

# FRIDA: A Collaborative Robot Painter with a Differentiable, Real2Sim2Real Planning Environment

Peter Schaldenbrand<sup>1</sup>, James McCann<sup>1</sup>, and Jean Oh<sup>1</sup>

**Abstract**—Painting is an artistic process of rendering visual content that achieves the high-level communication goals of an artist that may change dynamically throughout the creative process. In this paper, we present a Framework and Robotics Initiative for Developing Arts (FRIDA) that enables humans to produce paintings on canvases by collaborating with a painter robot using simple inputs such as language descriptions or images. FRIDA introduces several technical innovations for computationally modeling a creative painting process. First, we develop a fully differentiable simulation environment for painting, adopting the idea of real to simulation to real (real2sim2real). We show that our proposed simulated painting environment is higher fidelity to reality than existing simulation environments used for robot painting. Second, to model the evolving dynamics of a creative process, we develop a planning approach that can continuously optimize the painting plan based on the evolving canvas with respect to the high-level goals. In contrast to existing approaches where the content generation process and action planning are performed independently and sequentially, FRIDA adapts to the stochastic nature of using paint and a brush by continually re-planning and re-assessing its semantic goals based on its visual perception of the painting progress. We describe the details on the technical approach as well as the system integration. FRIDA software is freely available at: <https://pschaldenbrand.github.io/frida/>.

## I. INTRODUCTION

Painting is the artistic process of rendering visual content that achieves an artist's high-level, semantic goals. As opposed to its straightforward analogue printing—i.e., creating a copy of an original input—painting is a dynamic process where an artist's initial goals are generally vaguely defined and may change dynamically during the creative process [1], [2] and the artist's goals are specified semantically and at a high-level.

In this paper, we consider how we can enable robots with painting abilities. To this end, we introduce a Framework and Robotics Initiative for Developing Arts (FRIDA) that can computationally model creative processes such as painting to support human creativity. FRIDA showcases a fully integrated robotic system (Fig. 2) that can take inspirations from a human user in the text or image format, e.g., “two soldiers from different countries dancing,” to produce an artistic painting (Fig. 1). FRIDA does so by modeling painting as a *planning* problem where a canvas constitutes the state space and brush strokes are available actions. Here, the objective is to find a sequence of actions, e.g., brush strokes, that would result in a final canvas that satisfies an artist's goals or intentions. An additional objective is to allow for the adjustment of goals based on the progressive execution of a plan. Contrary to existing robot painting systems where

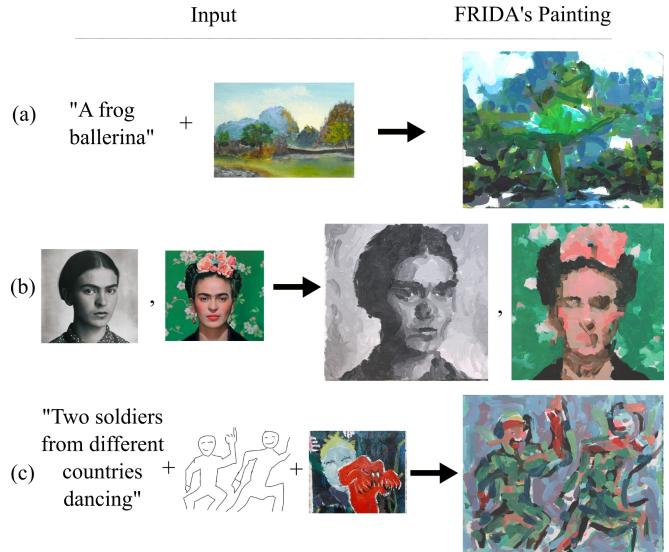


Fig. 1. Human users interacting with FRIDA can describe semantic goals using a variety of simple input such as (a) natural language and style images, (b) source images to be painted precisely, and (c) sketches with language description and style images. Input image sources: [3], [4], [5], [6].

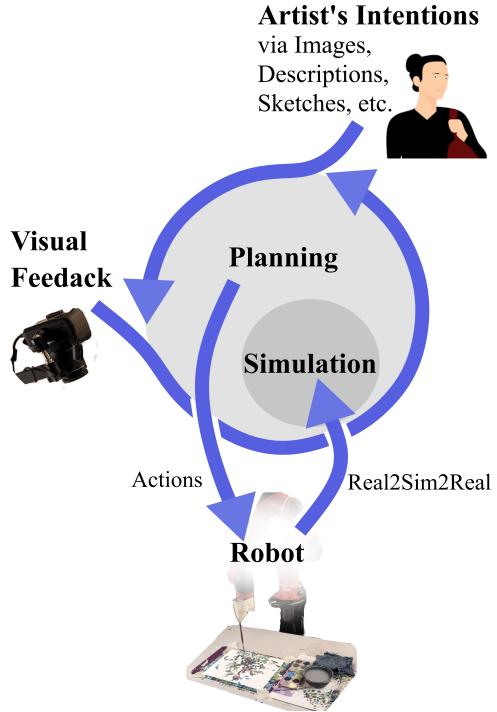


Fig. 2. Planning, in FRIDA, is designed to achieve the high-level goals of the human user, which are specified with multi-modalities. FRIDA plans in a simulated environment created from real robot data and continually optimizes its plan based on visual perception to ensure the artist's intentions are realized.

<sup>1</sup>The Robotics Institute, Carnegie Mellon University  
 {pschalde, jmccann, hyaejino}@andrew.cmu.edu

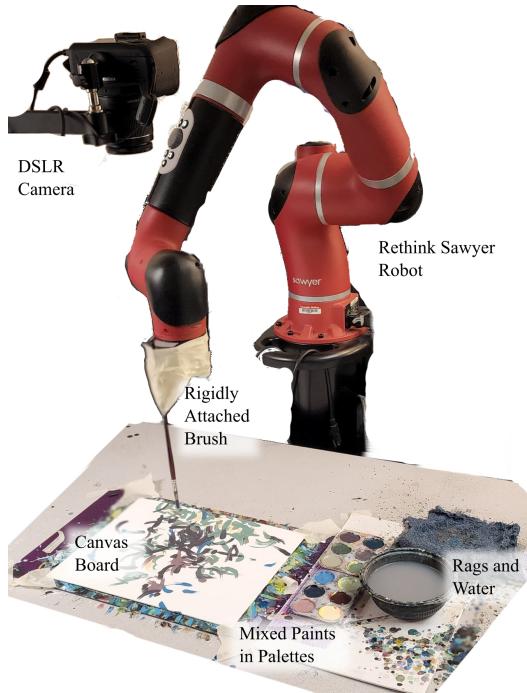


Fig. 3. FRIDA’s embodiment and workspace.

a fully programmed sequence of actions is blindly executed by a robot, FRIDA’s painting approach interleaves content generation and action planning to achieve continual content optimization.

FRIDA’s core technologies draw on two insights of the artistic process [1]: 1) art has high-level, semantic goals, and 2) art is a dynamic process which needs to adapt and reconsider its goals constantly during the creation process. To achieve high-level semantic goals, we design loss functions to compare the semantic goals (or user inputs) and the current canvas. Instead of computing the loss using the pixel values, we use the feature values from pre-trained deep neural networks (DNNs) as the features extracted from DNNs tend to correlate well with human perception of image style [7], [8], [9] and multi-modal alignment [10], [11]. We hypothesize that using DNN features would result in paintings that are semantically relevant to high-level goals as opposed to merely replicating (or printing) a given input. In order to plan using feedback from DNNs, we create a differentiable, simulated painting environment which enables our planner to optimize brush actions directly toward semantic goals using stochastic gradient descent. Inspired by the idea referred to as Real2Sim2Real used in other robotics problems [12], the simulation environment is created by modeling real brush strokes generated by the robot which reduces the Sim2Real gap.

Capturing the dynamic nature of painting is a challenge when framing it as a robot painting problem. The majority of existing work in robot painting models the painting process as analogue printing, that is, an input image is the same as their final goal to reproduce in painting. Departing from those views, our approach follows the idea of planning with partial information in robotics where an initial plan is generated to initiate the execution but continuously gets updated as a robot acquires more information from an environment.

Our contributions:

- 1) A robotics framework and initiative for interdisciplinary research to promote human creativity, addressing technical challenges in core robotics fields including planning, simulation, and human-robot interaction in the domain of visual arts. FRIDA’s complete software stack is open sourced (<https://github.com/pschaldenbrand/Frida>);
- 2) A differentiable, high-fidelity painting simulation environment developed using Real2Sim2Real methodology that reduces the Sim2Real gap from prior works;
- 3) A planning algorithm for performing multiple tasks that have high-level, semantic goals under stochastic circumstances;
- 4) An intuitive interaction interface, e.g., human users can describe semantic goals using natural language, sketches, and/or style images.

## II. RELATED WORK

### A. Simulated Painting

Stroke-Based Rendering (SBR) recreates a given target image using a set of primitive elements that usually resemble brush strokes of paint. Procedural SBR methods generally use rules and heuristics to generate the stroke plan [13], [14]. Planning-based SBR methods use search, optimization, or learning models such as Reinforcement Learning or Recurrent Neural Networks to generate a stroke plan with an objective of replicating an input image [15], [16].

Recent SBR methods expands the input space to incorporate high-level goals to generate brush stroke simulated paintings based on language descriptions and/or style specification [10], [9], [17]. While these methods present appreciable results in simulation, technical challenges specific to transitioning from simulation to real robots have not been addressed.

### B. Robot Painting

There have been numerous robot-created paintings including notable works that had competed in an annual competition in 2016–2018 [18], but technical details of most works have not been published. Based on published works, existing robot painting approaches can roughly be categorized into two groups: engineered systems and learning-enabled systems.

*1) Engineered robotic painting systems:* use well measured equipment to ensure that the planning environment is accurate to the real environment and use rules and heuristics for planning. The Dark Factory portraits [19] utilize a highly accurate robotic arm with known models of brush shape and size. They plan a full sequence of actions *a priori* such that the plan can be blindly executed. E-David [20] uses a simulated environment constructed to be similar to its painting equipment then draws strokes perpendicular to gradients in the target image. In general, the engineered systems are capable of high-fidelity reproductions of input images as they meticulously engineer to minimize the sim2real gap in their setup; however, they are not generalizable to different equipment or settings. Furthermore, these approaches generally do not support more than replicating a given image.

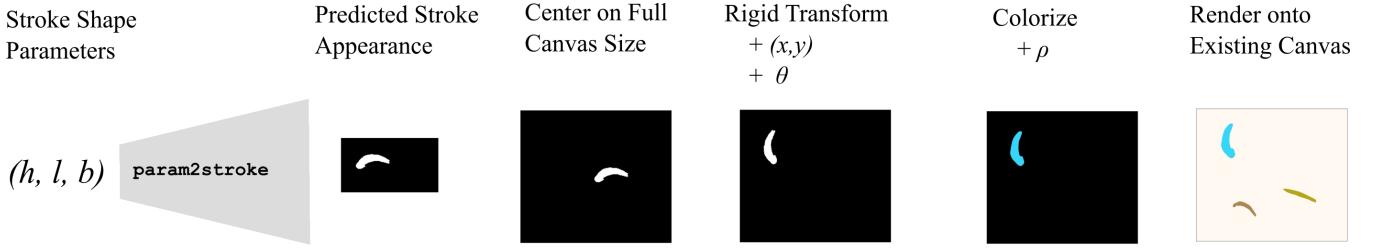


Fig. 4. The process of rendering a stroke, given its parameters, onto an existing canvas in our differentiable simulated painting environment.

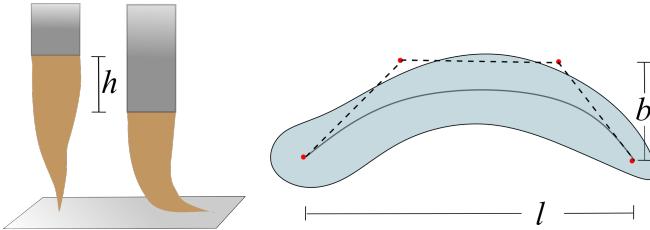


Fig. 5. Our brush shape model has three parameters: thickness ( $h$ ), bend ( $b$ ), and length ( $l$ ).

2) *Learning-enabled robotic painting systems*: generally use simulation environments to plan brush strokes and then execute the plan using a physical robot. Due to a huge sim2real gap, brush stroke plans based directly on simulation methods [10], [9] produce poor-quality paintings or are even infeasible on real robot systems. It has been shown that additional constraints help reducing the sim2real gap to enable robots to paint according to a generated plan [21], [22], but such rigid constraints sometimes result in vague or imprecise outcomes. In line-drawing, [23] used reinforcement learning to learn both the SBR instructions and the low level robot instructions for the reproduction of sketches. Their approach is designed to plan once and execute a given plan as is without observation feedback in the loop. In painting, however, visual feedback is crucial as painting is a continuously evolving process [1].

### C. Brush Stroke Modeling

Brush strokes can be represented using a height map and a color map as in [24] where Generative Adversarial Networks are used to map a user input trajectory into a synthesized brush stroke. In their work, both training and testing were done using data synthesized using a volumetric oil painting simulator based on WetBrush [25]. While the outputs appear impressive in simulation, the challenge still remains unanswered how such a simulated input can be translated into a real painting, for example, by a robot.

Wang et al. [26] use brush parameters such as the width, drag, and offset of the brush's bristles to create a very accurate brush stroke model. They use pseudospectral optimal control to optimize trajectories of brush strokes to fit the target calligraphy character, which works well with calligraphy where an initial path is given in a reasonably accurate form and the brush strokes are clearly separated by white space. In the painting domain, however, a more generalizable approach is needed due to the fact that the shapes of brush strokes used in painting are highly flexible and unconstrained and that brush strokes frequently overlap with previous ones.

## III. METHODS

### A. Brush Stroke Model

Inspired by [26], we parameterize the space of brush strokes using three parameters as shown in Fig. 5. In addition to brush shape attributes, i.e., the length  $l$  of the stroke, and the amount  $b$  that the stroke bends up or down, the thickness  $h$  of the stroke specifies how far the brush is pressed proportionally to the canvas. A brush stroke is parameterized by its shape, denoted by  $(h, l, b)$ , the location coordinates on a canvas  $(x, y)$ , orientation  $\theta$ , and color  $\rho$  in the RGB format. The stroke trajectory can then be represented by a cubic Bézier curve where the horizontal coordinates are a linear interpolation between 0 and  $l$ , and the vertical coordinates are 0 at the end points and  $b$  in the center points.

### B. Real Data to Simulation

Whereas existing models such as [26] use only shape features of the rendered images that would require some model of a brush tool for a robot control interface, our definition of thickness connects the parameter space with a brush tool and a robot. During the calibration phase, we generate random brush strokes to train the `param2stroke` model, a Neural Network comprised of two linear layers followed by two convolutional layers and an bilinear upscaler, that translates a brush stroke shape tuple directly into the appearance map of the brush stroke. The brush stroke shape tuple can deterministically be translated into control inputs for a real robot.

A rudimentary approach to creating a differentiable, simulated robot painting environment would be to allow the robot to paint randomly and continuously to collect a large enough dataset of paired robot actions to the effects on the canvas to model this relationship. While this method works well in simulated environments [24], [21], [16], [17], where thousands of brush strokes can be produced on the order of seconds, generating a similarly large-sized real dataset is impractical. Painting in real life is slow, and if the brush or other materials were altered, the entire process would need to be restarted. Instead, we augment the dataset using existing differentiable functions, such as rigid transformations for positioning and orienting strokes and stamping methodology for rendering individual brush strokes onto an existing canvas, to allow our painting environment to be simulated with a reasonably small number of real brush strokes for modeling.

### C. Differentiable Simulated Painting Environment

The stroke rendering process in our simulation is depicted in Fig. 4: the `param2stroke` network translates

$$l_1 = l_{text} = \cos(CLIP_{img}(r(p)), CLIP_{text}(t)) \quad (1)$$

$$l_2 = l_{style} = EMD(VGG(r(p)) - VGG(t)) \quad (2)$$

$$l_3 = l_{print} = \|r(p) - t\|_2^2 \quad (3)$$

$$l_4 = l_{semantic} = \|CLIP_{conv}(r(p)) - CLIP_{conv}(t)\|_2^2 \quad (4)$$

$$p_{next} = \min_p \sum_{i=1}^4 (w_i l_i), w_i \in \{0, 1\} \quad (5)$$

the thickness, bend, and length parameters into a 2d magnitude map of the brush stroke’s predicted appearance. This magnitude map is then padded such that it is the size of a full canvas. Then the map is translated and rotated to the specified orientation and location. The magnitude map is converted into an RGBA image, and then the stroke is applied to a given existing canvas. Strokes can be layered upon each other to create a complete simulated painting. They can also be rendered onto a photograph of the real canvas for planning throughout the painting process. The whole rendering process is differentiable, meaning that the loss value computed using the rendered canvas can be differentiated, back-propagated through the simulator, and a Stochastic Gradient Descent algorithm updates the brush stroke parameters such that the parameters minimize the loss function.

#### D. Planning Algorithm

Algorithm 1 sketches our planning algorithm for painting with visual feedback given high-level semantic goals. The `optimize` function compares the goals (image or text) with the simulated painting according to one or more objective functions detailed in Section III-E. The simulated painting is the rendering of the brush stroke plan onto the current canvas—i.e., the current image observed by the camera sensor.

To start the `paint` function,  $n\_strokes$  brush strokes are randomly initialized by sampling uniformly over the brush stroke parameters. An initial pass at optimization can be performed with initial high-level goal(s), then the plan is optimized to use the full objective(s). The first  $batch\_size$  brush strokes from the plan are executed, then the remainder of the plan is re-optimized based on the executed brush strokes and objective(s).

#### E. High-Level Goal Objective Functions

To enable our system to achieve high-level goals, we employ a variety of objective functions from recent image synthesis literature. Each objective function has a loss function that compares the brush stroke plan ( $p$ ) to the target input ( $t$ ) which may be language or an image. A plan  $p_{next}$  for the next time step is rendered into a raster image using a differentiable simulated environment ( $r$ ). These objective functions can be used in different combinations to achieve high-level, artistic tasks, e.g., painting from language description with or without a specified style, painting images conceptually, or painting from a sketch.

**Image-Text Similarity Objective** (Eq. 1) This objective optimizes the brush stroke plan ( $p$ ) such that the cosine distance between the CLIP [27] embeddings of both the painting and the language description ( $t$ ) is minimized, guiding the painting to resemble the content of the text, as

---

#### Algorithm 1: Painting Planning Algorithm

---

```

1 Def optimize (plan, targets, objectives) :
2   canvas = camera()
3   while plan is not optimized do
4     sim_painting = canvas + plan.render()
5     loss = 0
6     for objective, target in objectives, targets do
7       loss += objective(sim_painting, target)
8     # Update plan to decrease loss with SGD
9     plan.update(loss)
10    return plan
11 Def paint (targets, n_strokes, objectives,
12   init_objectives) :
13   # Initialize a brush stroke plan
14   plan = init_brush_strokes(n_strokes)
15   # [optional] Do an initial pass at optimization
16   if init_objectives is not None then
17     plan = optimize(plan, targets, init_objectives)
18   plan = optimize(plan, targets, objectives)
19   while plan.n_strokes > 0 do
20     # Paint some strokes from plan
21     plan.execute(batch_size)
22     plan = plan[batch_size:]
23     # Update Plan
24     plan = optimize(plan, target, objective)

```

---

is common in recent CLIP-guided text-to-image synthesis methods [9], [10], [28], [11].

**Style Objective** (Eq. 2) Given an example style image, the style objective guides the painting to resemble the colors, shapes, textures, and other style features of the given image. This objective was created for style transfer methodology [7], [8]. The style objective minimizes the Earth Mover’s Distance ( $EMD$ ) between style features that are extracted using a pretrained object detection model ( $VGG$  [29]), from the brush stroke plan ( $p$ ) and the style image ( $t$ ).

**Simple Replication Objective** (Eq. 3) Image replication is not considered a high-level goal. Instead, it is a straightforward minimization of the  $L_2$  distance between the rendered brush stroke plan and the target image ( $t$ ).

**Semantic Replication Objective** (Eq. 5) Following [30], features can be extracted from the convolutional layers of CLIP which are rich in both semantic and geometric information. For a high-level semantic replication objective, we minimize the  $L_2$  difference of features extracted from the target image and painting from the last convolutional layer of CLIP ( $CLIP_{conv}$ ).

## IV. ROBOT SETUP DETAILS

We used a Rethink Sawyer robot [31] as a machine to test our approach. Any Robotics Operating System (ROS) compatible machine with a similar morphology to the Sawyer could feasibly be adapted to execute our approach with only minimal changes to the robot interface code.

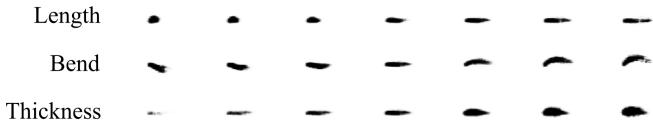


Fig. 6. Depictions of interpolating between minimum and maximum values of each of the three stroke shape parameters with the trained param2stroke model.

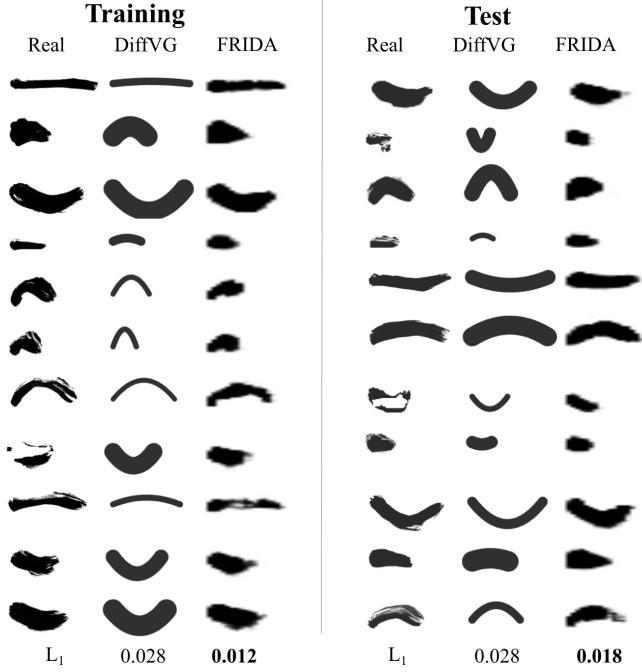


Fig. 7. We compare using DiffVG [32] and FRIDA’s param2stroke model for modeling brush stroke shapes. The average  $L_1$  distance computed on 50 samples between the modeled and real brush strokes is displayed at the bottom.

A photograph of our painting equipment and setup can be seen in Fig. 3. We use a Canon EOS Rebel T7 to perceive the canvas. For all examples in this paper, we used  $11 \times 14$  inch canvas board as painting surfaces. Premixed acrylic paints are provided to the robot in palette trays with up to 12 color options available. Alternatively, from an initial painting in simulation, the colors are discretized to a user-specified number using K-Means cluster; palette preparation is performed accordingly by a human. A rag and water are provided for the robot to clean paint off of the brush, which is performed when switching colors. The brush is rigidly attached to the robot’s end effector and is always held perpendicular to the canvas. Indirect, diffused lighting is necessary, since direct lighting can cause too much glare from the wet paint into the camera. The locations of all the painting materials (canvas, paint, water rag) with respect to the robot are explicitly programmed.

## V. RESULTS

### A. Simulated Painting Environment

The trained param2stroke model can produce strokes with continuous parameter values as seen in Fig. 6. Fig. 7 shows the difference between real brush strokes and FRIDA’s modeled brush strokes using the same input parameters. We also compare these strokes to DiffVG [32] which was

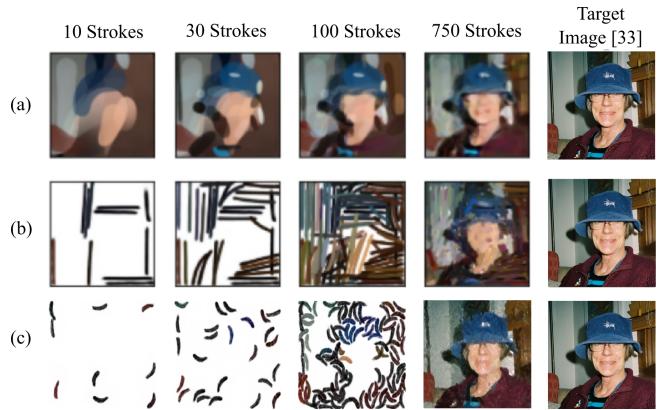


Fig. 8. Comparing the simulation environments of three painting methods painting with various numbers of brush strokes: (a) Huang et al. 2019 [16], (b) Schaldenbrand & Oh 2021 [21], (c) FRIDA (proposed)

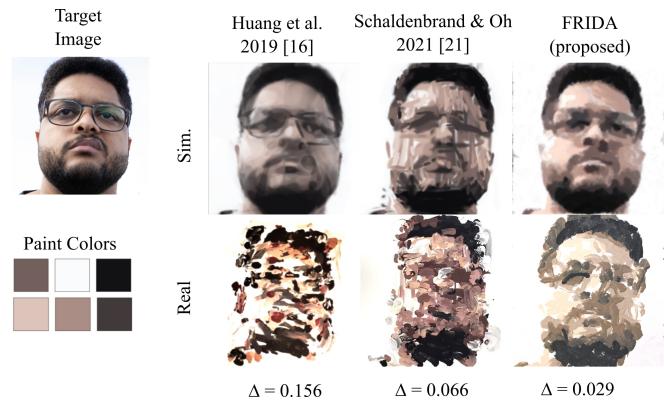


Fig. 9. We compare the Sim2Real gap between FRIDA and two existing methods. The MSE between the simulated plan and the real painting is displayed below each pair.

used for brush stroke planning in [22]. The average  $L_1$  loss between the modeled and real strokes was significantly ( $p$ -value  $< .01$ ) less for FRIDA’s stroke model than DiffVG.

We qualitatively compared our approach to Huang et al. 2019 [16] and Schaldenbrand & Oh 2021 [21]. Fig. 8 compares the brush strokes in early stages of painting simulation where we can observe drastic differences.

Fig. 9 shows the comparison in terms of the sim2real gap for entire paintings. In simulation, Huang et al. 2019’s Reinforcement Learning (RL) model is able to almost perfectly replicate a given image due to their unconstrained stroke model, e.g., allowing strokes that are huge in size and have varying opacity; however, when we fed the strokes to a painting robot, the painting produced was vastly dissimilar to both the simulation and target image. Schaldenbrand & Oh 2021 constrained the brush stroke parameters (length, width, color, and opacity) such that a robot was more capable of executing the strokes; however, the constraints made it challenging for their RL model to accurately replicate a target image. For the proposed approach, we used our simulation to recreate the target image using the Simple Replication Objective (Eq. 3) and did not re-plan with perception for fair comparison. Our proposed approach shows clearly visible improvement in recreation both in simulation and real painting.



Fig. 10. We replicate StyleCLIPDraw [9], a method for generating drawings using language descriptions and style from example images, using FRIDA.

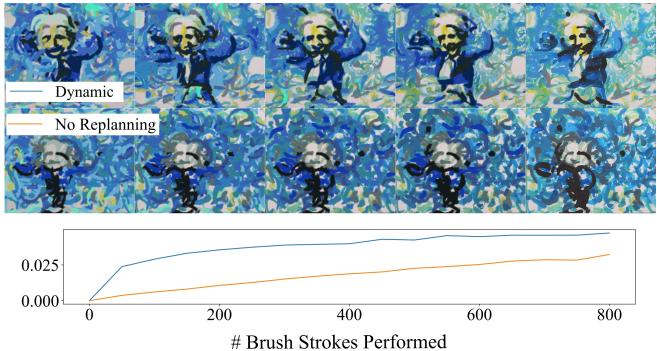


Fig. 11. FRIDA painting with text input “Albert Einstein Dancing” in the style of van Gogh’s *The Starry Night* with and without replanning. The left most images are the initial plan followed by the plan after 200 brush strokes performed. Below, the mean squared error between the current plan versus the initial plan is plotted.

### B. Dynamic Planning and Adaptation

We painted with and without FRIDA’s dynamic replanning system and plotted the deviation from the initial plan in Fig. 11. Without replanning, the difference between the current and initial plan grows linearly as the plan is executed from simulation to reality stroke by stroke. With replanning, the plan changes more significantly from the initial plan as the algorithm adapts to the stochastic execution of the plan, resembling the creative process of human artists [1].

### C. Painting with High-Level Goals

**1) Painting from Language Description with Specified Style:** We use the two loss functions from the StyleCLIPDraw [9] approach of generating visual content using a style image and language description with our simulated painting environment by conceretely optimizing the Style Objective (Eq. 2) and the Image-Text Similarity Objective (Eq. 1). Results can be seen in Fig. 10.

StyleCLIPDraw images tend to lack the original compositional elements of the style image. To retain the style image’s composition, we do an initial optimization to replicate the style image. The initial brush stroke plan is now in a local minimum which will be adapted with the full style and text objectives. Fig. 10 shows that faces and colors appear

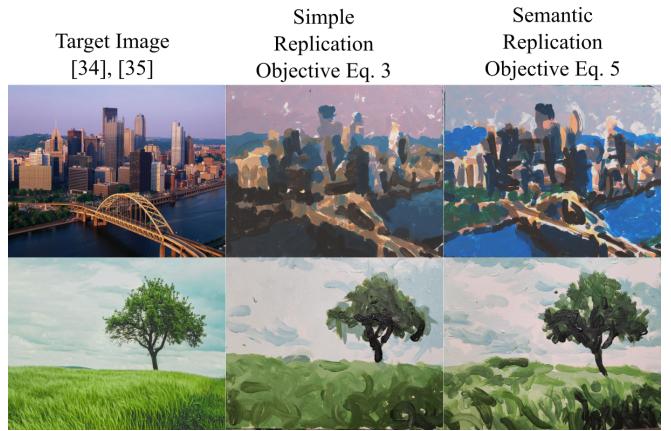


Fig. 12. FRIDA’s paintings using the Simple Replication Objective (Eq. 3) versus the High-Level Semantic Replication Objective (Eq. 5).



Fig. 13. Results from two surveys assessing how well paintings replicated the reference image and how well they retained the high-level content. Percentages shown are preferences for Semantic Loss (Eq. 5) paintings.

where they were initially located in the content image thereby providing a method of transferring compositional elements of style.

**2) Painting Images Conceptually:** We compare painting using the Simple and Semantic Replication Objectives in Fig. 12. We hypothesized that the Semantic Replication Objective would better capture high-level content of the target image. To test this quantitatively, we recruited 103 Amazon Mechanical Turk participants to (1) “select the painting that looks the most like the target image” and (2) “select the painting that captures the high-level ideas of the reference image’s scene better” and to explain how they made their decision. We refer to these surveys as the replication and high-level questions, respectively. Simulated paintings were used to avoid noise generated by human error in palette preparation of which six pairs were generated with 10 evaluators for each question, painting pair. 73.3% and 68.3% of participants selected Semantic Replication Objective paintings for the replication and high-level questions, respectively. These two averages were both significantly larger than 50% at a p-value of 0.01 and were not statistically distinct. While the two questions were different, we noticed that participants claimed to use many of the same features to make their decisions for each question which included colors, shapes, and particular details such as grass and clouds. A breakdown of selections for each painting pair is in Fig. 13.

**3) Sketch to Painting:** User generated sketches can be used to guide the composition of the painting using the Semantic Replication Objective as seen in Fig. 1.

## VI. LIMITATIONS

Our system has been simplified in many ways: painting with discrete color options, brush strokes assumed to be independent, no modelling of the wetness of paint, not modeling how much paint is on the brush, and the brush being fixed perpendicular to the canvas. In future work we hope to explore methods to reduce these limitations.

The second insight of art from [1] is that the painting process is adaptable and dynamic enough that new styles can emerge. While our system does adapt its goals as it paints, our definition of dynamic in the system is far more constrained than that of [1]. In future work we hope to have a system that can explore more and create novelty as it plans.

## VII. CONCLUSIONS

We introduce FRIDA, a robotics framework that computationally models the creative process of painting including stochastic nature of acting with paint and a brush. FRIDA is also a robotics initiative to promote human creativity, rather than replacing it, by providing intuitive ways for humans to express their ideas using natural language or sample images. Whereas visual content generation in simulation and that in physical robot painting have been separated in existing work, FRIDA’s differential planning environment enables seamless interweaving of various deep neural networks-based content generation models and the actual painting planning. FRIDA’s simulation environment developed from real data, an idea known as real2sim2real, reduces the sim2real gap and achieves higher-fidelity to the robot’s capabilities than prior learning-enabled painting robots. FRIDA’s planning environment enables use of pretrained models as feedback functions, which we have shown can achieve high-level content replication in a survey. FRIDA is open sourced to advocate interdisciplinary research and education in robotics and arts.

## ACKNOWLEDGMENT

We thank Jia Chen Xu for his contribution to FRIDA’s perception, and Jesse Ding and Heera Sekhr for meaningful conversations and domain expertise. This work was supported by NSF IIS-2112633 and the Technology Innovation Program (20018295, Meta-human: a virtual cooperation platform for a specialized industrial services) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea).

## REFERENCES

- [1] A. Hertzmann, “Toward modeling creative processes for algorithmic painting,” *arXiv preprint arXiv:2205.01605*, 2022.
- [2] C. Walia, “A dynamic definition of creativity,” *Creativity Research Journal*, vol. 31, no. 3, pp. 237–247, 2019.
- [3] Dave Gehman, “Painting of one of your creative commons photos from flickr, san cristobal de la barranca, rural jalisco, mexico,” 2009. [Online]. Available: \url{https://www.flickr.com/photos/wonderlane/4154304447}
- [4] Guillermo Kahlo, “Frida on bench,” 1926, frida Kahlo Museum Trust.
- [5] Nickolas Muray, “Frida on bench,” 1939, carbon print, 18 x 14 in.
- [6] Troy Taylor, “Given the current circumstances,” 2020, [Online; accessed August 10, 2022]. [Online]. Available: \url{https://www.instagram.com/p/CDrQo3\_gtan/}
- [7] N. Kolkov, J. Salavon, and G. Shakhnarovich, “Style transfer by relaxed optimal transport and self-similarity,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 051–10 060.
- [8] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2414–2423.
- [9] P. Schaldenbrand, Z. Liu, and J. Oh, “Styleclipdraw: Coupling content and style in text-to-drawing translation,” in *Proceedings of the International Joint Conference on Artificial Intelligence*, 2022.
- [10] K. Frans, L. Soros, and O. Witkowski, “Clipdraw: Exploring text-to-drawing synthesis through language-image encoders,” *arXiv preprint arXiv:2106.14843*, 2021.
- [11] A. Smith and S. Colton, “Clip-guided gan image generation: An artistic exploration,” *Evo\* 2021*, p. 17, 2021.
- [12] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg, “Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting,” in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8282–8289.
- [13] A. Hertzmann, “Painterly rendering with curved brush strokes of multiple sizes,” in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, 1998, pp. 453–460.
- [14] K. Zeng, M. Zhao, C. Xiong, and S. C. Zhu, “From image parsing to painterly rendering.” *ACM Trans. Graph.*, vol. 29, no. 1, pp. 2–1, 2009.
- [15] A. Hertzmann, “Paint by relaxation,” in *Proceedings. Computer Graphics International 2001*. IEEE, 2001, pp. 47–54.
- [16] Z. Huang, W. Heng, and S. Zhou, “Learning to paint with model-based deep reinforcement learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8709–8718.
- [17] Z. Zou, T. Shi, S. Qiu, Y. Yuan, and Z. Shi, “Stylized neural painting,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 689–15 698.
- [18] “Robot art competition,” <https://robotart.org/>.
- [19] Rob Carter and Nick Carter, “Dark factory portraits,” <http://www.robandnick.com/dark-factory-portraits>, 2017.
- [20] T. Lindemeier, “e-david: Non-photorealistic rendering using a robot and visual feedback,” Ph.D. dissertation, University of Konstanz, 2018.
- [21] P. Schaldenbrand and J. Oh, “Content masked loss: Human-like brush stroke planning in a reinforcement learning painting agent,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 1, 2021, pp. 505–512.
- [22] M. C. Sola and V. Gulajeva, “Dream painter: Exploring creative possibilities of ai-aided speech-to-image synthesis in the interactive art context,” *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 5, no. 4, pp. 1–11, 2022.
- [23] 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.
- [24] R. Wu, Z. Chen, Z. Wang, J. Yang, and S. Marschner, “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.
- [25] Z. Chen, B. Kim, D. Ito, and H. Wang, “Wetbrush: Gpu-based 3d painting simulation at the bristle level,” *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, pp. 1–11, 2015.
- [26] S. Wang, J. Chen, X. Deng, S. Hutchinson, and F. Dellaert, “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)*. IEEE, 2020, pp. 6696–6703.
- [27] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, “Learning transferable visual models from natural language supervision,” *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: <https://arxiv.org/abs/2103.00020>
- [28] F. Galatolo, M. Cimino, and G. Vaglini, “Generating images from caption and vice versa via clip-guided generative latent space search,” *Proceedings of the International Conference on Image Processing and Vision Engineering*, 2021.
- [29] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
- [30] Y. Vinker, E. Pajouheshgar, J. Y. Bo, R. C. Bachmann, A. H. Bermano, D. Cohen-Or, A. Zamir, and A. Shamir, “Clipasso: Semantically-aware object sketching,” *arXiv preprint arXiv:2202.05822*, 2022.
- [31] “Rethink sawyer,” <https://www.rethinkrobotics.com/sawyer>.
- [32] T.-M. Li, M. Lukáč, M. Gharbi, and J. Ragan-Kelley, “Differentiable

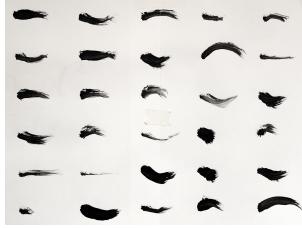


Fig. 14. During the calibration phase, the robot makes random brush strokes to create training data for the brush stroke shape model which learns the relationship between the robot parameters and the appearance of a brush stroke.

vector graphics rasterization for editing and learning,” *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–15, 2020.

## APPENDIX

### A. Robot Calibration

Minor shifts in the canvas location, different brushes, and altered lighting conditions can greatly divide the simulation from reality. To account for shifts in the canvas location, a photograph of the canvas is taken, a user uses their mouse to click the corners of the canvas, then the canvas can be isolated from the camera’s field of view and oriented properly. There will still be a difference in the coordinate locations of where the robot intends to paint and where it actually paints. To close this gap, the robot paints 16, small dots evenly distributed across the canvas. The robot then takes a picture of the canvas and computes a homography to translate where the points were intended to be located to where they were actually painted. This homography can be used to ensure that the location that the robot paints on the real canvas corresponds accurately with the simulated canvas.

To calibrate the camera’s color and lighting perception, we use a color checker device with 24 colors standard in the photo and video industry. A transformation function is estimated between the perceived color checker values and the true RGB values, and is used for further perception of the canvas.

When a new brush is attached to the robot, the system needs to determine how far the brush protrudes from the end effector and what the brush is capable of producing when painting. A human operator uses the robot’s keyboard interface to lower the brush onto the canvas setting two values: the lightest touch of the brush on the canvas and the hardest brush on the canvas. Then, the robot makes a series of random brush strokes on the canvas, sampling uniformly from the three brush stroke parameters detailed in Section III-A. Examples of such random brush strokes can be seen in Fig. 14.

For brush stroke modeling, the starting and ending points of a stroke have fixed height  $h$  values at 0.2, which corresponds to a light touch of the brush. This is to keep the brush shapes consistent, since when the brush first presses to the page, it is impossible to predict which way the bristles will fan out. Setting a small  $h$  ensures that the bristles will not fan out drastically until the stroke moves along its trajectory and it consistently drags behind this trajectory.

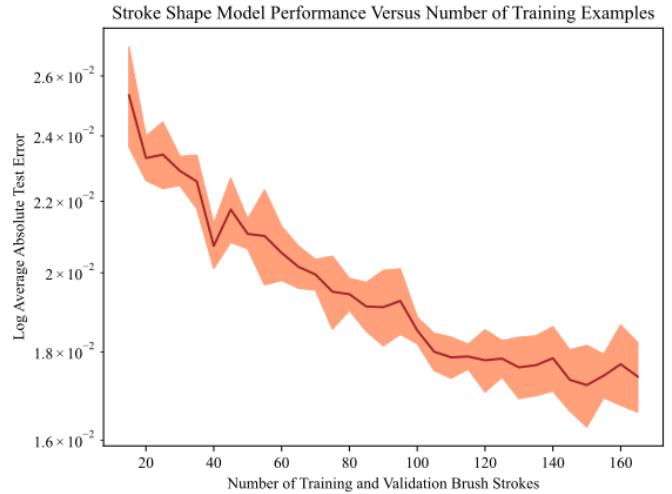


Fig. 15. Increasing the number of brush strokes used for training/validation of `param2stroke` from left-to-right versus the logarithm of the error of the model.

### B. Brush Stroke Shape Modelling

We investigated the number of random strokes needed to accurately train the `param2stroke` model which translates the stroke shape parameters to the appearance of the stroke. In Fig. 15, we plot the number of strokes in the training and validation sets versus the logarithm of the mean absolute error on the test set. The models were tested on a held out set of 42 strokes. 20% of the stroke-parameter pairs were used for validation to avoid over-fitting `param2stroke`, and we used five-fold cross validation to compute the log absolute error standard deviation to depict how much the error can fluctuate depending on which random strokes were used for training. The error generally decreases steadily with more strokes used for training, however, the improvement in performance is small after about 100 training/validation strokes.

We modeled three parameters that affect the shape of the stroke, however, there are many variables that we do not account for because of how challenging it is to perceive them. This noise is depicted in the error standard deviation of Fig. 15 as well as some qualitative examples of real versus modeled brush strokes in Fig. 7. Based on our observations, the most significant unaccounted for variable is the amount of paint on the brush. As seen in Fig. 14, some strokes are lighter or darker depending on how much paint was used. It is a technical challenge to estimate how much paint is on the paint brush and what part of the brush it is on, and we will investigate this in future work. Despite this noise, the model is still capable of learning the most likely appearance of the stroke. In some instances, as depicted in Fig. 7, the model’s predicted brush stroke looks more likely than the real brush stroke, since the real brush stroke ran out of paint. The model is capable of learning the full, continuous range of stroke shapes accurately, as seen in Fig. 6.

### C. Paint Colors and Mixing

FRIDA does not currently support automatic paint mixing. Instead, FRIDA relies on a human operator to mix a discrete

number of paints and provide them. The paint colors are determined during the initial planning algorithm step (Algorithm 1). The brush stroke colors can be clustered with the K-Means algorithm, then displayed to a human operator in a figure similar to that in Fig. 9.

Under certain circumstances, color discretization can significantly alter the perceived content of the painting, e.g., a very colorful simulated painting is generated but when discretizing to a small number of colors, the content is no longer apparent. For this reason, we frequently discretize throughout the optimization process to force the algorithm to operate under the constraint of the limited number of paint colors the user specified.

Painting with an unconstrained order of paint colors leads to two problems: (1) switching colors frequently which creates long execution times spent cleaning the paint brush between colors and (2) touching wet, dark paint on the canvas while painting with a light color and dragging it around to other canvas regions. To solve both of these problems, we order the brush strokes in the plan from light to dark. This minimizes the amount of time spent cleaning the brush, and eliminates the ability of the brush incidentally picking up wet, dark paint and spreading it around where light paint should be. Like discretization, ordering the brush strokes by color also affects optimization, so we frequently sort the strokes throughout the optimization process.