Rendering 360º Images into Paintings

Allen Dufort, Gabby Litterio, Muskaan Patel

Brown University, CSCI 1290, Fall 2023

Abstract

For the final project of CSCI 1290, we rendered 360° panoramic images into artistic stylizations using three methods: a patch-based texture patching algorithm and two different neural network style transfer models. Our neural network methods separate the content and style of images, allowing us to recombine the content of globe panoramas with the style of any given artwork patch. We derived neural representations of content and style from a pre-trained convolutional neural network and optimized novel images to match these representations. To address challenges of working with wide-field images, we introduced masking techniques to focus stylization on regions of interest (i.e., the sky) while preserving details elsewhere. Our project also provides two different methods of masking: one involving OpenCV libraries and the other involving manually sculpting out the mask using Adobe Photoshop. We compared results from our methods and found that deep convolutional neural networks best capture artistic styles across the full panoramic field. The ability to render immersive scenes in distinct visual styles opens new possibilities for studying perception and the generation of virtual artistic experiences.

KEYWORDS: Neural Style Transfer, Texture Transfer, Convolutional Neural Networks, Patch-Based Algorithm, TF-Hub, Panoramic Images, Immersive Scenes, Image Masking, Globe Panoramas, Insta360 Photos

1. Introduction

Our objective was to convert complete 360º images into artistic renditions inspired by renowned painting styles. We applied stylizations to distinct areas of interest, such as the sky, while maintaining the authenticity of other segments within the original image. Beyond the stylized transformations, we introduced an interactive dimension, akin to Google Street View, enabling users to rotate the image a full 360 degrees for immersive exploration.

To capture our 360º images, we utilized the Insta360 One camera—an invaluable tool enabling us to encapsulate immersive scenes. Our goal wasn't solely to create static paintings; rather, we aimed to provide an interactive viewing experience, enriching user engagement with the dynamically transformed panoramas.

This project is built on influential works in neural style transfer, drawing inspiration from "A Neural Algorithm of Artistic Style" by Gatys et al. [2], Efros et al.'s "Image Quilting for Texture Synthesis and Transfer" [1] and "Exploring the Structure of a Real-Time, Arbitrary Neural Artistic Stylization Network" by Ghiasi et al. [3]. Our project delves into both style and texture transfer methods. This involved implementing neural networks, including Convolutional Neural Networks (CNNs) and the TensorFlow model TF-Hub, and employing a patch-based texture transfer algorithm.

The choice of methodologies was driven by the need to address challenges inherent in working with wide-field images. Furthermore, our exploration includes the introduction of masking techniques to focus stylization on specific regions of interest, a crucial step in preserving intricate details, such as the sky, while enhancing overall visual appeal.

In this paper, we will present our findings, comparing results obtained through different methods. Additionally, we will discuss the specific techniques we used and challenges encountered, and the valuable outcomes derived from our exploration. We aim to showcase the artistic transformations achieved and begin to analyze the efficacy of various approaches for rendering immersive scenes in distinct visual styles.

2. Methodology

2.1. Patch-Based Texture Transfer Algorithm

Our methodology involves applying the Image Quilting for Texture Synthesis and Transfer method introduced in the SIGGRAPH 2001 paper by Alexei A. Efros and William T. Freeman [1]. This technique enables both texture synthesis—creating images of various sizes from small samples like grass—and texture transfer—rendering images in the style of others, such as re-rendering an image like Abraham Lincoln in a different artistic style.

In our patch-based algorithm, we re-render the input image using texture samples from the style image. Each integrated sample patch in our synthesized result considers two key constraints: coherence with existing segments (used in texture synthesis) and alignment with the desired re-rendered image. We adjust parameters, including α, in Equation 1, computing patch correspondence via grayscale intensity-based sum of squared differences (SSD). Through iterative refinement involving reduced tile sizes and adjusted α, we improve the texture transfer quality, optimizing the algorithm's performance.

We run through 2 iterations of this while decreasing the tile size and adjusting α each time to get the best results. The error term in each iteration is Equation 1, where $\mathrm{prev\_error}$ is the error calculated in the previous iteration and $\mathrm{corr\_error}$ is the value calculated by SSD.

$$\mathrm{error} = \alpha * (\mathrm{overlap\_error} + \mathrm{prev\_error}) + (1 - \alpha) * \mathrm{corr\_error} \quad (1)$$

2.2. Neural Network Style Transfer Algorithms

a CNN Model

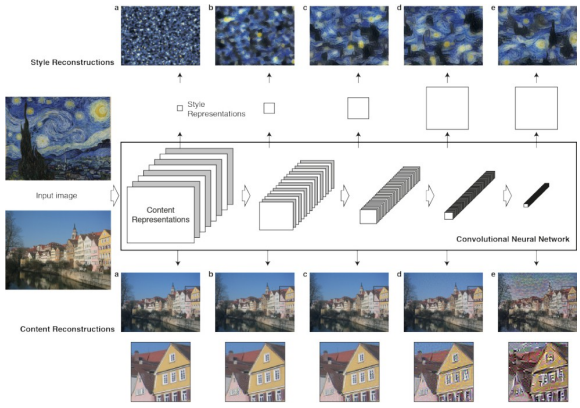

Figure 1. Diagram of the CNN model [2]

In Gatys paper[2], the input image is reconstructed from layers 'conv1_1' (a), 'conv2_1' (b), 'conv3_1'(c), 'conv4_1' (d) and 'conv5_1'(e). The style of the input image is reconstructed from style representations built on different subsets of CNN layers: 'conv1_1' (a), 'conv1_1' and 'conv2_1' (b), 'conv1_1', 'conv2_1' and 'conv3_1' (c), 'conv1_1', 'conv2_1', 'conv3_1' and 'conv4_1' (d), 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1' (e).

In our CNN model, we do content and style reconstructions. For content reconstructions, we reconstruct the input image from layer 'conv5_2' of the pretrained VGG-model. For style reconstructions we build a new feature space that captures the style of an input image on top of the original CNN representations in order to make correlations between the different features in different layers of the CNN. We reconstruct the style of the input image from style representations using this subset of the CNN layers: 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1'. For a visualization of this description, look at Figure 1.

After this reconstruction, we optimize the model using an Adam optimizer. We also use the total variation to adjust the loss of the image at each training epoch in order to reduce the noise in the output image. We then train the image for 7 epochs, 100 steps per epoch to get the final stylized image.

b TF-Hub Model

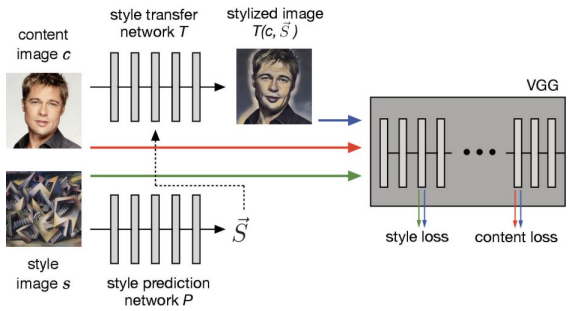

Figure 2. Diagram of the TF-Hub model architecture [3].

The underlying way the TF-Hub model works is the style prediction network P predicts an embedding vector $\vec{S}$ from an input style image and sends $\vec{S}$ to the style transfer network T. T then transforms the photograph into a stylized image based on the style image. Then, the content and style losses are calculated and are minimized during backpropagation [3]. All of this work is done for us by Google when we call TF-Hub in our code. Please see Figure 2 to see more about the structure of the TF-Hub model.

2.3. Masking

a OpenCV Based Masking Method

For masking the images, we created a binary mask based on the intensity threshold of the source image in grayscale using OpenCV, an open-source computer vision library, to efficiently process and manipulate the image data according to the established intensity threshold. This mask was generated by employing the following equation:

$$\mathrm{new\_img} = \mathrm{source\ img} * \mathrm{mask} + \mathrm{stylized\ img} * (1 - \mathrm{mask}) \quad (2)$$

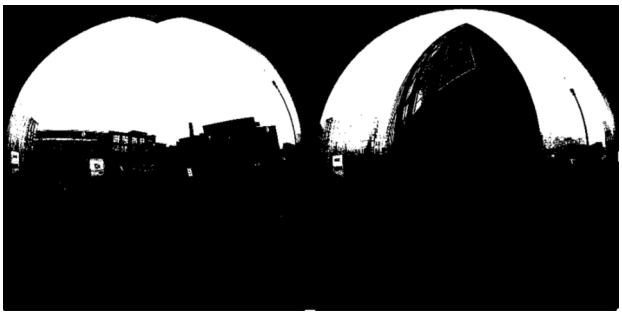

Figure 3. Mask generated using OpenCV Python Script

b Manual Masking Method

The method involving Adobe Photoshop entailed manually sculpting the mask by meticulously selecting and delineating specific regions within the image. This process allowed for detailed customization and precision, enabling the creation of the desired mask through intricate manual adjustments. However, it required substantial human intervention and proved time-intensive due to the meticulous nature of manual selection and editing within Adobe Photoshop.

Figure 4. Mask generated using Adobe Photoshop

3. Results

The VGG model exhibits outputs that closely resemble the artistic portrayal of skies found in paintings. Its ability to capture and replicate the essence of the sky within the artwork stands out prominently. On the other hand, the TF-Hub model successfully incorporates the style of the image into the sky region, skillfully blending it with the original image's aesthetics. However, the TF-Hub model encounters challenges when extending its styling capacity to the entirety of the image, resulting in less impressive outcomes beyond the sky.

The Patch-based method is good at copying the brushstrokes from the original style image, creating a similar artistic texture in the final picture. Yet a problem arises where you can notice the edges of the patches when looking at the entire image. This makes it hard for the patches to blend well together, affecting how the artwork looks as a whole. Figure 5 shows the sample input pictures captured

Figure 5. Sample Input Insta360 Images

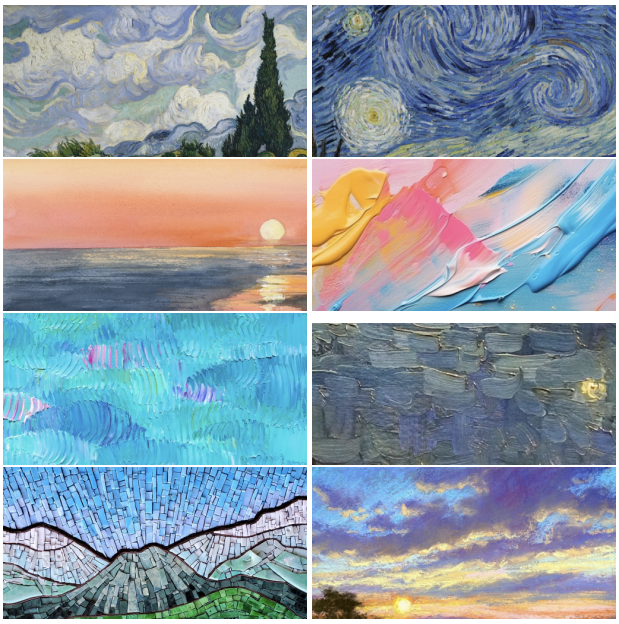

Figure 6. Sample Input Texture Images

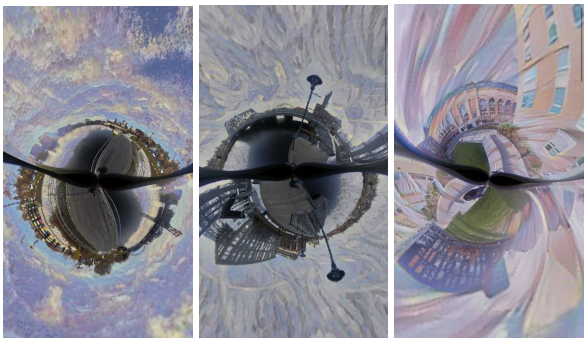

Figure 7. Tiny Planet Effect Stylized Outputs

using the Insta360 Photos. As the photos taken with this

specific camera equipment are in the .insp format, we converted them to .jpg format to apply our code. These images are in the .jpg flattened format.

Figure 6 illustrates the various stylistic patches that we experimented with for our painterly rendering tasks.

Figure 7 displays the results of our stylized painterly rendering code applied to our .insp photos. To enhance the visual appeal, we utilized these photos to generate a tiny planet effect on the output panorama, contributing to the creation of these stunning outputs.

Figure 8 represents the output of the texture based patching method on our mask-added input images.

Figure 9 represents the flattened .jpg outputs of the two style transfer methods mentioned in Efros paper [1].

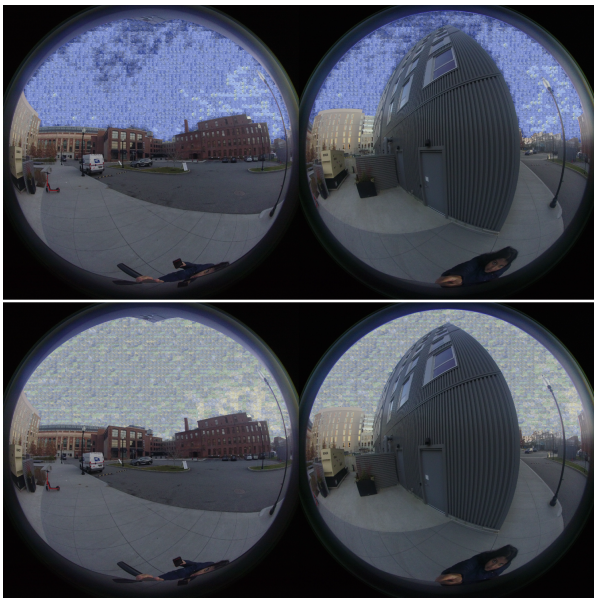

Figure 8. Texture Based Patch Outputs

4. Challenges

1. One challenge we faced was making the 360º images look seamless in the immersive videos. We could not find software that could turn the 2-dimensional images into 3-dimensional interactive images seamlessly. The Insta360 One app™ had the best results, but there are still black spots and seams in the interactive images.
2. Additionally, we found that working with entire images was challenging. Style transfers gave poor results on foreground objects and texture transfer was time consuming. As a result, we attempted to mask out areas of interest.
3. One of the other issues we had was developing masks by code. We did better when making them manually, but the generated masks still look similar.
4. Another challenge was applying style transfer with little noise in the VGG model. We were able to reduce this by factoring in the total variation of the image when calculating the loss in the neural network.

5. Contributions and Extra Credit

5.1. Contributions

a All Team Members:

We collectively sourced various styles and textures for all the methods, and each of us executed the code files. Furthermore, we collaborated on creating content for both the presentation slides and this report.

b Allen Dufort:

I developed the neural network and masking code in `style_transfer_final.ipynb`, including the CNN and TF-Hub models. Additionally, I conducted research on Gatys' paper and Ghiasi's paper [2, 3]. I contributed to the documentation in the README.md file and played a part in crafting sections of this final report.

c Gabby Litterio:

I developed the texture transfer code in main.py and student.py, experimenting with various tile and overlap sizes to optimize performance. Additionally, I conducted research on Efros' paper titled "Image Quilting for Texture Synthesis and Transfer" [1]. For comparison purposes, I manually crafted a mask using Adobe Photoshop as part of the experimentation process. I also presented our presentation to the class.

d Muskaan Patel:

I utilized the Insta360 One Camera to capture 360º photos and applied outcomes from both the `style_transfer_final.ipynb` notebook and `main.py` to generate interactive 360º stylized photo videos within the Insta360 One app™. Throughout this process, I conducted extensive experimentation involving over 20 distinct styles and more than 30 different 360 images. This comprehensive approach allowed me to implement and evaluate both style transfer and texture transfer methods, aiming to select the most optimal results. Moreover, I conducted essential pre-processing on the Insta360 images to enhance the quality of the final outputs. Lastly, I also created the videos of the interactive 360º stylized photos on the Insta360 One app™.

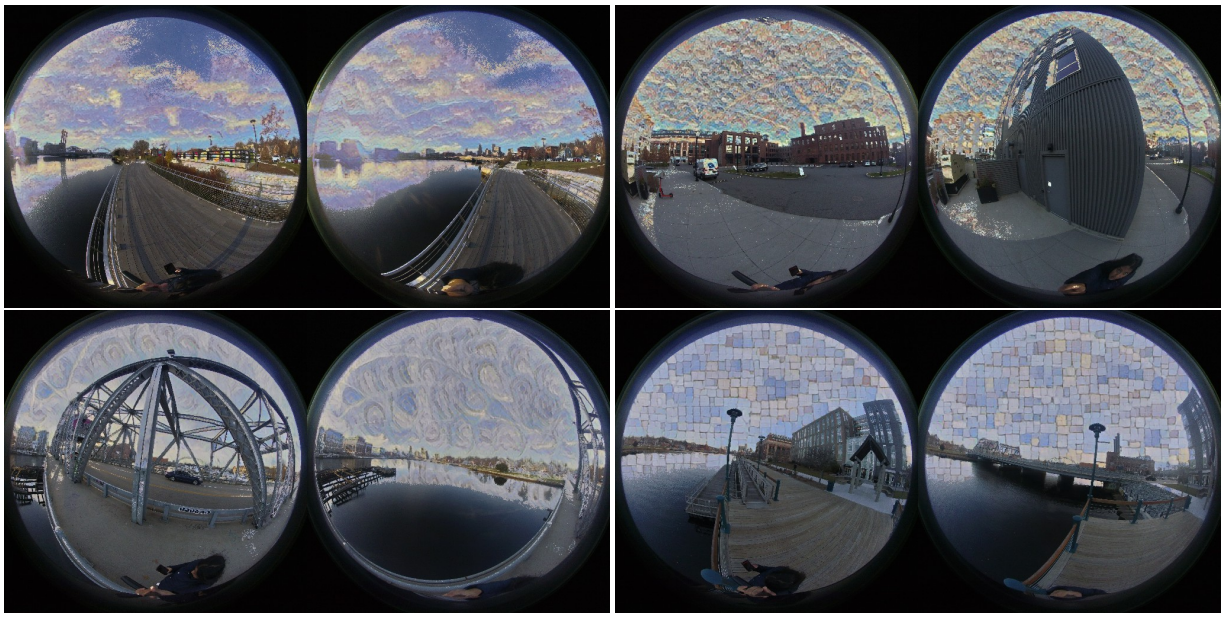

Figure 9. Insta360 Style Transferred Outputs

5.2. Commensurate Extra Credit

In this project, we made commensurately extra efforts that surpassed the standard expectations of the course. The project comprises three distinct style transfer implementations, showcasing a patch-based algorithm covered in class, a convolutional neural network (CNN) style transfer that goes beyond the course content, and the integration of TensorFlow Hub with a pretrained CNN. This represents a commensurately extra effort as it involved self-directed learning, model training, and applying advanced deep learning concepts not explicitly covered in our coursework. Additionally, we transformed the style-transferred images into interactive 360-degree visuals on the Insta360 One app™, integrating our classwork with a real-world application. Overall, the project's extra credit is the diversity of techniques, the integration of machine learning principles and real-world applications.

References

[1] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. https://api.semanticscholar.org/CorpusID:9334387, 2001.

[2] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. http://arxiv.org/abs/1508.06576, 2015.

[3] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. Exploring the structure of a real-time, arbitrary neural artistic stylization network. http://arxiv.org/abs/1705.06830, 2017.