Making Impressionist Paintings Out of Real Images

Aditya Joshua Batra

December 13, 2017

Abstract

In this project, I have used techniques similar to those used by Patrick Callahan [1] and Peter Litwinowicz [2], in order to render actual photos as impressionist paintings. Because impressionism is defined by thin brush strokes that capture the essence of a scene rather than the fine details, the main challenge is replicating the imperfections that come with manual painting. Ultimately, a base method is implemented that draws random brush strokes with a user defined radius and length using data from the original image to recreate photos in an impressionist style. On top of this base, edge detection and edge clipping is added to maintain structures from the original photo in the impressionist rendering, and strokes are oriented normal to the gradient of the original image to make the strokes look more natural. Finally, random perturbations are added to simulate human imperfections in painting. After going into detail about how to render an impressionist painting out of a real image, this paper describes a technique for rendering images in a consistent way across an image sequence or video and delves into the idea of automatically adjusting the radius of brush strokes based on edge data.

Introduction

Impressionism was an art movement that began in the nineteenth century and involved using thin quick strokes of paint to capture the essence of an ordinary subject rather than precise details. Impressionism also put focus on properties such as minimal color mixing and the interplay of natural light. Non photo-realistic rendering is an area of Computer Graphics that involves creating certain styles of art digitally, and impressionism was one of the styles that was most heavily studied. One benefit of attempting to render impressionist paintings from real images is that one gets the natural interplay of light from the original image that was so important in impressionism. The main hurdles with creating impressionistic paintings from real images is that one had to make the painting look spontaneous and varied rather than having a uniform, precise style that would be typical of something that is computer generated. The methods that are described in the next section detail attempts to emulation the human imperfections and somewhat random variations of manual impressionist paintings and look into ways of keeping painting style consistent across images in a sequence.

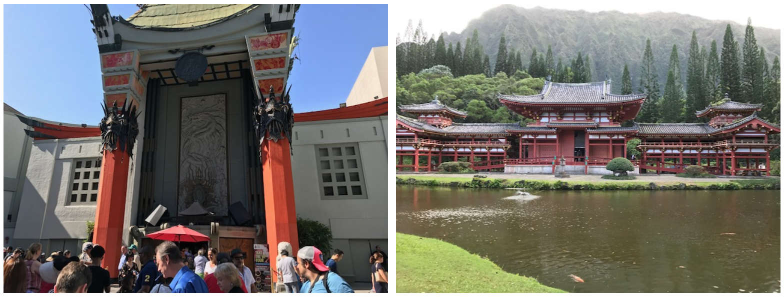

Figure 1. Original Photos. The picture on the left is a photo of the Chinese Theater in Hollywood. The picture on the right is a photo of a temple in Hawaii. These photos will be used as examples to show the various impressionist methods that have been implemented.

Methods and Implementations

The base algorithm involves randomly selecting image centers from the given image. The color is extracted from this image center, and a box is formed around this center based on a user defined radius and length and an orientation that is randomly selected. The pixels in this box are colored with the color extracted from the image center to create an image stroke. Image strokes are added to a blank canvas until the canvas is full. This canvas is an impressionistic rendering of the original image, but the strokes are drawn in the same direction and colors escape from their boundaries distorting edges in the original image.

The next addition to this base algorithm is edge clipping. The given image is first converted to gray scale and then run through a canny edge detector that first Gaussian filters the image with a sigma of 2. Given a set of edges in the image, the stroke generation code first traces the stroke from the chosen center and stops the stroke whenever it intersects with an edge in either the forwards or backwards direction. The results from this edge clipping show that edges are mostly preserved in the impressionistic renderings,

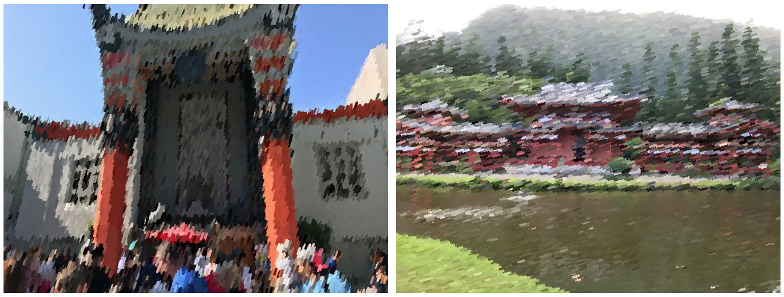

Figure 2. Base Algorithm. The picture on the left is an impressionist rendering of the theater using the base algorithm, and the picture on the right is a rendering of the temple. The temple's brush strokes have a smaller radius than the theater to capture detail accurately.

but there remains the problem that all the strokes are oriented in the same direction. Thus, the image still looks too processed.

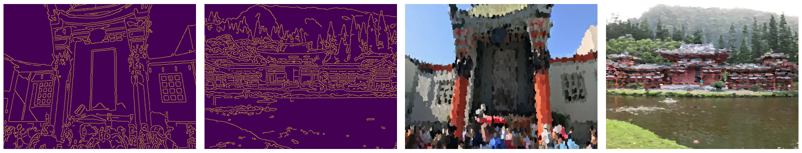

Figure 3. Edge Clipping. The renderings now use their edges as seen in the left two pictures to clip brush strokes.

Next, attention is turned towards the orientation of the strokes. To get natural looking strokes, the x and y gradients of the given image are calculated. The orientation is then calculated for each stroke center by taking the arc tangent of the y-gradient divided by the x-gradient and adding 90 degrees. If both the x and y gradients are 0, the orientation is just chosen randomly.

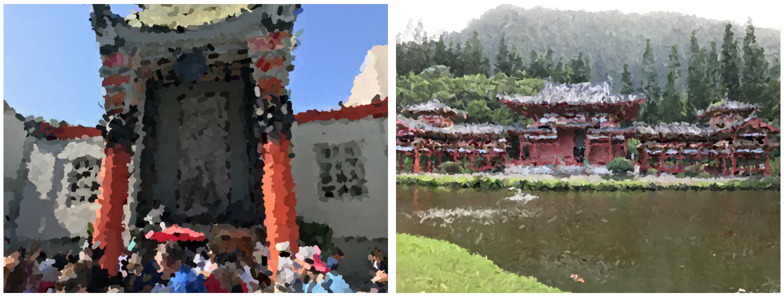

Figure 4. Gradient Normal. The brush strokes are now oriented normal to the gradient of the image, making the renderings look more natural.

Finally, random perturbations are added to simulate human imperfections. The r,g, and b values of the color of each stroke are modified by a value randomly picked between -15 and 15 divided by 255, the intensity is multiplied by value randomly picked between .92 and 1.08, and the orientation is modified by a value between -15 and 15 degrees. The results look more natural as a result.

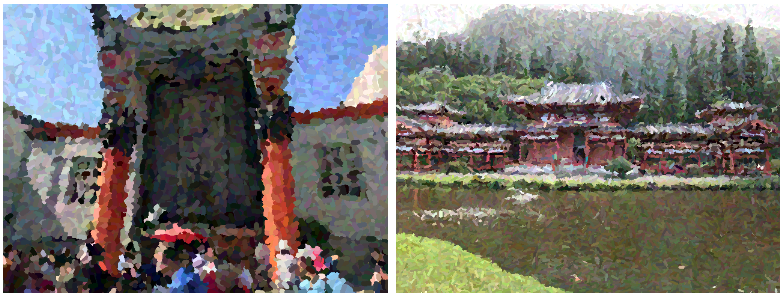

Figure 5. Random Perturbations. The renderings now include random variations to make them look more natural.

After rendering an impressionist painting properly, the next focus becomes attempting to get a consistent rendering across images in a sequence. The way to do this is to use optical flow to move brush strokes from one image to the next. Given two frames, the optical flow is calculated using the equation

$$0 = \frac{\partial I}{\partial x} * V_x + \frac{\partial I}{\partial y} * V_y + \frac{\partial I}{\partial t} \tag{1}$$

where the partial derivatives refer to the gradient with respect to x, y, and time. Using the velocities of x and y, strokes can be shifted across frames. However, this could leave blank spots.

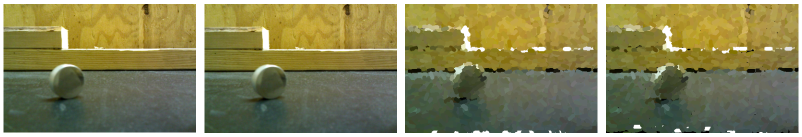

Figure 6. Sequence Rendering. The two pictures on the left are part of a video, and the two pictures on the right are impressionist renderings linked where the second rendering consists of strokes shifted from the first rendering.

Finally, noticing that images with more clutter need smaller strokes to be rendered properly, I devise a method that automatically adjusts the radius of brush strokes based on the number of edge pixels in an 8 pixel neighborhood of stroke centers.

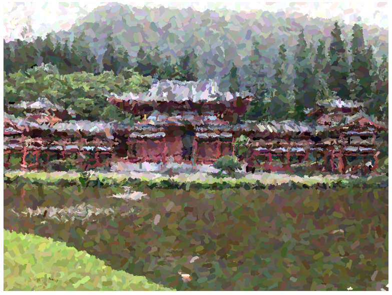

Figure 7. Automatic Radius. The picture shows a rendering done where the radius of brush strokes is automatically adjusted.

Future Work and Conclusion

Ultimately, the techniques described above make for a rendering that looks natural to the typical observer. For future work, perhaps more effort could be put into automatically adjusting the size and length of strokes based on facts about the given image.

Acknowledgments

I am grateful to Professor Efros and Jun-Yan for putting together this amazing class. This was a great learning experience, and I had a lot of fun.

Appendix

More images and info about the processes I used are available on my project website.

References

[1] Patrick Callahan. *Fake Impressionist Paintings for Images and Video*. Research Paper, 2010

[2] Peter Litwinowicz *Processing Images and Video for an Impressionist Effect* SiGGRAPH, 1997