2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Sim-to-Real Robotic Sketching using Behavior Cloning and Reinforcement Learning

Biao Jia[1], Dinesh Manocha[2]

***Abstract*— Robotic sketching in real-world scenarios poses a challenging problem with diverse applications in art, robotics, and digital design. We present a novel approach that bridges the gap between digital and robotic sketching, leveraging behavior cloning and reinforcement learning techniques. This paper introduces an approach aimed at bringing the gap between simulated and real-world robotic sketching closer together through the integration of behavior cloning and reinforcement learning techniques. Our approach trains painting policies that operate effectively in both virtual environments and real-world robotic sketching systems. We have implemented a robotic sketching system featuring an UltraArm robot equipped with a RealSense D415 camera, closely emulating the MyPaint virtual environment. Our system can perceive its environment and adapt painting policies to natural painting media. Our results highlight the effectiveness of our agent in terms of acquiring policies for high-dimensional continuous action spaces, enabling the seamless transfer of brush manipulation techniques from simulation to practical robotic sketching. Furthermore, we demonstrate our robotic sketching system's capability to generate complex images and strokes using various configurations. https://sites.google.com/view/sketchingrobot**

I. Introduction

Painting, a diverse and complex art form, spans various styles from watercolors to oil portraits. Efforts to simulate these styles have used non-photorealistic rendering techniques [1], [2] with some success. In generative image models conditioned on text, advancements have enabled diverse image synthesis [3], [4], yet transferring these techniques to real robots remains a challenge. Machine learning has been applied to painting, including brush modeling [5], stroke-based drawings [6], and artistic style emulation [7]. However, existing approaches often rely on manual engineering or lack stroke-based methods. Our work bridges these gaps in the realm of robotic sketching systems.

Distinct from these efforts, a category of robotic sketching systems has emerged that often translates stroke-based methods directly into real robotic sketching [8]. Other studies, like those by Chen et al. [9], El et al. [10], and Vempati et al. [11], have concentrated on learning low-level manipulation policies to address challenges posed by uneven painting surfaces.

Our approach tackles a broader and more intricate challenge: training robotic painting policies using reinforcement learning techniques, behavior cloning, and stroke modeling.

[1]Biao Jia is with the Department of Computer Science, University of Maryland at College Park, MD, 20740. biao@umd.edu

[2]Dinesh Manocha is with the Departments of Computer Science and Electrical & Computer Engineering, University of Maryland at College Park, MD, 20740. dmanocha@umd.edu

Project website: https://sites.google.com/view/sketchingrobot

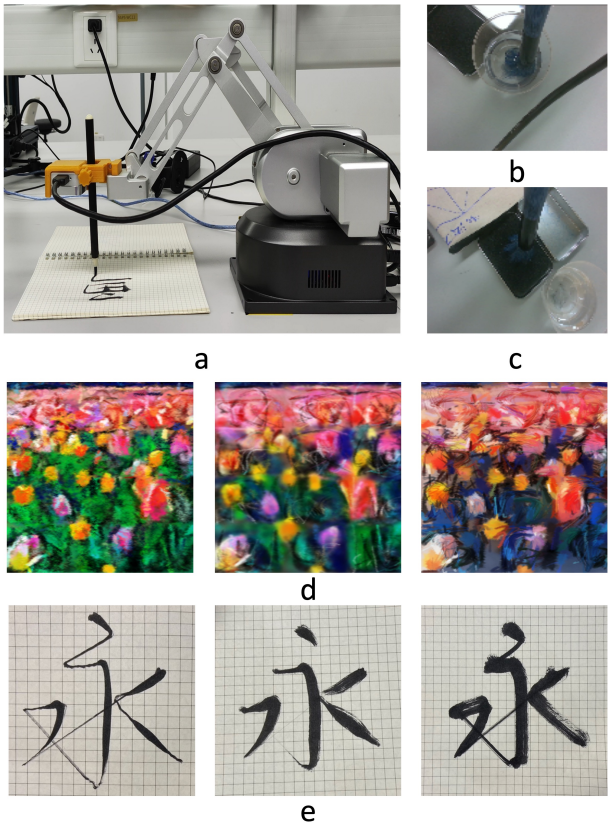


Fig. 1: Our Robotic Sketching System: Developed through reinforcement learning and behavior cloning, our system can recreate identical or transformed versions of a reference image in both simulated and real-world environments. Components used for real robot painting include (a) the Robot setup with Realsense D415, UltraArm robot, and paintbrush; (b) a Water pot; and (c) an Inkpot. The results generated by our robotic sketching system are illustrated in (d) simulated paintbrushes (from left to right: charcoal, pencil, and watercolor), employing 100 strokes; and in (e) a real paintbrush, which utilizes three different stroke models, combining 5 long strokes with 68 small strokes.

Our aim is to design a brand-new robotic sketching system capable of transferring the trained painting policy while incorporating sophisticated brush manipulation techniques.

Main Results: We introduce an innovative robotic sketching system that leverages a painting policy trained via reinforcement learning and behavior cloning for natural media painting. In the simulated environment, our model can acquire intricate painting policies through reinforcement learning, while behavior cloning allows us to fine-tune and adapt these policies. In the real-world context, we have developed a method for transferring the acquired policies while

979-8-3503-8457-4/24/$31.00 ©2024 IEEE

preserving their artistic capabilities, including precise brush manipulation, thus enabling the creation of intricate and expressive sketches by the robotic system.The contributions of our work include:

- Introducing a novel reinforcement learning-based approach to model natural painting media in a simulated environment. Our approach demonstrates the versatility to learn with or without human supervision and excels in navigating continuous high-dimensional action spaces, enabling it to effectively handle large and intricately detailed reference images.
- Developing an adaptive sim-to-real methodology tailored for deformable brushes. This methodology includes estimating contact force and modeling strokes using a Gaussian model. It leverages behavior cloning to initialize policies for painting tasks, facilitating the seamless transfer of learned policies from simulation to reality.
- Creating a robotic sketching system comprising a robotic arm, egocentric view camera, and brush, mirroring the MyPaint virtual environment. This real-world setup empowers us to undertake complex artistic endeavors, including painting various subjects.

We conducted a rigorous evaluation of our results, encompassing a diverse set of reference images that span a wide range of artistic styles, as illustrated in Figure 1. Our virtual painting agent demonstrates its ability to generate high-resolution outputs tailored to various painting media. It excels in creating intricate, multi-stroke images, replicating complex patterns with precision. Simultaneously, our robot sketching system adeptly produces long, thin strokes with smooth variations in thickness and pressure, closely resembling the nuanced strokes of a human artist. These achievements are made possible through the combined techniques of behavior cloning and stroke modeling.

II. Related Work

A. Learning-based Drawing

Several related efforts have tackled similar challenges in this field. Xie et al. [5], [12], [13] introduced methods employing reinforcement learning and inverse reinforcement learning to simulate strokes. These techniques derive policies from either reward functions or expert demonstrations. However, Xie et al. [5], [12], [13] primarily concentrate on creating reward functions for generating oriental painting strokes, necessitating expert demonstrations for supervision. More recently, Ha et al. [6] gathered a substantial dataset comprising millions of basic object sketches with recorded painting actions. They trained a recurrent neural network model in a supervised manner to encode and reproduce action sequences, showcasing the model's capacity to generate new sketches. Building on this, Zhou et al. [14] utilized a combination of reinforcement learning and imitation learning to reduce the supervision required for training similar sketch generation models. In contrast to [6], [14], our painting policies operate in a complex painting environment characterized by a continuous action space encompassing brush width and color. Our approach learns its policy network with minimal human supervision and readily adapts to real robotic systems.

B. Visual Generative Methods

In the realm of generative image models conditioned on text, significant advancements have enabled high-fidelity, diverse, and controllable image synthesis [3], [4], [15], [16], [17], [18], [19]. These improvements are attributed to large-scale, aligned image-text datasets [20] and scalable generative model architectures. Notably, diffusion models have excelled in learning high-quality image generators with a stable and scalable denoising objective [21], [22], [23].

Visual generative methods typically directly synthesize visual output in pixel spaces. Recent approaches employ CNNs and large datasets for learning mapping functions [24]. Some use variational autoencoders [25] to implement style transfer [26]. Others apply generative adversarial networks (GANs) [27], such as Cycle-Consistent Adversarial Networks [7], which excel at generating natural and artistic images [28], [29], [30] and videos [31], [32].

However, these generative methods may fall short in achieving high-resolution results. Conversely, our stroke-based approach generates paintbrush trajectories that are adaptable to diverse synthetic and real painting environments using robotic arms.

C. Robotic Sketching Systems

In the development of robotic sketching systems, various approaches have been investigated. In significant work by Lee et al. [8], a hierarchical reinforcement learning (RL) model was proposed for painting tasks, where a high-level controller learns the painting policy and a low-level policy to control the robot arm. This is the most relevant previous work to our research, as both aim to address robotic sketching using RL. The key difference lies in our implementation of a more complex virtual and real painting environment, capable of handling various intricate painting media such as oil painting, watercolor, and ink, as well as deformable paintbrushes through a larger action space and a behavior cloning framework. Consequently, we achieve more intricate and detailed results.

Other studies, such as those by Chen et al. [9], El et al. [10], and Vempati et al. [11], have focused on learning low-level manipulation policies to tackle challenges presented by uneven painting surfaces. A distinctive feature of our approach, compared to these studies, is that our method does not require explicit environmental modeling. Consequently, our algorithm exhibits broader applicability in real-world scenarios and a wider range of painting tasks, marking a significant contribution to the field of robotic painting algorithms.

III. Training a Painting Policy

In this section, we delve into the technical intricacies of our reinforcement learning-based painting policy. We start by introducing the core components of reinforcement learning,

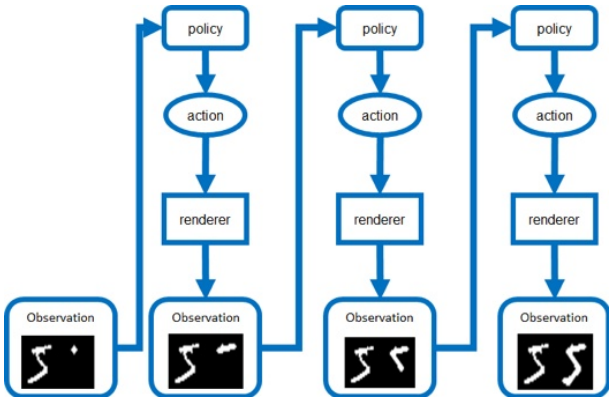


Fig. 2: *Overview of Training/Rollout Process:* For each time step, the current state of the canvas and the reference image form the observation for the policy network. Based on the observation, the policy network selects an action to execute and update the canvas accordingly.

including the action space, observation, reward, and policy network. We then elaborate on our training and runtime algorithms, discussing techniques aimed at improving learning efficiency, such as curriculum learning. Additional technical details can be found in [33].

1) Action Space: To capture the essence of painting behavior, we represent actions using stroke properties, including angle, length, size, and color. Specifically, we define the action as a 6-dimensional vector, $a_t = [\alpha_t, l_t, w_t, c_t] \in \mathbb{R}^6$, with each value normalized to $[0, 1]$. The action space is continuous, enabling us to employ policy gradient-based reinforcement learning algorithms. Notably, when $w = 0$, the brush moves above the canvas without applying paint.

2) Observation: Our approach extends the observation o_t to include both the current state s_t and the reference image s^*. We use an egocentric observation strategy, centering the paintbrush on the canvas. This simplifies the action space, eliminates the need for a replay buffer, and enables training in a continuous action space and large state space. The state observation o_t is defined in Equation 1 where (h_p, w_p) represents the paintbrush position, and (h_o, w_o) denotes the egocentric window size.

$$o_t = \left\{ s_t \left[h_p - \frac{h_o}{2} : h_p + \frac{h_o}{2}, w_p - \frac{w_o}{2} : w_p + \frac{w_o}{2} \right], \right. \\ \left. s^* \left[h_p - \frac{h_o}{2} : h_p + \frac{h_o}{2}, w_p - \frac{w_o}{2} : w_p + \frac{w_o}{2} \right] \right\}. \quad (1)$$

3) Reward: In our setup, the reward for each action is determined by the difference between the canvas and the reference image. A loss function is employed to calculate the action's reward during each reinforcement learning iteration. To incentivize the painting agent to match the color and shape of the reference image precisely rather than aiming for an average color, we slightly modify the L_2 loss into $L_{\frac{1}{2}}$,

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

where the image s and the reference image s^* are matrices with dimensions $h \times w \times c$. Here, w and h denote the width and height of the image, while c represents the number of color channels.

After defining the loss between I and I^{ref}, we normalize r_t using Eq. 3, such that $r_t \in (-\infty, 1]$.

$$r_t = \frac{L(s_{t-1}, s^*) - L(s_t, s^*)}{L(s_0, s^*)} \quad (3)$$

4) Policy Network: The first hidden layer applies convolution with 64 8×8 filters and a stride of 4. The second layer employs convolution with 64 4×4 filters and a stride of 2, followed by the third layer using convolution with 64 3×3 filters and a stride of 1. Subsequently, the network connects to a fully connected layer comprising 512 neurons. All layers employ the ReLU activation function [34]. We illustrate our training and rollout algorithm in [35].

5) Curriculum Learning: In the context of a continuous action space $a \in \mathbb{R}^6$, we face challenges with growing sampling space and noise from policy gradient-based reinforcement learning. To address this efficiently, we adopt curriculum learning, progressively increasing sampled trajectories during training. In RL, the optimal policy π^* maximizes the expected long-term reward q_t, which accumulates rewards r_t over a time horizon $t_{\max}$ with a discount factor $\gamma \in \mathbb{R}$:

$$q_t = \sum_{t=1}^{t_{\max}} r_t \gamma^t. \quad (4)$$

For painting policies, numerous goal configurations are sparsely distributed in a high-dimensional space, challenging the agent's convergence. We adapt the horizon parameter $t_{\max}$ by introducing a reward threshold r_{thresh}, gradually increasing it during training as:

$$\hat{t}_{\max} = \arg\min_i (r_i > r_{\text{thresh}}). \quad (5)$$

This redefined horizon parameter allows the policy gradient algorithm to converge effectively when dealing with complex goal configurations, encouraging the policy to prioritize rewards within limited time steps, reducing exploration space.

IV. Sim-to-Real Brush Manipulation

In this section, we explain our sim2real transfer methods from the painting policy in Section III with the aim of seamlessly applying the policy to real-world robotic drawing tasks for precise brush manipulation and stroke control. Accurate pressure estimation is crucial for controlling stroke shapes and interactions with various painting media like ink and water. Rather than using force sensors, we employ advanced modeling and image analysis techniques for pressure estimation, making it applicable in situations where force sensing is impractical. Our practical experiments combine end-effector image capture to determine optimal pressure ranges with stroke image sampling for precise mapping. We

Algorithm 1 Hybrid Pressure Estimation

Input: Initial robot action in C-space for pressure [$a_{\text{p\_max}}$, $a_{\text{p\_min}}$], real robot renderer $s = R(a)$

Output: Optimal maximum pressure $a^*_{\text{p\_max}}$, minimum pressure $a^*_{\text{p\_min}}$, mapping function from stroke to the real robot configuration $a' = M(s)$

function EstimatePressure($a_{\text{p\_max}}, a_{\text{p\_min}}$)
if $(a_{\text{p\_max}} - a_{\text{p\_min}}) \leq$ a_step **then**
  return $M(s)$
else
  $a_{\text{p\_guess}} \leftarrow (a_{\text{p\_min}} + a_{\text{p\_max}})/2$
  $s \leftarrow R(a_{\text{p\_guess}})$
  update policy M with s and $a_{\text{p\_guess}}$
  return EstimatePressure($a_{\text{p\_max}}$, $a_{\text{p\_guess}}$)
  return EstimatePressure($a_{\text{p\_guess}}$, $a_{\text{p\_min}}$)
end if
end function
$M(s) \leftarrow$ EstimatePressure($a_{\text{p\_max}}, a_{\text{p\_min}}$)
return $M(s)$

deconstruct our acquired policy into high-level and low-level components. The high-level policy is trained using behavior cloning, standardizing stroke order, especially for handwriting. In contrast, the low-level policy is developed via efficient sampling-based reinforcement learning, translating the original RL policy into real-world actions.

A. Contact Force Estimation

Accurately estimating the contact force between the pen tip and the painting medium is a crucial aspect of robotic brush manipulation. However, precise force sensors are often unavailable. Therefore, we employ image analysis methods to infer pressure values, as described in Alg. 1.

1) Observation of Stroke Images: This approach involves indirectly observing environmental changes, specifically the stroke images on the paper, to infer variations in pressure. It is an intuitive method in which we record the shapes of strokes and the configuration of the robotic arm. We can then interpolate to obtain the desired stroke characteristics.

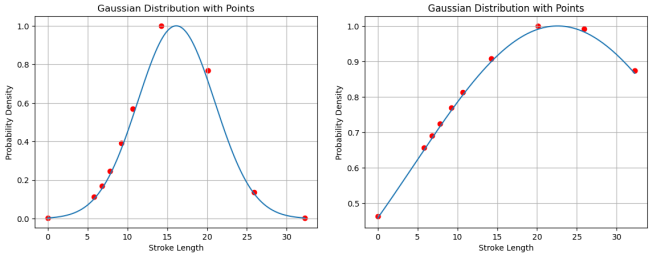
However, finding a suitable arm configuration is not straightforward. Like training reinforcement learning in simulation, this method requires extensive sampling, with many instances yielding no positive rewards due to the limited deformation range of the brush.

2) Observation of End-Effector Images: In contrast to observing stroke images, this method offers a more direct approach. It involves capturing the shape changes of the flexible end effector.

While this method may be susceptible to image noise, it provides valuable information about the pressure limit of the flexible object. We utilize linear fitting to identify the point at which deformation no longer occurs, treating it as the pressure limit.

B. Mapping Actions from Simulation to Reality

In Section III, we defined actions in a simulated environment, which may differ from the actions required in the real-world environment. Therefore, we need to map robot actions from the simulated environment's action space to the robot's configuration space in the real world.



(a) Mean: 0.5, Standard Deviation: 0.3. **(b)** Mean: 0.7, Standard Deviation: 0.8.

Fig. 3: *Effect of Gaussian Stroke Model on Stylization:* We model a long stroke composed of segments using a Gaussian distribution. These correspond to the first column in Fig. 1, where variations in artistic style are achieved by adjusting the Gaussian parameters.

The first challenge is that the painting plane in the simulated environment differs from the real robot environment. Therefore, we need to find a 2D plane in the 3D configuration space to serve as the painting space. The action mapping formula is computed similarly to the camera's extrinsic calibration.

The second challenge arises because certain actions cannot be directly translated into robot movements but still have a limited visual effect. These include:

1) Stroke thickness, which can only be adjusted by changing the brush's contact force.
2) Color, which, in our setup, is limited to monochrome. Color changes are achieved through interactions with the environment, such as dipping in ink, water, or interacting with a sponge.
3) Tilt, which our 3-DoF robot cannot achieve directly due to limited kinematics.

To approximate these effects, we employ the following methods:

1) Gaussian Modeling of Strokes: To achieve an artistic font treatment, we emulate the stroke characteristics of human artists. This is accomplished through Gaussian modeling for each stroke, which captures the distribution of the stroke's centroid and pressure. This approach empowers us to create artistic fonts with diverse styles. By fine-tuning these parameters, we can generate various types and styles of strokes, leading to font diversity as illustrated in Fig. 3. For all straight-line strokes, we employ this method to merge them into long, thin strokes, enhancing the overall line's smoothness and natural appearance.

2) 2D to 3D Action Projection: To match the actions from the simulated environment to the real robot's configuration space, we need to project 2D actions into a 3D configuration space. This projection can be defined using the following equation, which is like a camera's extrinsic calibration projection:

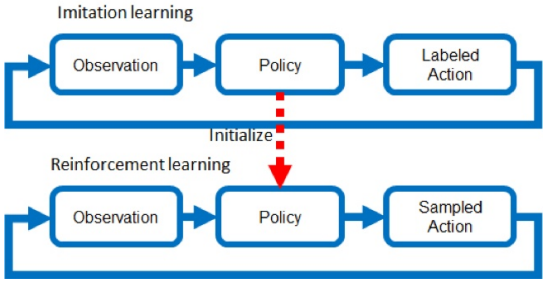


Fig. 4: *Behavior Cloning for Policy Initialization:* We utilize a behavior cloning algorithm to train the policy, extending the action space to initialize the reinforcement learning (RL) policy within a real environment setup. The action space used in behavior cloning is a subspace of the RL action space and includes direction and on/off canvas actions. This initialization process bridges the gap between behavior cloning and RL, facilitating effective policy learning in the real environment.

$$\begin{bmatrix} x_{\text{robot}} \\ y_{\text{robot}} \\ z_{\text{robot}} \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{\text{painting}} \\ y_{\text{painting}} \\ 1 \end{bmatrix}$$

Here, x_{robot}, y_{robot} and z_{robot} represent the robot's coordinates. x_{painting} and y_{painting} are the desired painting coordinates in 2D space. The transformation matrix $\begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$ maps the 2D painting coordinates to the 3D robot configuration, allowing us to generate actions that correspond to the desired painting locations and orientations in the real world.

C. Behavior Cloning

Behavior cloning utilizes a dataset that pairs observations with corresponding actions to train a policy, mimicking expert trajectories or behaviors. In our specific scenario, the expert trajectory is represented in the paired dataset $o_{(t)}, a_{(t)}$. We employ behavior cloning to initialize the policy network for reinforcement learning, utilizing the supervised policy trained with this paired data. The dataset can be generated by a human expert or an optimal algorithm possessing global knowledge, a feature absent in our painting agent. Once we acquire the paired dataset $o_{(t)}, a_{(t)}$, a common approach involves employing supervised learning techniques, such as regression or classification, to train the policy. The training process can be framed as an optimization problem:

$$\pi^* = \arg\min \sum_{t}^{N} ||\pi(o_t) - a_t||. \tag{6}$$

Generating an expert dataset for our painting application can be challenging due to the significant variation in reference images and painting actions. However, we can create a paired dataset by rolling out a policy during the RL training process. Additionally, there are existing datasets like KanjiVG and Google's Quick, Draw! that provide paired, supervised data [36], [37].

When we train behavior cloning with real data, the painting policy tends to produce long, thin strokes rather than multiple simple strokes. This method also enables us to achieve a natural brushwork effect, as demonstrated in Fig. 1(e).

V. Experiment

A. Robotic Sketching System Setup

In our simulated painting setup, we have created an environment that allows the painting agent to explore a high-dimensional action space and observation space based on MyPaint [38]. This setup aligns with the description provided in Sec. III.

For the real brush manipulation experiment, we implement our approach using an UltraArm, which features 3 DoFs for movement as shown in Fig. 1. The primary experimental setup includes a water pot and foam, allowing the robot to manipulate a paintbrush by absorbing water, squeezing it, or using the object to reshape it. This setup serves to demonstrate that our method can effectively learn the complexity of high DoF end-effector manipulation tasks in a practical and realistic scenario.

By incorporating the water pot and foam into the experimental setup, we introduce additional challenges that the robot must learn to overcome. These include controlling the amount of water absorbed by the paintbrush, adjusting the pressure applied when squeezing or reshaping the brush, and maintaining a stable grip on the brush throughout the manipulation process. These added complexities showcase the adaptability and effectiveness of our approach in handling diverse manipulation tasks involving deformable materials and intricate interactions with the environment.

B. Data Preparation

In the scope of our real-robot experiments, we selected the KanjiVG dataset [36] for our training endeavors. This dataset, rich in depth, provides detailed stroke information for approximately 2,000 distinct characters. This dataset, having been meticulously collated from human participants, establishes itself as a premier choice when leveraging behavior cloning in the domain of robotic calligraphy.

Within the framework of our reinforcement learning (RL) strategy, we leaned on the acclaimed CelebA dataset [39] to facilitate the training of our painting agent. It's important to note that our rollout algorithm was architectured employing MyPaint [38], a decision made to ensure the results seamlessly mirror the characteristics of natural media. The nuances of the painting model are distilled implicitly, rooted in the foundational knowledge embedded in the environment model. The versatility and robustness of our algorithm are showcased in Fig. 1.

C. Evaluation

We demonstrated the advantages of our approach by computing performance and comparing visual effects. We designed two experiments to evaluate the performance of our algorithms.

In the first experiment, we delineated the learning curves of both the baseline model and the model implementing curriculum learning (Sec. III-.5), visually represented in

Fig. 5: *Illustration of Stroke Order*: Demonstration of the stroke order generated by our behavior cloning algorithm (Columns 1 and 3) and the final sketches generated by our robot sketching system (Columns 2 and 4).

Fig. 6. Convergence for both models was achieved within $78,000$ episodes. The vertical axis of the graph illustrates the average rewards attained by the trained model on a validation dataset, while the horizontal axis denotes the progression of training episodes. Notably, the incremental growth in average rewards throughout the training process underscores the beneficial impact of curriculum learning on enhancing the convergence of reinforcement learning towards a more optimal policy.

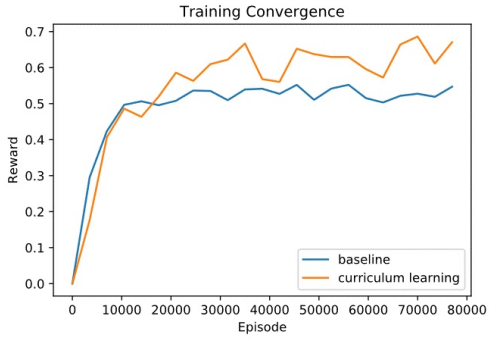


Fig. 6: *Curriculum Learning*: This figure presents a comparative analysis of the learning curve between the approach employing curriculum learning and the baseline. The y-axis represents the average rewards of the trained model on a validation dataset, while the x-axis delineates the training episodes. Both approaches converged after a specific number of steps, with the curriculum learning approach demonstrating superior performance by achieving a higher reward value. The total training steps for both approaches amounted to approximately 10^6.

In the second experiment, we assessed the performance of high-resolution reference images by computing the L_2 loss and cumulative rewards. A comprehensive comparative analysis was conducted against behavior cloning, reinforcement learning, and a combined approach. The benchmark was established by extracting 1000 patches, each of dimensions 400×400, from 10 reference images. Additionally, we iteratively applied both algorithms 1000 times to replicate the reference images, employing the same training dataset for model training. The outcomes, detailed in Table I, show that self-supervised learning exhibited a diminished L_2 loss, highlighting its superiority in this regard, while both methods demonstrated commendable performance concerning cumulative rewards.

Approaches	Cumulative Rewards	L_2 Loss
Behavior Cloning	20.15	512
Reinforcement Learning	97.74	1920
Our Combined Scheme	98.25	1485

TABLE I: *Evaluation of Painting Approaches* We evaluated the performance of behavior cloning, reinforcement learning, and our combined scheme by computing the average cumulative reward and L_2 loss between the final rendering and the reference image on the test dataset.

We also generate various results at different resolutions and with different drawing tools using both the simulated and real robotic sketching systems. For a visual comparison, please refer to the technical report [33].

VI. Conclusion, Limitations, and Future Work

In summary, we have introduced a novel reinforcement learning-based approach and an adaptive sim-to-real methodology. This methodology includes contact force estimation and Gaussian stroke modeling facilitated by behavior cloning for seamless policy transfer between simulation and reality. Our robotic sketching system, mirroring the MyPaint virtual environment, empowers intricate artistic tasks in the real world. These contributions mark advancements in the field of robotic sketching systems, enhancing the creative potential of machines across diverse contexts.

For future research directions, our goal is to extend the temporal horizon and action space within the painting policies, particularly in challenging real-world settings. Furthermore, while our current framework encompasses common stroke parameters, there is untapped potential in integrating additional parameters like pen tilting and pen rotation into our policy framework. Exploring these dimensions holds the promise of enhancing the expressive capabilities of our robotic sketching system, pushing the boundaries of robotic artistic creation.

REFERENCES

[1] A. Hertzmann, "Painterly rendering with curved brush strokes of multiple sizes," in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. ACM, 1998, pp. 453–460.

[2] G. Winkenbach and D. H. Salesin, "Rendering parametric surfaces in pen and ink," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 469–476.

[3] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *ICML*, 2022.

[4] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, "Zero-shot text-to-image generation," *ICML*, 2021.

[5] N. Xie, H. Hachiya, and M. Sugiyama, "Artist agent: A reinforcement learning approach to automatic stroke generation in oriental ink painting," *CoRR*, vol. abs/1206.4634, 2012. [Online]. Available: http://arxiv.org/abs/1206.4634

[6] D. Ha and D. Eck, "A neural representation of sketch drawings," *CoRR*, vol. abs/1704.03477, 2017. [Online]. Available: http://arxiv.org/abs/1704.03477

[7] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2223–2232.

[8] G. Lee, M. Kim, M. Lee, and B.-T. Zhang, "From scratch to sketch: Deep decoupled hierarchical reinforcement learning for robotic sketching agent," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5553–5559.

[9] G. Chen, S. Baek, J.-D. Florez, W. Qian, S.-w. Leigh, S. Hutchinson, and F. Dellaert, "Gtgraffiti: Spray painting graffiti art from human painting motions with a cable driven parallel robot," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4065–4072.

[10] M. El Helou, S. Mandt, A. Krause, and P. Beardsley, "Mobile robotic painting of texture," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 640–647.

[11] A. S. Vempati, R. Siegwart, and J. Nieto, "A data-driven planning framework for robotic texture painting on 3d surfaces," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 9528–9534.

[12] N. Xie, T. Zhao, F. Tian, X. H. Zhang, and M. Sugiyam, "Stroke-based stylization learning and rendering with inverse reinforcement learning," *IJCAI*, 2015.

[13] N. Xie, T. Zhao, and M. Sugiyama, "Personal style learning in sumi-e stroke-based rendering by inverse reinforcement learning," *Information Processing Society of Japan*, 2013.

[14] T. Zhou, C. Fang, Z. Wang, J. Yang, B. Kim, Z. Chen, J. Brandt, and D. Terzopoulos, "Learning to doodle with deep q networks and demonstrated strokes," *British Machine Vision Conference*, 2018.

[15] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," 2022.

[16] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi, "Photorealistic text-to-image diffusion models with deep language understanding," 2022.

[17] C. Saharia, W. Chan, H. Chang, C. A. Lee, J. Ho, T. Salimans, D. J. Fleet, and M. Norouzi, "Palette: Image-to-image diffusion models," 2021.

[18] T. Hu, Z. Chen, H. Sun, J. Bai, M. Ye, and G. Cheng, "Stein neural sampler," 2022.

[19] C. Saharia, J. Ho, W. Chan, T. Salimans, D. J. Fleet, and M. Norouzi, "Image super-resolution via iterative refinement," 2021.

[20] C. Schuhmann, R. Beaumont, C. W. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, P. Schramowski, S. R. Kundurthy, K. Crowson, R. Vencu, L. Schmidt, R. Kaczmarczyk, and J. Jitsev, "Laion-5b: An open large-scale dataset for training next-generation image-text models," 2022.

[21] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," 2020.

[22] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," 2015.

[23] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *CoRR*, 2021.

[24] L. A. Gatys, A. S. Ecker, and M. Bethge, "A neural algorithm of artistic style," *arXiv preprint arXiv:1508.06576*, 2015.

[25] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[26] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European Conference on Computer Vision*. Springer, 2016, pp. 694–711.

[27] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[28] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of gans for improved quality, stability, and variation," *arXiv preprint arXiv:1710.10196*, 2017.

[29] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, 2017, pp. 386–396.

[30] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, "Scribbler: Controlling deep image synthesis with sketch and color," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[31] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Advances In Neural Information Processing Systems*, 2016, pp. 613–621.

[32] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, "Flow-grounded spatial-temporal video prediction from still images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 600–615.

[33] B. Jia and D. Manocha, "Sim-to-real robotic sketching using behavior cloning and reinforcement learning," *arXiv preprint*, September 2023.

[34] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[35] B. Jia, J. Brandt, R. Mech, B. Kim, and D. Manocha, "Lpaintb: Learning to paint from self-supervisionlpaintb: Learning to paint from self-supervision," *Pacific Graphics*, 2019.

[36] K. Contributors. (2023) Kanjivg. [Online]. Available: https://kanjivg.tagaini.net/

[37] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. (2016) The quick, draw! - a.i. experiment. [Online]. Available: https://quickdraw.withgoogle.com/

[38] libmypaint contributors, "libmypaint," *https://github.com/mypaint/libmypaint*, 2018.

[39] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.