ARTISTIC RENDERINGS FOR IMAGES AND VIDEO

Alexander Christoforides
chrisa4@rpi.edu

Andrew Aikens
aikena@rpi.edu

April 3, 2019

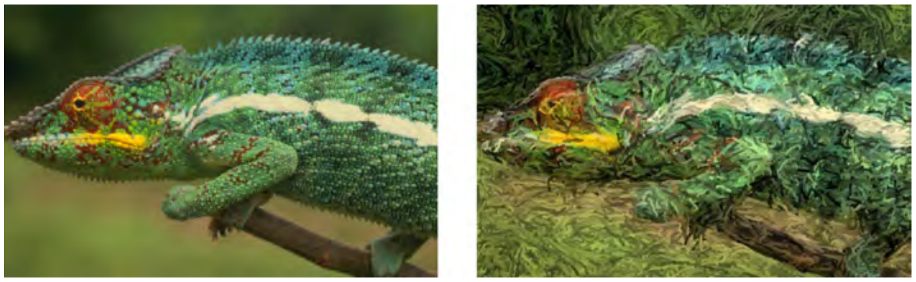
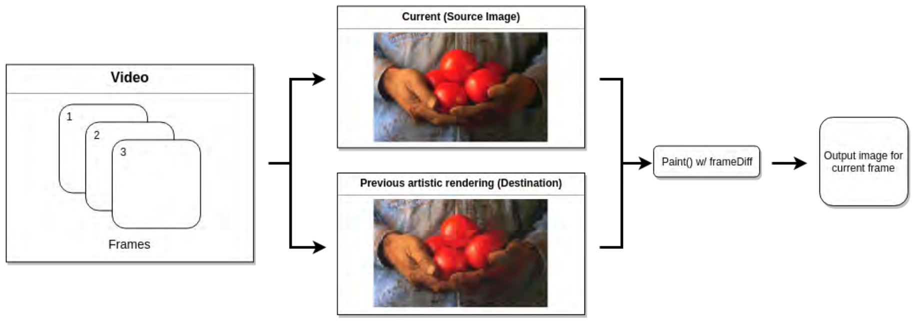


Figure 1: Artistically-rendered source image

1 Introduction

A major focus of computer graphics research is the synthesis and production of realistic images from three-dimensional scenes and traditional two-dimensional images. Contrary to this trend, this project aims to tackle the issues of producing non-photorealistic renderings of two-dimensional images in various artistic styles and combining these renderings into a video where visual flickering is significantly reduced. A common problem that many identify when addressing the topic of creating artistic video renderings is that visual flickering occurs between frames, significantly affecting the quality of the synthesized result. To handle this, we propose implementing an algorithm based on difference-masking to efficiently and significantly reduce visual flickering between frames of the rendered video, thus improving the temporal coherency of the overall result.

To accomplish this, we need to devise a maintainable and extendable framework to transform the proposed algorithms into a functional application which can produce artistic renderings of both static images and videos with the assistance of 3rd-party modules (OpenCV, numpy).


Video
1
2
3
Frames
Current (Source Image)
Previous artistic rendering (Destination)
Paint() w/ frameDiff
Output image for current frame
Figure 2: Artistic rendering pipeline for videos

2 Summary of External References

Below you can find summaries of relevant academic papers discussing the required background knowledge and algorithms required to artistically render static images and videos.

2.1 Paint by Numbers: Abstract Image Representations (Paul Haeberli, 1990) [1]

This paper provides a wealth of required background knowledge to understand the dissertation by Aaron Hertzmann and to implement the proposed algorithms. It additionally introduces the topic of non-photorealistic rendering and a brief summary of the use of brush strokes by actual Impressionist artists. This paper puts forth a method of creating abstract image representations of realistic images as a collection of brush strokes which can be iteratively applied to generate a new image. Not only does Haeberli describe how the direction, size, and brush type affect the resulting image, but he notes how introducing random noise in the brush stroke locations, color, size, and direction can have a significant effect on the output, possibly producing a more desired result. These noise modifications are adapted in the 2001 dissertation in the `paint()` and `paintStrokes()` functions to facilitate stroke changes during the duration of the program creating a stroke in the canvas.

2.2 Painterly Rendering with Curved Brush Strokes of Multiple Sizes (Aaron Hertzmann, 1998) [2]

This paper will prove to be useful when we implement the `paint()` and `paintStrokes()` algorithms in our application. It provides informative commentary on the limitations of their current algorithms and additionally provides insights into the theory behind modeling a brush stroke as a B-spline. While our main focus in the project will be the non-B-spline implementation of the artistic rendering algorithms, an optional feature that we would like to include, if time permits, is a migration from the existing implementation to the B-spline implementation so that we can attempt to further improve our temporal coherence in generated videos with the concept of optical flow mentioned in the dissertation. The paper also discusses parameters that they found which visually mimic actual art styles that exist today (Impressionist, Expressionist, Colorist Wash, and Pointillist), but also address the fact that trying to "solve" for other parameters which emulate other existing art styles is quite difficult.

2.3 Algorithms for Rendering in Artistic Styles (Aaron Hertzmann, 2001) [3]

Our focus will mostly be on chapters three and four of this paper. Chapter three introduces the required algorithms to be able to generate the artistic renderings. This is broken into two parts: `paint()` and `paintStrokes()` (which we will be using for our implementation). These functions facilitate the layered-based painting of an image over multiple passes. For the paint function, this relates to the paint brush sizes for the target painting style. This function is also highly configurable, with multiple parameters to create many different painting styles. During experimentation, Hertzmann defines methods such as Impressionist, Expressionist, Colorist Wash, Pointillist, and Psychedelic. There are a couple of challenges that arise when attempting to handle videos, though. The algorithm provided in the paper best works for videos with a frame rate of 10-15 frames per second. This means that the video will need to be down sampled to prevent flickering. To combat this, optical flow examples are provided in this paper to better enhance temporal coherence.

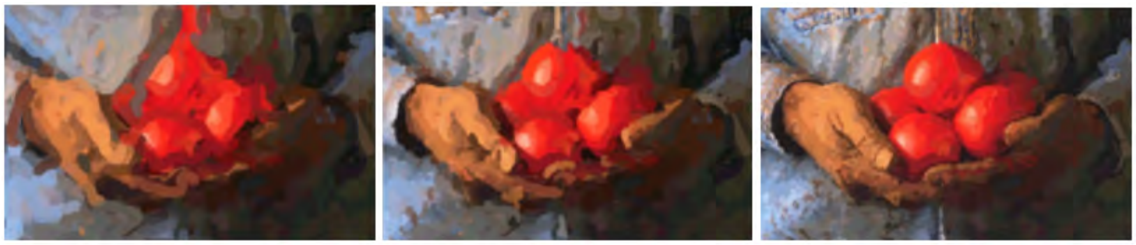


Figure 3: Layered paint stroke example (from iteration 1 to 3)

3 Examples

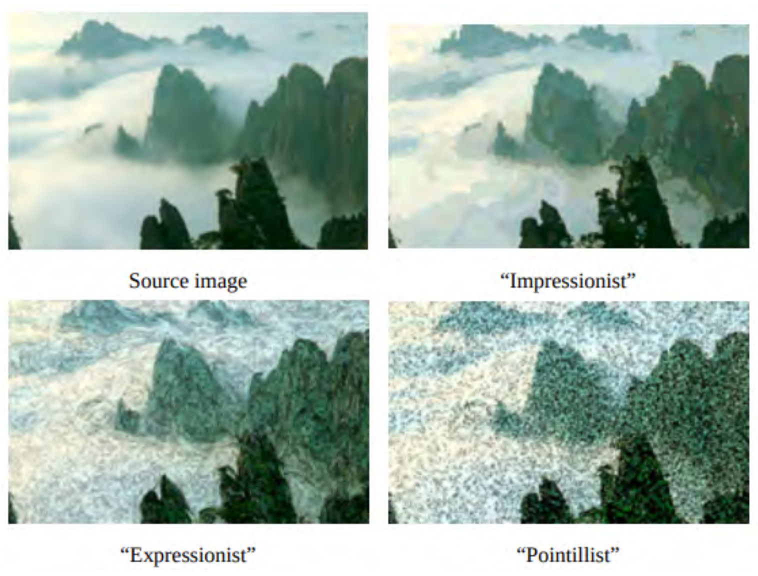


Figure 4: Static image examples

For images, we plan to present the different paint styles as laid out by the paper. This includes Impressionist, Expressionist, Colorist Wash, Pointillist, and Psychedelic painting styles. In addition to this, we would like to display intermediate steps when rendering the images (as shown in figure 3). This can be used to better visualize and explain the algorithm to the audience.

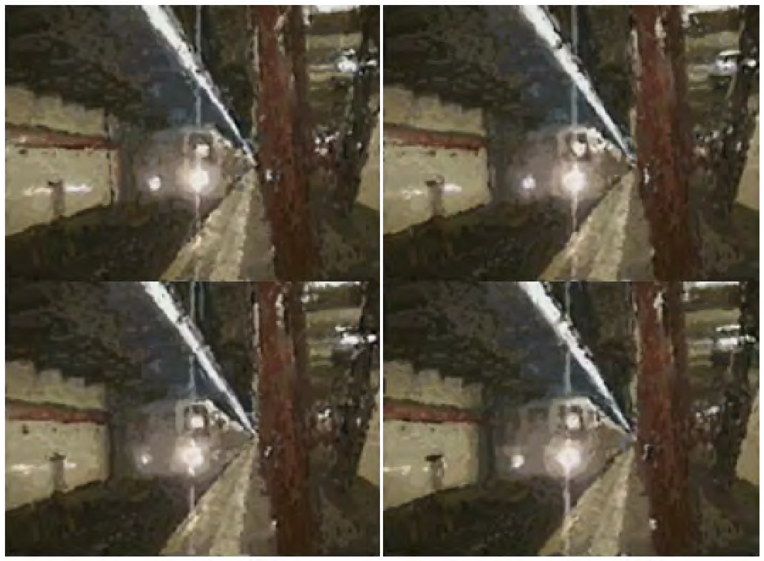
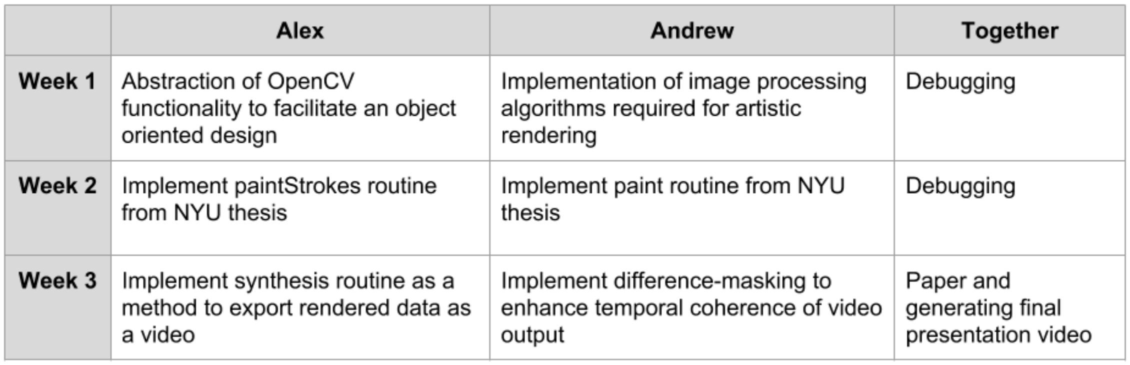


Figure 5: Video frame example

For the video part of the presentation we are planning to have a compressed playable video that we could show during our demonstration. This could possibly be a music video or something more static with few moving objects in the scene. We are going to investigate different videos from little variance from frame to frame to videos with large variance. From our investigation, we will choose a video that demonstrates the algorithm's capabilities while producing minimal flickering for the audience.

Note: Example screenshots taken from: Algorithms for Rendering in Artistic Styles (Aaron Hertzmann, 2001) as described above.

4 Timeline



	Alex	Andrew	Together
Week 1	Abstraction of OpenCV functionality to facilitate an object oriented design	Implementation of image processing algorithms required for artistic rendering	Debugging
Week 2	Implement paintStrokes routine from NYU thesis	Implement paint routine from NYU thesis	Debugging
Week 3	Implement synthesis routine as a method to export rendered data as a video	Implement difference-masking to enhance temporal coherence of video output	Paper and generating final presentation video

Figure 6: Timeline

Core features

- Artistic renderings of images.
  - In order to accomplish this task we will have to implement the data structures outlined in figure 7. These structures make image processing uniform operations.
- Artistic renderings of videos with difference-masking technique to improve temporal coherence.
  - The pipeline is shown in figure 2. There are a couple differences between images and video for the given algorithms provided in the paper. A difference masking must be performed from frame to frame in order to limit the amount of flickering (effectively improving temporal coherence).
- Reusable and modular code implementation.
  - This enables others to be able to pickup and readily utilize and extend our code. This abstraction will make the implementation easier to digest.

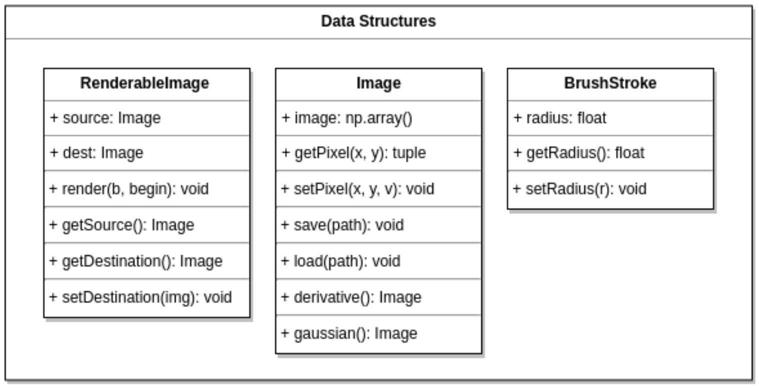


Figure 7: Implementation data structures

Future/Optional work (if time permits):

- Optical flow implementation which will further improve temporal coherence for our video output.
- Store B-Spline representation of brush strokes. This is required in order to modify the image to implement the "relaxation" method in chapter 6 of the provided thesis.

References

[1] HAEBERLI, P. Paint by numbers: Abstract image representations. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1990), SIGGRAPH '90, ACM, pp. 207–214.

[2] HERTZMANN, A. Painterly rendering with curved brush strokes of multiple sizes. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 453–460.

[3] HERTZMANN, A. Algorithms for rendering in artistic styles. Tech. rep., NYU, 2001.