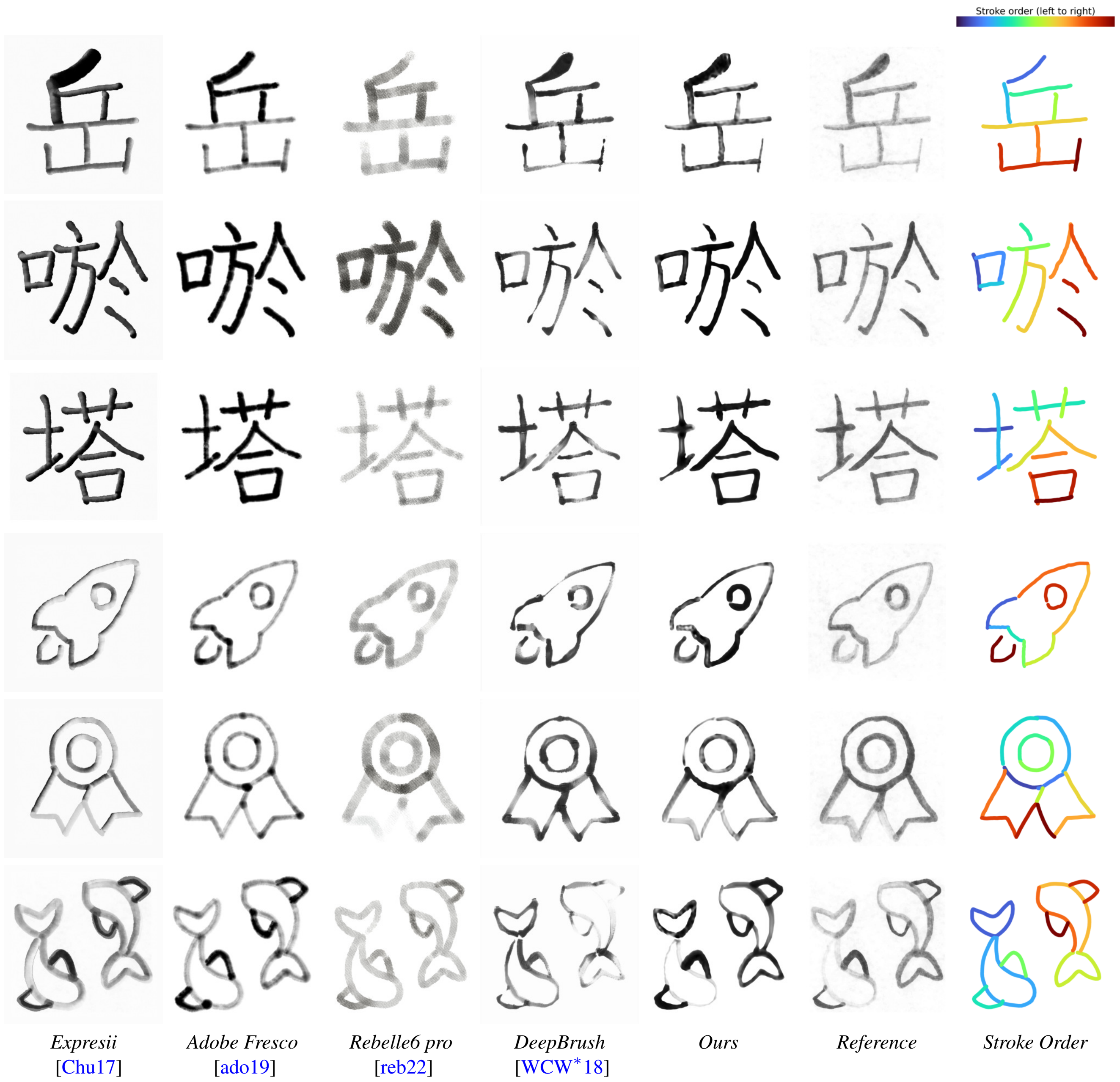

Figure 5: ***Qualitative ink painting results for different settings.***. *The proposed approach consistently shows a similar style to the artist-created references. See the supplementary material for additional examples.*

For the DeepBrush model, we used our proposed dataset and preprocessed it with light correction for a fair comparison.

5.2. Quantitative Comparison

We compare Expresii, Adobe Fresco, and Rebelle6 pro using 21 test images with 148 strokes including the results shown in Fig. 1 and Fig. 5. For each image we compute the LPIPS [ZIE*18] at 1024 $\times$ 1024 pixel resolution with VGG network, which captures differences between shape and texture between the images. We use SSIM and MSE to objectively evaluate the image quality. Numerical results are summarized in Table 2. We can see that the results significantly favor our approach over existing approaches, indicating that it is able to better capture all the different ink painting effects. Specifically, for the models that use our proposed dataset (DeepBrush and ours), LPIPS and SSIM show better scores compared with the commercial software systems, which more accurately capture structural and perceptual differences. Furthermore, we can see that, even when

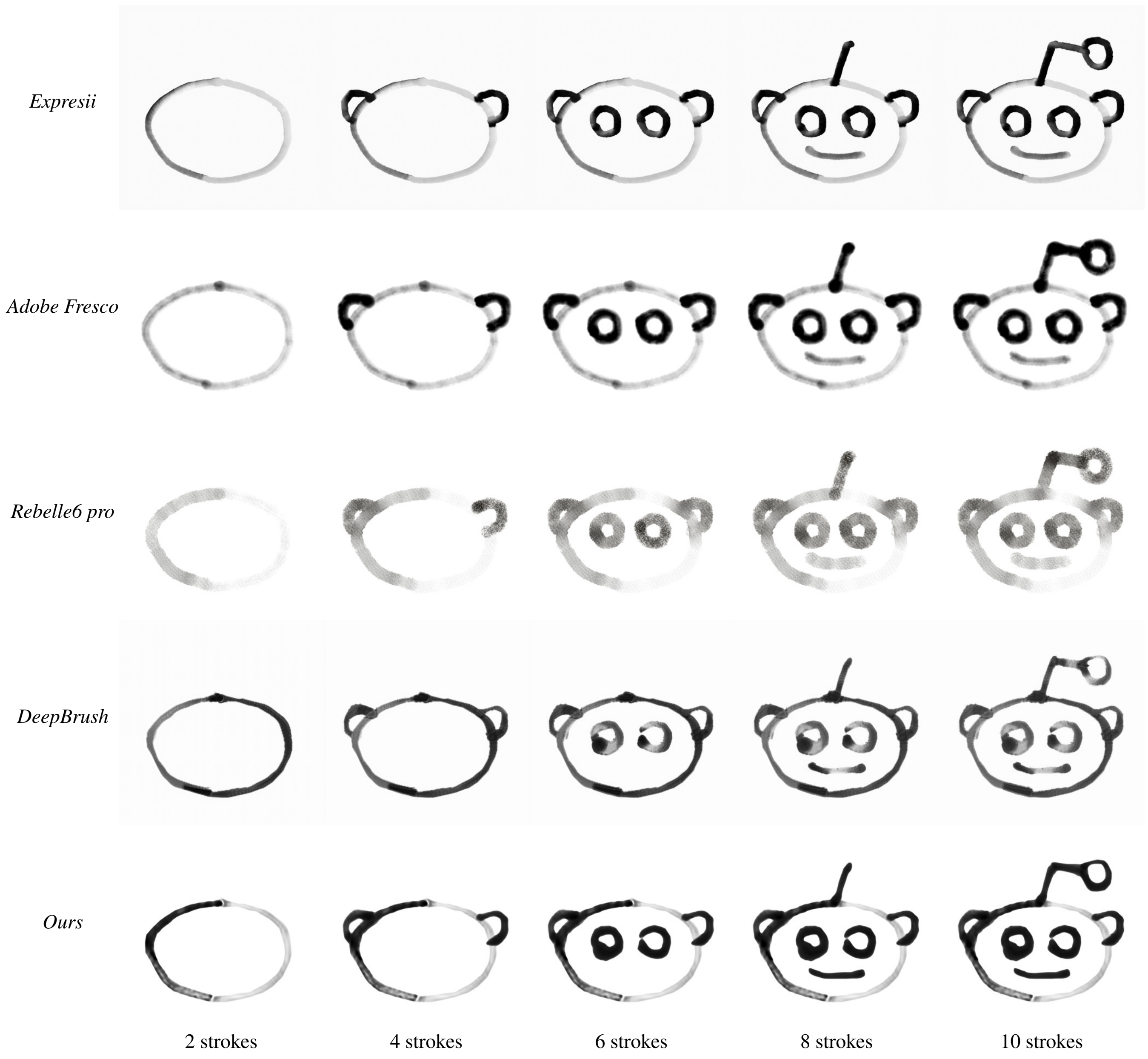

Figure 6: ***Stroke-by-stroke break down.*** *We show the painting process for an example image.*

trained on the same dataset, our approach is able to represent the ink paintings.

5.3. Qualitative Comparison

We compare with existing approaches using a diversity of different inputs and show some example results in Fig. 1 and Fig. 5. Although all approaches use brushes designed for ink painting, we can see that Expresii fails to correctly blend overlapping strokes, Adobe Fresco is unable to represent complex textures, and Rebelle6 pro produces dissimilar textures with reference images. Furthermore, Expresii, Adobe Fresco, and Rebelle6 pro fail to capture the strokes fading as ink decreases. DeepBrush can produce more accurate results on some rendering results. However, their model fails to capture the fading effects on many of the strokes, and thus failing to capture the important global effects of ink paintings.

By leveraging our dataset and model improvements, our approach is able to capture complex stroke interactions and obtain the most plausible results when compared with the reference human-drawn images. In particular, the character in Fig. 1 is composed of a single long stroke with overlapping parts where it can be seen that the existing approaches fail to capture the subtle complexity of the real

Table 2: ***Quantitative comparison.*** *Evaluation with LPIPS [ZIE*18] is done over 21 images (1024 × 1024 px) with 148 strokes. The best values are highlighted in* ***bold****. DeepBrush and Ours are trained with the proposed dataset while the other approaches are based on their own simulator's rendering system.*

	Expresii	Fresco	Rebelle6	DeepBrush	Ours
LPIPS ↓	0.3413	0.2824	0.2898	0.2691	**0.2593**
SSIM ↑	0.8617	0.8768	0.8518	0.9011	**0.9070**
MSE ↓	0.0364	0.0236	**0.0110**	0.0146	0.0134

drawing, while our approach is able to significantly outperform them.

We also show a breakdown of the drawing process in Fig. 6. The characters in Fig. 6 are composed of several line intersections. Our approach gives the most plausible results at the stroke-by-stroke level whereas compared approaches fail to reproduce the physical phenomenon, *e.g.*, line intersection or color fading, and is able to reproduce the drawing process most accurately.

5.4. Lighting Correction Analysis

We visualize the effect of lighting compensation in Fig. 4. Our light correction model can handle shadows on paper in both backlit and non-backlit settings. The light correction scheme has an important role in correcting artifacts from the data collection process introduced during training with our dataset.

6. Limitations and Discussion

We have presented a dataset and an approach for interactively drawing realistic ink paintings. Our dataset is able to capture accurate high-resolution ink paintings in the reproducible manner with a robot arm. This data allows our model to learn the complex interactions between strokes and paper, and experiments show that our results are much more convincing than existing approaches. We believe our model is an important first step in data-driven ink painting and hope that the release of our dataset will stimulate research in this field as there are still open problems as discussed below.

Although the proposed approach significantly improves rendering results by simulating complex interactions between ink and paper, it still struggles with more difficult cases of wet-on-wet painting, that is, painting on areas of the paper that is not dry and susceptible to blotting. We show such a challenging example in Fig. 7, where existing approaches struggle to accurately represent blotting. Although the results of our approach are still significantly better than existing methods, there is still significant room for improvement when compared with the reference image. To improve such results, a more sophisticated recursive model capable of capturing wetness to understand blotting may be necessary.

Additionally, our proposed CNN updating scheme, uses the the canvas in its entirety as input. While this can limit the resolution size, ink paintings can have complex interactions with paper and water, affecting areas much larger than just the stroke, and patch-based approaches are limited in what parts of the image can be updated in each iteration. Given that there is a trade-off between rendering quality and computational complexity, merging patch-based and image-based approaches is left as future work and is out of the scope of this paper.

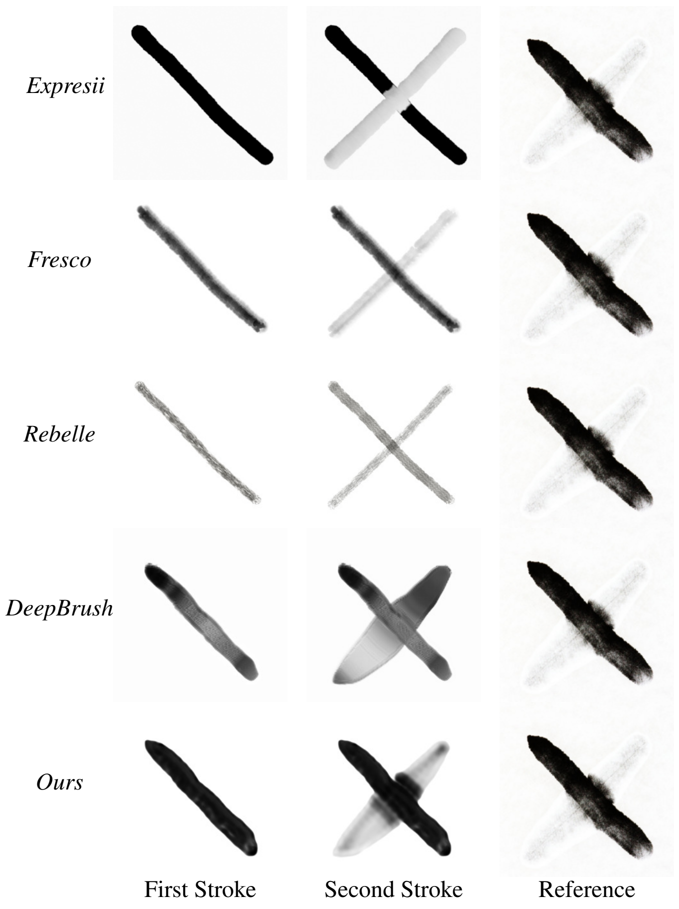

Figure 7: ***Limitations.*** *Complex wet-on-wet interactions such as drawing a dark stroke across a wet area are still an open challenge for existing approaches. Reference images are real ink painting effects.*

In addition to rendering improvements, utilizing the metadata (e.g., 6DoF parameter, amount of ink, etc.) is one of the directions to improve the approach. The proposed approach is suitable for real-time usage, however, given the stroke-based nature of our approach, the current interface is not optimal for user experience (e.g., faster computational speed. real-time usage). The focus of this work is proposing an efficient and high-quality data-driven approach to reproduce complex real-world effects, which are not possible with other existing approaches. Extending the proposed approach to be more usable and improving the user interface is out of the scope of the current work and we believe is an exciting future direction.

References

[ado19] Adobe Fresco. https://www.adobe.com/jp/

products/fresco.html, 2019. 1, 2, 6, 7

[BDGH*20] Bidgoli A., De Guevara M. L., Hsiung C., Oh J., Kang E.: Artistic style in robotic painting; a machine learning approach to learning brushstroke from human artists. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)* (2020), IEEE, pp. 412–418. 2, 3

[CAS*97] Curtis C. J., Anderson S. E., Seims J. E., Fleischer K. W., Salesin D. H.: Computer-generated watercolor. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (1997), pp. 421–430. 2

[Chu17] Chu N. S.: Expresii watercolor. In *ACM SIGGRAPH 2017 Appy Hour*. 2017, pp. 1–2. 1, 2, 6, 7

[CKIW15] Chen Z., Kim B., Ito D., Wang H.: Wetbrush: Gpu-based 3d painting simulation at the bristle level. *ACM Trans. Graph. 34*, 6 (oct 2015). 2

[CT04] Chu N. S., Tai C.-L.: Real-time painting with an expressive virtual chinese brush. *IEEE Computer Graphics and applications 24*, 5 (2004), 76–85. 2

[CT05] Chu N. S.-H., Tai C.-L.: Moxi: Real-time ink dispersion in absorbent paper. *ACM Trans. Graph. 24*, 3 (jul 2005), 504–511. 2, 6

[Dou18] Dougherty E.: *Mathematical morphology in image processing*, vol. 1. CRC press, 2018. 4

[GZY*20] Gao F., Zhu J., Yu Z., Li P., Wang T.: Making robots draw a vivid portrait in two minutes. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 9585–9591. 3

[HCL*18] He M., Chen D., Liao J., Sander P. V., Yuan L.: Deep exemplar-based colorization. *ACM Transactions on Graphics (TOG) 37*, 4 (2018), 1–16. 3

[HGM*18] He B., Gao F., Ma D., Shi B., Duan L.-Y.: Chipgan: A generative adversarial network for chinese ink wash painting style transfer. In *Proceedings of the 26th ACM International Conference on Multimedia* (New York, NY, USA, 2018), MM '18, Association for Computing Machinery, p. 1172–1180. 3

[ISSI16] Iizuka S., Simo-Serra E., Ishikawa H.: Let there be color! joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics (ToG) 35*, 4 (2016), 1–11. 3

[ISSI17] Iizuka S., Simo-Serra E., Ishikawa H.: Globally and locally consistent image completion. *ACM Transactions on Graphics (ToG) 36*, 4 (2017), 1–14. 3

[KB14] Kingma D. P., Ba J.: Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014). 6

[KCK*17] Kim T., Cha M., Kim H., Lee J. K., Kim J.: Learning to discover cross-domain relations with generative adversarial networks. In *International conference on machine learning* (2017), PMLR, pp. 1857–1865. 3

[KLA19] Karras T., Laine S., Aila T.: A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 4401–4410. 3

[LBDF13] Lu J., Barnes C., DiVerdi S., Finkelstein A.: Realbrush: Painting with examples of physical media. *ACM Trans. Graph. 32*, 4 (jul 2013). 2

[LFHK21] Liu D., Fisher M., Hertzmann A., Kalogerakis E.: Neural strokes: Stylized line drawing of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 14204–14213. 3

[LHM*19] Liu M.-Y., Huang X., Mallya A., Karras T., Aila T., Lehtinen J., Kautz J.: Few-shot unsupervised image-to-image translation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 10551–10560. 3

[LWKH20] Liu R., Wan W., Koyama K., Harada K.: Multi-pen robust robotic 3d drawing using closed-loop planning. *arXiv preprint arXiv:2009.14501* (2020). 3

[PKD*16] Pathak D., Krahenbuhl P., Donahue J., Darrell T., Efros A. A.: Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2536–2544. 3

[reb22] Rebelle 6. https://www.escapemotions.com/products/rebelle/about?//products/rebelle/index.php, 2022. 1, 2, 6, 7

[SKT*20] Singhal A., Kumar A., Thukral S., Raina D., Kumar S.: Chitrakar: Robotic system for drawing jordan curve of facial portrait. *arXiv preprint arXiv:2011.10781* (2020). 3

[SLF22] Shugrina M., Li C.-Y., Fidler S.: Neural brushstroke engine: Learning a latent style space of interactive drawing tools. *ACM Transactions on Graphics (TOG) 41*, 6 (2022). 2

[SSV17] Suárez P. L., Sappa A. D., Vintimilla B. X.: Infrared image colorization based on a triplet dcgan architecture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (2017), pp. 18–23. 3

[Suc01] Succi S.: *The lattice Boltzmann equation: for fluid dynamics and beyond*. Oxford university press, 2001. 2

[SYL*18] Song Y., Yang C., Lin Z., Liu X., Huang Q., Li H., Kuo C.-C. J.: Contextual-based image inpainting: Infer, match, and translate. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 3–19. 3

[Ull79] Ullman S.: The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences 203*, 1153 (1979), 405–426. 4

[VKN20] Venkataramaiyer R. B., Kumar S., Namboodiri V. P.: Can i teach a robot to replicate a line art. In *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2020), IEEE, pp. 1922–1930. 3

[WCD*20] Wang S., Chen J., Deng X., Hutchinson S., Dellaert F.: Robot calligraphy using pseudospectral optimal control in conjunction with a novel dynamic brush model. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2020), IEEE, pp. 6696–6703. 3

[WCW*18] Wu R., Chen Z., Wang Z., Yang J., Marschner S.: Brush stroke synthesis with a generative adversarial network driven by physically based simulation. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering* (2018), pp. 1–10. 1, 2, 6, 7

[WTZ*20] Wang T., Toh W. Q., Zhang H., Sui X., Li S., Liu Y., Jing W.: Robocodraw: robotic avatar drawing with gan-based style transfer and time-efficient path optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 10402–10409. 3

[Wu13] Wu C.: Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision - 3DV 2013* (2013), pp. 127–134. 4

[XHS13] Xie N., Hachiya H., Sugiyama M.: 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. 2

[Xue20] Xue A.: End-to-end chinese landscape painting creation using generative adversarial networks, 2020. 2, 3

[XWF*20] Xu Z., Wang T., Fang F., Sheng Y., Zhang G.: Stylization-based architecture for fast deep exemplar colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 9363–9372. 3

[XYW12] Xu T.-C., Yang L.-J., Wu E.-H.: Stroke-based real-time ink wash painting style rendering for geometric models. In *SIGGRAPH Asia 2012 Technical Briefs* (New York, NY, USA, 2012), SA '12, Association for Computing Machinery. 2

[YMLS03] YU D., MEI R., LUO L.-S., SHYY W.: Viscous flow computations with the method of lattice boltzmann equation. *Progress in Aerospace sciences 39*, 5 (2003), 329–367. 2

[YXDW19] YANG L., XU T., DU J., WU E.: Easy drawing: Generation of artistic chinese flower painting by stroke-based stylization. *IEEE Access 7* (2019), 35449–35456. 3

[ZHL*19] ZHANG B., HE M., LIAO J., SANDER P. V., YUAN L., BERMAK A., CHEN D.: Deep exemplar-based video colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 8052–8061. 3

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018). 7, 9

[ZMGS17] ZHENG M., MILLIEZ A., GROSS M., SUMNER R. W.: Example-based brushes for coherent stylized renderings. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (2017), pp. 1–10. 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. 3

[ZZI*17] ZHANG R., ZHU J.-Y., ISOLA P., GENG X., LIN A. S., YU T., EFROS A. A.: Real-time user-guided image colorization with learned deep priors. *arXiv preprint arXiv:1705.02999* (2017). 3

[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). 3