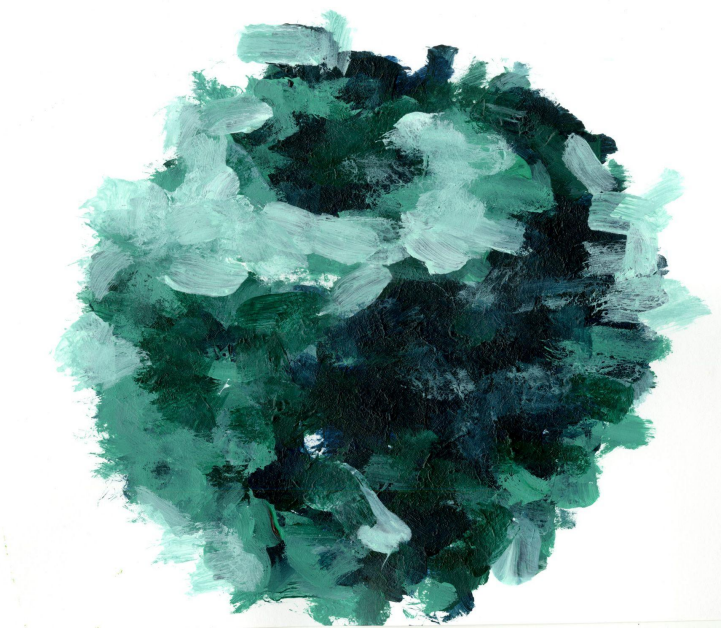
ART AND MACHINE LEARNING
CMU 2022 SPRING
FINAL PROJECT

Isn’t It An Apple?



Jiaying Wei
Master of Science in Computational Design, School of Architecture
Zhenfang Chen
Master of Science in Computational Design, School of Architecture
Guanzhou Ji
PhD in Building Performance and Diagnostics, School of Architecture

Date: 5/1/2022

DESCRIPTION

This project is based on the techniques of robotic painting, with the concept of translating an image into a given style and later painted by a robotic arm. In still life painting, form finding and color mixture are two key parts. Apple is an important part of elements in mythology and tales, as well as in human art history, making apple to be a symbolic object beyond ordinary fruit. For example, apple is the eating of the forbidden fruit by Adam and Eve. In the area of machine learning, apple is commonly used in style transfer, image synthesis, and 3D generation. Thus, we choose an apple as the target object. The research question in this project is "To what extent, do you think it is an apple?"

Concept

We are interested in the idea of robotic painting because we think how a robot executes a command (draw on stroke) and hundreds and even thousands that after is like following a grammar, in a computational way. The painting by a robot is "accurate", as the robot follows all the commands that it receives. But there still exists some space for "ambiguous", since a lot of parameters can affect the results, such as the height of the brush, the angle the robot holds the brush, or color mixture.

"Appropriation is the intentional borrowing, copying, and alteration of existing images and objects. A strategy that has been used by artists for millennia, it took on new significance in the mid-20th century with the rise of consumerism and the proliferation of images through mass media outlets from magazines to television."[1]

Campbell's Soup Cans (1965 colored version), by Andy Warhol, reminds us of mass production, reproducing, juxtaposing, and repeating essences of what an industrial robotic arm is used for. In our case, we are not going to make criticism, rather to bring up our question: "To what extent, do you still think it is an apple?", especially we have applied various styles and sampling techniques to the target image to get different results.

[1] Available at: https://www.moma.org/learn/moma_learning/themes/pop-art/appropriation/.

Figure 1. Campbell's Soup Cans (1965 colored version), by Andy Warhol[2]

Figure 2. Apple Style Transfer

[2] https://branditative.wordpress.com/2012/07/14/andy-warhol-and-his-muse-the-campbell-soup-can/

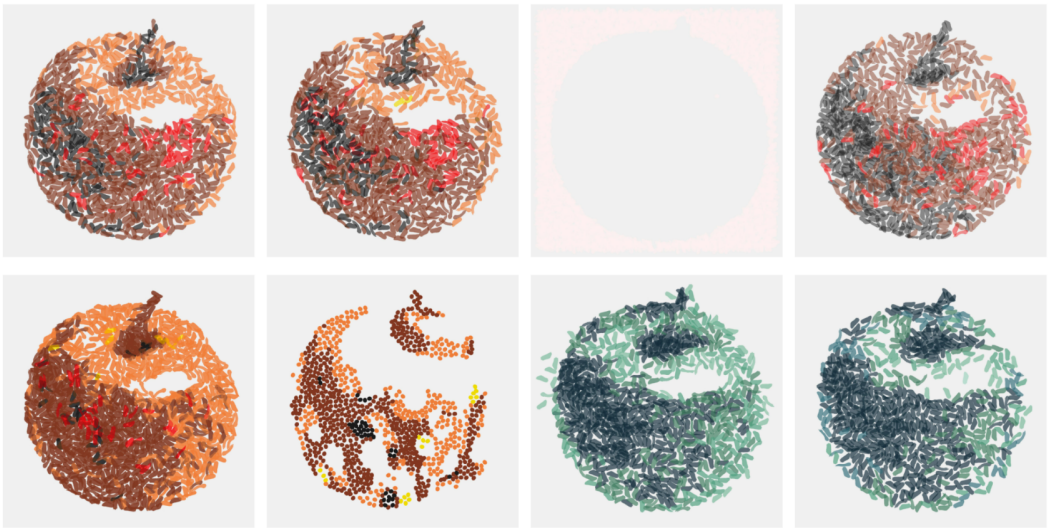


Figure 3. Stroke from Style Transfer

After seeing several paintings by our robotic arm, we get an interesting observation: no matter what kind of techniques we are using, there are always differences between the target object/image and the generated/produced object/image. We cannot help ourselves from thinking about what are the gain and loss of an interpretation/implementation? What are the affordances of each interpretation/implementation? Do we lose the essence of the thing during interpretation/implementation?

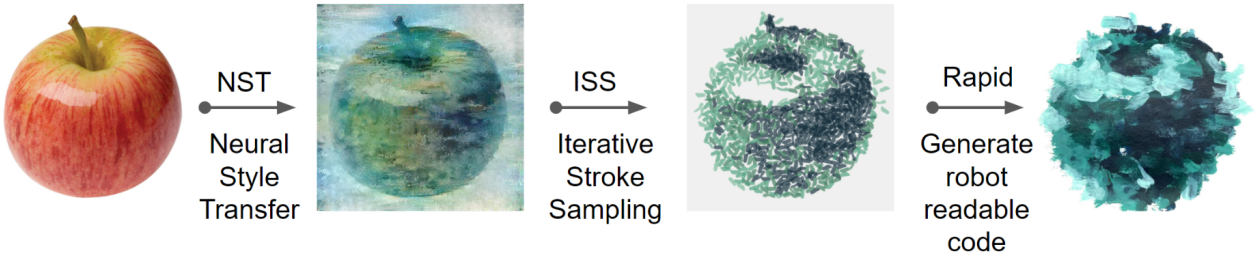


Figure 4. NST, ISS, and Rapid Workflow for Apple

Technique

The base model, smART[3], developed by a team at TU Vienna, employs the workflow of Neural Style, Iterative Stroke Sampler, and rapid code generator in python scripts. NST uses 16-layer Deep CNN by Simonyan and Zisserman, 2014. The Iterative Stroke Sampler(ISS) is based on the preset stroke Numpy file. While the workflow to produce rapid code is organized by one main python file, a set of robots, color configurations and rapid file writer.

To adapt the base model to our setup, we organized the strokes to a single brush setting, disabled waterwash for a better mixture in color gradient and reduced replenish frequency. Adjustments are designed to follow more humane habits in color treatment.

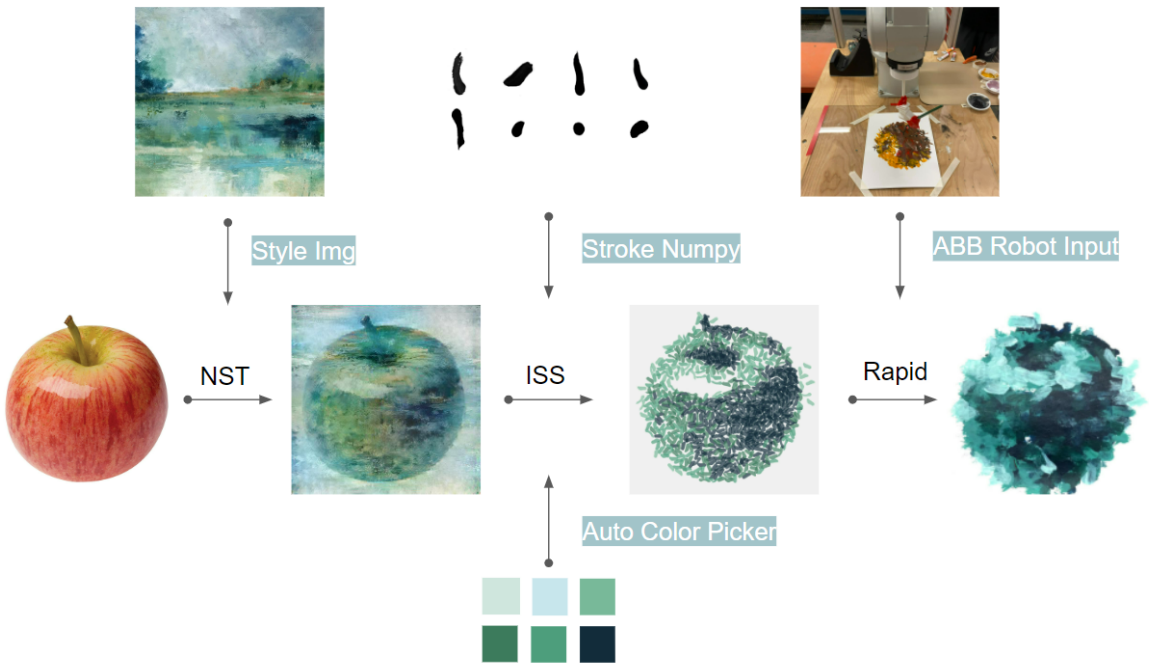


Figure 5. ML Robotic Painting Workflow Overview

[3] https://github.com/EricSteinberger/smART

Due to the lack of pre-boarding color mixing process, we decided to manually prepare color mixture with human labor by eyeballing the color swatch. Base color covers a widest range of basic colors but does not necessarily render best color combination quality, while black in the default setting needs to be carefully leveraged due to its strong desaturation. A limited number of major colors in the transferred image is picked, and their RGB colors are printed for humans to work on mixing the colors in order.

Figure 6. Comparison between Eyeballed Color Wwatch and Auto Generated color swatch.

Process

1. Code Testing

The first part of this project is style transfer and stroke generation. We set up an environment for smART code to generate the Rapid code for robotic arm, which has been discussed in the aforementioned sections. In order to simplify the understanding of stroke behavior, we ‘ve disabled all other color parameters except black, and chose only one numpy stroke to see how density and stroke perform in the short-time simulation.

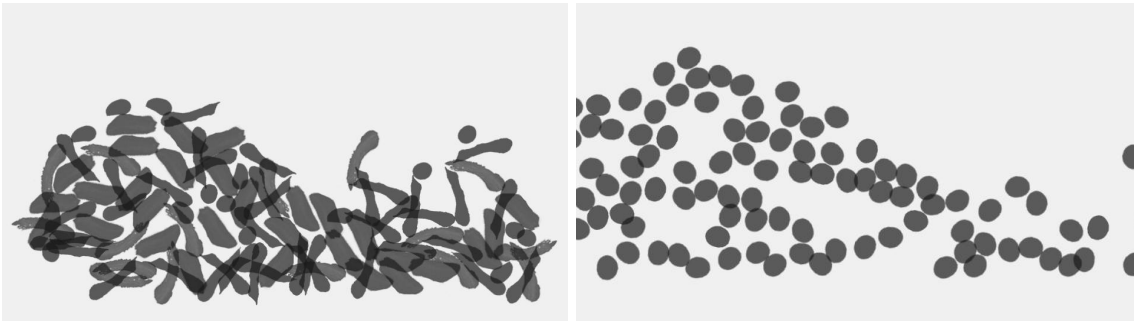


Figure 7. Stroke Examples

Although simulation painting does not indicate an accurate prediction in physical range, it provides an overview of stroke overlap that hints at the mixture of colors and the thickness of

painting. By learning how to analyze the simulation painting, we avoided a lot of traps and saved a considerable amount of time for physical testing. For example the following shows an excessive amount of stroke and deficient amount of stroke, which ultimately give us a conclusion that controlling the number of strokes around 600 to 800 would be optimal for a canvas size in 200mm by 200mm.

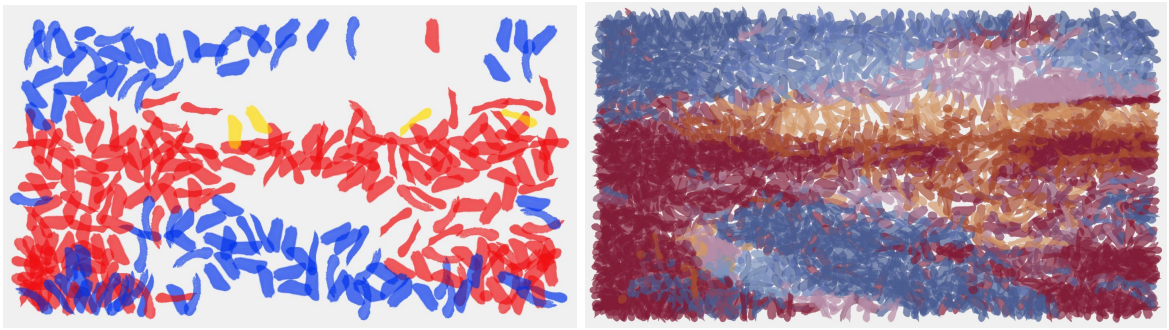


Figure 8. Comparison of Stroke Density

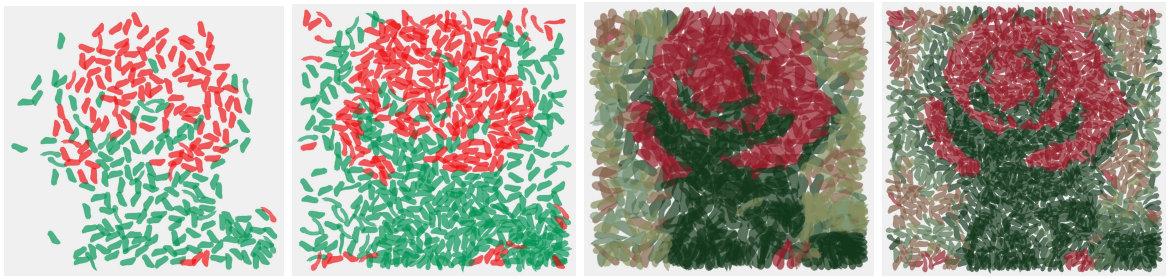


Figure 9. Test in Irregular and contourless still life like Rose, with a variation in ISS iterations and color settings

Although we found that the 8 basic preset gradients are generally able to satisfy the formation of Image composition. It is rigid in color mixing as well as color transition, while a lot of aesthetic appreciation in paintings rely on artists' selection of colors. Take historical precedents like the chiaroscuro style proposed by Rembrandt, and the fashion of impressionism, many of the famous art masterpieces heavily rely on insights in colors. Therefore we decided to create a customized color swatch picker and print out a list of RGBs as some of the most prominent colors in the composition.

```
(smART) C:\Users\vinaw\smART_zf>python main.py --target-image-filename apple.jpg
[[207, 230, 221], [199, 230, 236], [120, 185, 153], [60, 123, 92], [77, 158, 124], [18, 44, 58]]
'Color0', 'Color1', 'Color2', 'Color3', 'Color4', 'Color5']
```

Figure 10. Printed Color List

Figure 11. A comparison of Rose image painted by default color and customized color swatch.

2. Robot Setup and Implementation

We've gone through a long learning process of robot setup, simulation, debugging and testing process, about 25 hours before assembling the end-effector and loading the brush. The IRB120 model by ABB has its own simulation platform but requires self-defined tools for its orientation coordinates and rapid files. However the licensure of robot studio and server has produced many difficulties in running a whole process of simulation prior to physical painting. Due to the limitation of manual mode, we were only able to upload the mod file through Flexpendant's USB portal that has a limitation of 65,000 lines of rapid codes, and larger files would even result in freeze screen and hard restart of the robot. This gave us a huge challenge to modify and produce concise yet effective rapid code to adapt to our own needs. In the latest version of our code organization, we reduced the category of brush from B6, B12 to only B6, and changed color dipping from once per stroke to one per 5 strokes, which reduced lines of code by almost 75% and performance time by 60%. In addition, to ensure the successful implementation of the painting, we tested individual movement of stroke functions to estimate the range of the painting, the speed of the robot(which was set to be 50% of full speed) and cleaning wait time. In the resulting adjustment, a painting with around 900 brush strokes would take around 3 hours to complete.

The second part is to implement robotic arm to achieve the real paintings. We first designed a brush holder to mount the brush to the robotic arm. The entire holder consists of a base with space for screws and a holder component for holding the paintbrush. The holder is extendable when the robotic arm moves the brush for drawing (Figure 6). Then, we tested different sized brush and shape of stroke (Figure 7), as well as the color combination (Figure 8) In order to eliminate water dripping conditions that happened in Figure 6, we decided to use thick color texture to increase the coverage.

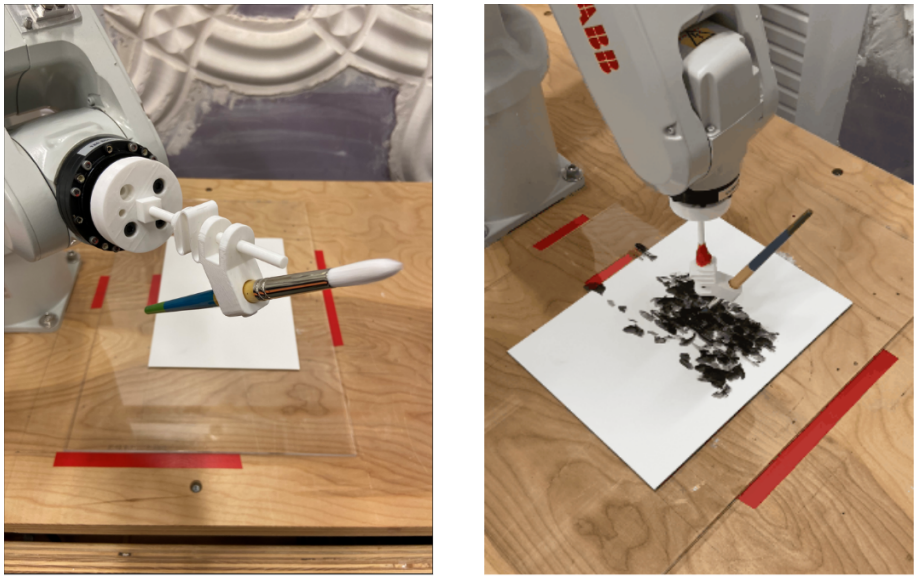


Figure 12. Brush Holder

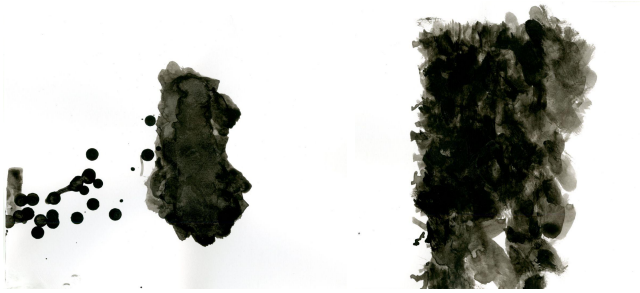


Figure 13. Results of Brush Holder Test

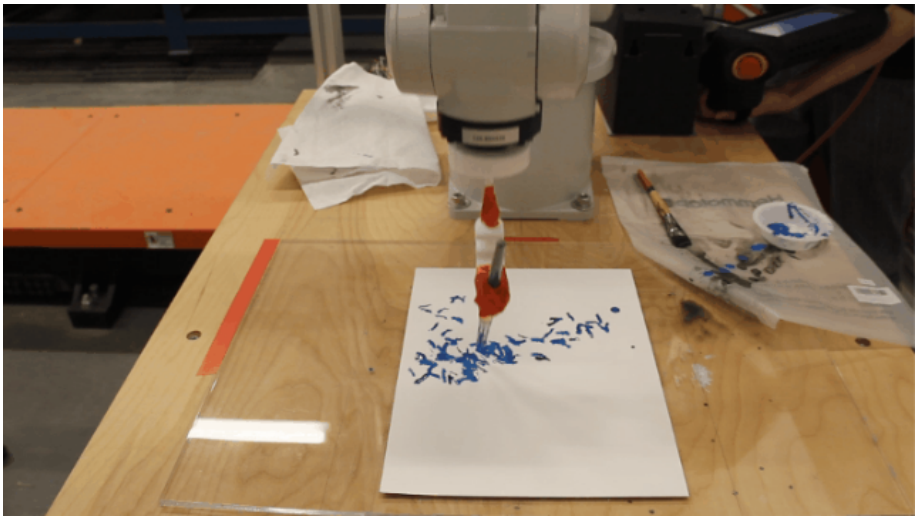


Figure 14. Stroke Test

Figure 15. Result of Stroke Test

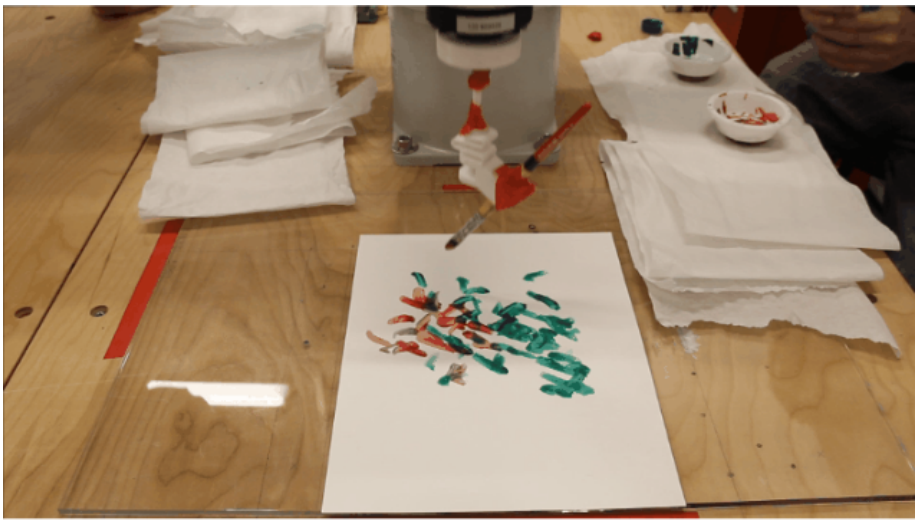


Figure 16. Paint with Two Colors

Figure 17. Result of Two Color Combination

Figure 18. The same exact rapid code using different sizes of brushes that exhibits dramatic difference in overlaps and spacing, leading to a preference towards larger and thicker brush.

Reflection

[Stroke Number vs Stroke Size] When the stroke size is small, it requires a high number of strokes to fill in the content on the canvas. By contrast, using a large sized stroke could reduce the number of strokes; however, it also reduced the resolution of the paintings, as the details will be missing.

[Stoke Path] We tested path stroke and point stroke as two primary means for robotic paintings. The former could leave more changes in direction of movement and sharp edges, it also causes the edge of the apple blurring; the latter point stroke relies on the density of points to shape the color block and geometry, which has a relatively higher control for the details and edges.

[Color and Mixing] We get the color from the smART code; however, the brush will mix those undried strokes with the new stroke, when changing the colors. Also, the level of water in the paintings also impacts the results. We noticed some unexpected color mixtures during the real paintings, and this situation normally cannot be simulated accurately.

[Style Transfer Parameters]Initially we assumed that style transfer image that contains a similar color-palette with the original image would produce a more reliable result, like rose should employ style images with primary red and green(Figure 19), whereas unsuccessful performance would be result in unpredictable outcome(Figure 20) However later we concluded that as long as the darkness variation or color range satisfies its minimal requirement.

Figure 19. Successful implementation of style transfer

Figure 20. unsuccessful implementation of style transfer with a style image that does not exhibit contrast in darkness or color gradient.

RESULT

In the final results, we explored four apples with different parameter settings, which are path stroke, color swatch, point stroke, stroke density. The path stroke option (Figure 21) generates an apple that has flexible stroke movement and some extra strokes show in the apple's outlines. Similarly, the color swatch option (Figure 22) replaces the previous colors with colors in a higher saturation. Although it still uses path stroke, due to the water level in the paintings, two apples still show some variations in color mixture and texture.

Figure 21. Path Stroke

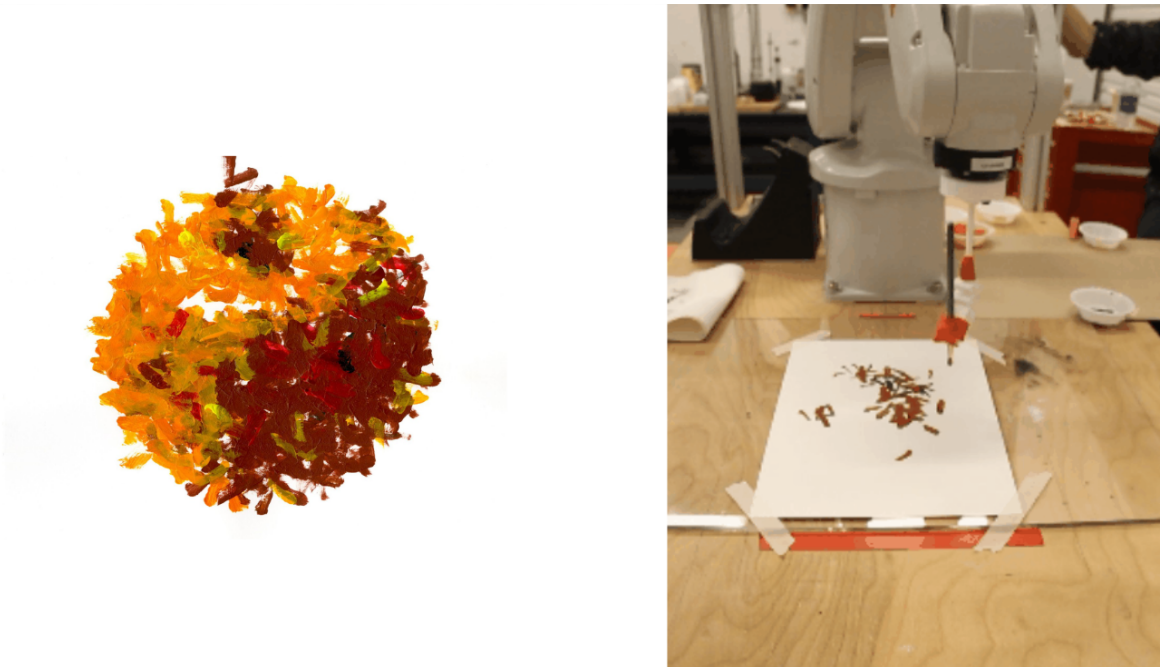


Figure 22. Color Swatch

The point stroke option (Figure 23) is using points to shape the apple. For each stroke, the point covers less area than the path stroke, which leaves more blank areas than the previous two apple paintings. In the last stoke density option (Figure 24), we used a thicker brush and the result has less blank area than previous three apples and meanwhile the painting still shows clear distinction for apple's different surfaces.

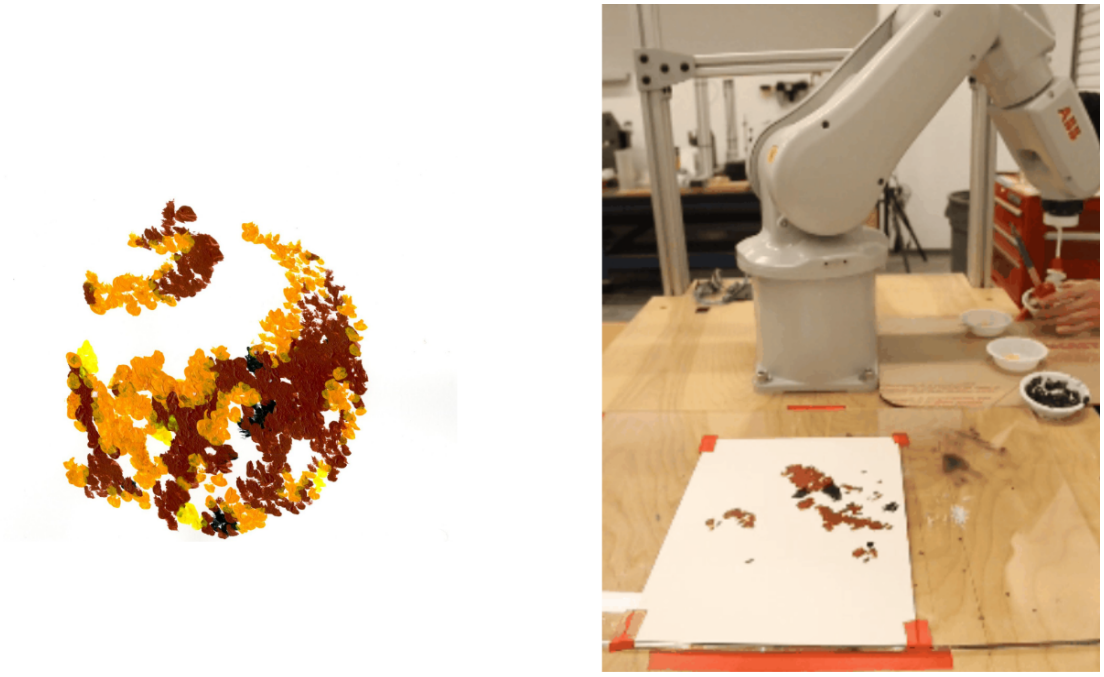


Figure 23. Point Stroke

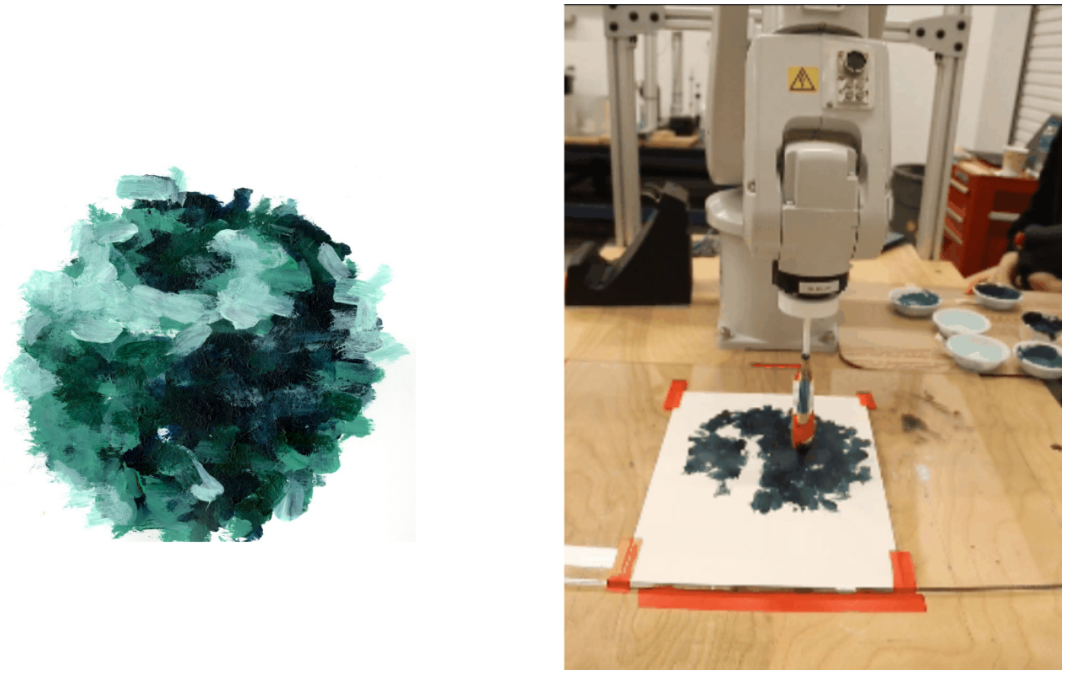


Figure 24. Stroke Density

CODE

The style transfer and stroke generation are based on smART[4], some modification on parameters and rapid generators, and also a major revision in color configuration file that extracts customized color for individual images. we also used RobotStudio[5] to simulate robotic movement.

[4] Available at: https://github.com/ericsteinberger/smart.
[5] https://new.abb.com/products/robotics/robotstudio