Immersive 360° Style Transfer

Cheong Zhi Xi Desmond
Brown University
Providence, RI 02912
desmond_cheong@brown.edu

Miku Suga
Brown University
Providence, RI 02912
miku_suga@brown.edu

Jian Cong Loh
Brown University
Providence, RI 02912
jian_cong_loh@brown.edu

Abstract

In this project, we first used image representations derived from Convolutional Neural Networks to combine the semantic content of an image with the style of another image, based on the technique developed by Gatys et al. Next, we scaled the style transfer technique up from 512 x 512 pixels in the original paper and applied it to 6000 x 3000 pixels to create immersive 360° panorama images. Due to GPU memory limitations, the maximum image size that we were able to run the algorithm on was 1500 x 750 pixels. We thereby ran style transfer on smaller, divided portions of the content image before performing post-processing to combine them seamlessly into a full-sized image.

1. Introduction

In *A Neural Algorithm of Artistic Style*, Gatys et al. introduced a method for image style transfer that uses the feature representations learned by Convolutional Neural Networks to extract the content and style information from two images to combine them into a single image [1]. They worked with the 19-layer VGG network pre-trained on Imagenet, and identified 'conv4_2' as the layer that captures the high-level content of an image, and layers 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1' as the ones having good style representations.

At a high-level, the algorithm performs gradient descent to minimise the loss between the feature representations of a noise image with the content representation of the original content image in the content layer, and the style representation of the original style image in the style layers.

2. Methodology

2.1. Style Transfer

We first implemented neural style transfer on small 128 x 128 pixels images using the method described above (Figure 1). Once it was successful, we attempted to scale the image up to the maximum size permitted by the GPU. On a NVIDIA Tesla K80 GPU, we were able to perform style transfer on images of up to 1500 x 750 pixels before reaching the memory limit of the machine. In order to generate images of 6000 x 3000 pixels, which is the size needed for images to look natural when visualised as 360° panorama images and to preserve high frequency visual information, we decided to divide the original content image into smaller images of size 1500 x 750 pixels each, run style transfer on them individually, then combining them into the full-sized image.

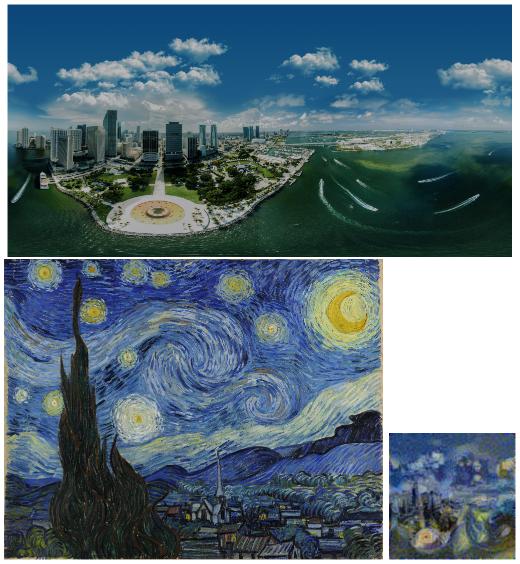


Figure 1. *Top:* Original content image, *Bottom left:* Original style image, *Bottom right:* Result of style transfer

According to Gatys et al., initialising the image synthesis process with the content image or the style image instead of noise would "bias the final image somewhat towards the spatial structure of the initialisation" and "deterministically leads to the same outcome". Given that a key challenge we anticipated was in ensuring that the semantic content of each smaller image aligns well with its neighbouring images' such that the seams are less obvious, we initialised

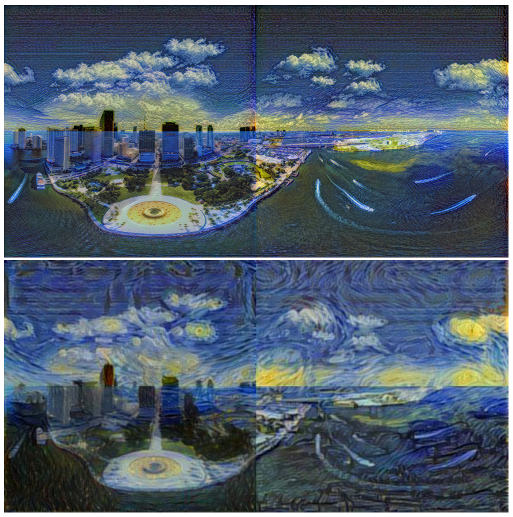


Figure 2. *Top:* Image produced from four 1500 x 750 pixels images, with style image resized to 1500 x 750 pixels *Bottom:* Image produced from four 300 x 150 pixel images, with style image resized to 600 x 300 pixels and divided into four. In the bottom image, we see that the clouds have a deeper yellow hue, which we hypothesise is a result of the style image having a moon in the top right quadrant. Because of the localisation of style features due to the spatial layout of the style image, the colours in the second image were less balanced across the smaller images.

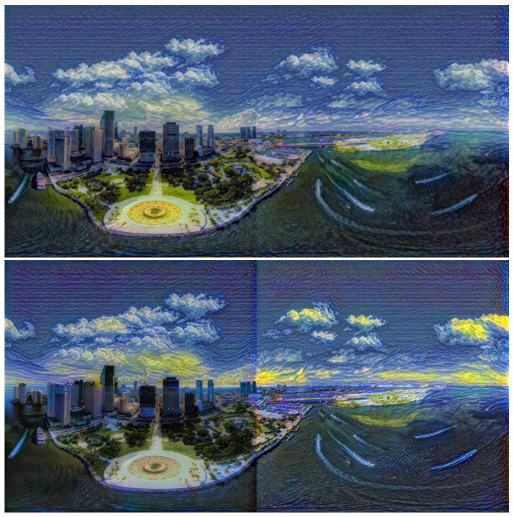


Figure 3. *Top:* 1500 x 750 pixels image produced from style transfer, which was resized to 3000 x 1500 pixels and divided into four, *Bottom:* 3000 x 1500 pixels image produced from using the top image in initialising style transfer on four 1500 x 750 pixels images.

the synthesis process using the content image to preserve more of the content information.

2.2. Generating Larger Images

When running style transfer on the smaller (1500 x 750 pixels) images, we had to choose between scaling the style image to the size of each small image, or resizing it to final output size (3000 x 1500 pixels for an image composed of 4 smaller images) then splitting it into smaller parts. The latter has the advantage of allowing us keep the relative scale of the content and style images invariant to the output resolution. However, splitting the style image into smaller images resulted in the localisation of style features within specific parts of the image, and this localisation is arbitrary depending on the distribution of features within the style image (Figure 2).

To reduce the seams between the smaller images and make the large image more coherent, we considered modifying the style transfer algorithm to minimise the content and style loss between neighbouring small images. One such modification that we made was in resizing the 1500 x 750 pixels image generated through style transfer to 3000 x 1500 pixels, then dividing it into smaller images to be used in the initialisation of the synthesis of smaller images. While this produced some stabilisation in style, it did not give significant improvements in overall coherence of the large image (Figure 3). Given that modifying the neural style transfer algorithm takes a long time to experiment and tweak and does not necessarily give significant gains, we decided that post-processing would give better results.

2.3. Post-processing

By dividing a large image into smaller patches then performing style transfer on each individual patch, we are able to scale the style transfer technique indefinitely even with limited memory constraints. However, because each patch underwent style transfer separately, after stitching the patches back into one larger image, the final output does not look coherent. Figure 4 shows an example of this issue. To improve the coherence of the combined image, we carry out a series of image processing procedures.

2.3.1 Feather Blending

In this project, feathering blending refers to a method of blending pixel values from another image onto a target image starting from the edges of the target image. The effect of the blending is applied more heavily along the edges, and quickly tapers off as the distance from the edge increases.

Let us call the width of pixels altered by the feather blend the 'blend width' w, and the magnitude of blending for a pixel that is x pixels away from the blending edge the 'blend factor' B_x. We define B_x such that it decays rapidly the further it is from the seam edge:

$$B_x = -\frac{1}{2} log \frac{x}{w}$$

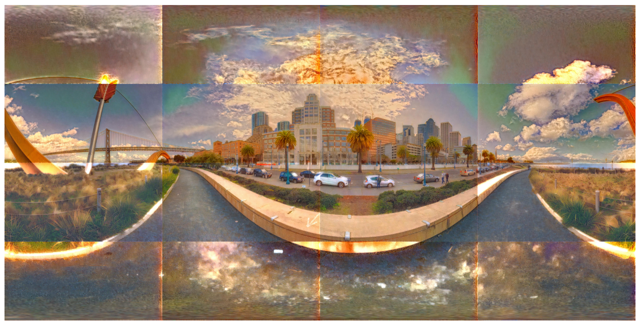


Figure 4. The unprocessed output from stitching together 4x4 images of size 1500x750 pixels each. Although the content elements are present, and each individual patch has been stylized, they obviously look like 16 different images being put together.

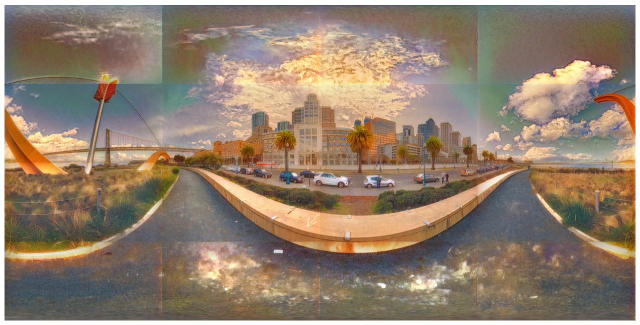


Figure 5. The effects of feather blending on the unprocessed output from Figure 4. The blend width used in this example was 200 pixels.

Using this formula, let the original pixel's value be I_0, and the value of the pixel we're blending it with be I_1, the final blended pixel will have a value I'_0 of:

$$I'_0 = \frac{I_0 + B_x I_1}{1 + B_x}$$

Feather blending is applied to the edges of each image patch by sampling pixel values from the corresponding points on a smaller stylised image. Figure 5 shows that this method is effective at softening the appearance of seams.

Applying more blending passes for larger blend widths could greatly improve the overall coherence of the image.

However, although this technique is effective at minimising the appearance of seams along the edges of stitches, it is best to avoid applying it too heavily or to use a blend width that is too wide. This is because we sample our blending values from a lower-resolution base image, so blended pixels contain a mix of high-frequency and low-frequency information. Increasing the amount of blending decreases the amount of high-frequency information, which is counterproductive for scaling up the resolution of style transfer images.

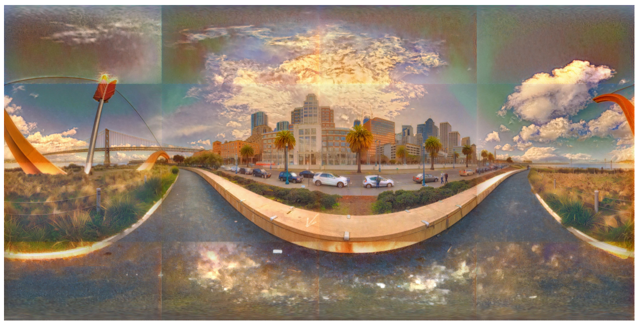


Figure 6. The effects of normalising each 1500x750 pixel patch on the unprocessed output from Figure 4 with a lower-resolution stylised image.

2.3.2 Normalisation

From a visual inspection of the unprocessed image, we see that there are sudden and large-scale changes in the colouring from patch to patch. This contributes significantly to the incoherence of the overall image.

To even out the larger-scale colour values between each patch, we adapt our previous strategy with feather blending: sample colour values from the corresponding points on a smaller stylised image, then use these values to normalise the colour values in our large image.

To illustrate, suppose we normalise the colours of an entire 1500x750 pixel patch P_0. Let the corresponding 1500x750 pixel patch from the smaller stylised image be P_1. We compute the new 1500x750 pixel patch P'_0 as follows (where μ represents a mean function and σ represents a standard deviation function):

$$P'_0 = \frac{P_0 - \mu(P_0)}{\sigma(P_0)} \times \sigma(P_1) + \mu(P_1)$$

Apply this process to every 1500x750 pixel patch from Figure 4 gives us the output in 6. While the large-scale colours have become more even, there are abnormally bright spots in the image. These spots are areas where pixel values were stretched across a range of values that is too large, and can be mitigated by further renormalising them to span a smaller range. This can be achieved by using the same renormalising method, but with smaller and more localised image patches. However, using image patches that are too small could cause the same issue of losing high-frequency information as with blending. The complete process that we use for our results is described in the following section. We found that this process strikes a balance between improving overall image coherence, and preserving the images' high-frequency information.

2.3.3 Overall Post-processing Pipeline

Take a smaller stylized image as a base image and scale it to the same size as the large image. Then perform the following steps:

1. Normalize each patch in the large image with the values of the corresponding patch in the base image.

2. Normalize every horizontal and vertical band in each patch in the large image with the values of the corresponding band in the base image.

3. Feather blend the edges of each patch of the large image with the pixel values in the base image.

3. Results

The full results of our style transfer on 7 images of 6000 x 3000 pixels, composed of 4 by 4, 1500 x 750 pixels images can be found in Appendix Figure 7. For each image, the center of the image is zoomed in on a 360° panorama visualiser, showing the corners of 4 individual patches stitched together post-processed. We can closely observe the effects of post-processing on each image.

We have also created a demonstration at https://miku-suga.github.io/cs1430-final-project/visualizer.html to visualise the images as 360° panoramas.

3.1. Scale of stylised features

There is a large variation in the size of content elements in our images. Image **H** shows an example with small baseball players in the center, but human faces on the edges that are many times larger. Because of this, 'semantic' objects are stylised with varying success. In this example, the baseball players are well-stylised, but the large facial elements seem to be simply coloured as if they were a large background. This area requires further exploration, possibly by adapting the use of the pretrained VGG network.

3.2. Post-processed image variations

There are slight differences in the quality of our final outputs. For most images the post-process worked well and the seams of the sewn patches are mostly unnoticeable, for example, image **D**. However, for example, in image **B**, if one were aware that the image had been initially divided up into patches, it is not difficult to locate the seams. We believe this is due to pixel intensity variations along the edges of the seam - the lower the variation, more pixels identified as low frequency, allowing normalisation in those low-frequency information, generating better post-process results.

To address the issues with some obvious seams, we tried a few methods. One attempt was to renormalise pixels around an edge with colour values on the other side of the edge from the neighbouring patch. This would be combined with a smoothing method similar to feather blending. However this attempt caused a mirror-like swap in colours along the edge, arguably making the seams more obvious.

Additionally, we tried to apply the normalisation method to even smaller divisions of the large image (for example, splitting the large image into 120x120 rectangles, then normalising each rectangle using a base stylised image). However this gave the image a grid-like appearance because every normalised block resulted in new seams along its edges. From these attempts, we believe there is still a lot of room for further experimentation and improvement of the techniques explored in this project.

4. Conclusion

In this project we scaled the style transfer technique for much larger images in order to generate immersive panoramas. To deal with memory limitations, we felt that simply switching to GPUs with larger capacities would not adequately address this problem, but merely push the problem further down the road until the increased memory limit is hit once again. As such, we explored a method that would allow one to apply style transfer to indefinitely large images even with a memory constraint. The results can be improved on, but we believe that this combination of deep learning methods and traditional image processing has potential for making style transfer more computationally viable.

4.1. Future Directions

That being said, taking advantage of more powerful GPUs can also open up more possibilities. Since we hit the memory limit on the NVIDIA Tesla K90 GPU for relatively small images, we could look at using other recommended GPUs. E.g. NVIDIA Tesla P100, NVIDIA Tesla T4.

4.1.1 Real time Video Immersive 360° Style Transfer

Another extension would be to perform style transfer repeatedly in a loop on still images from successive frames taken from a live video. It has been 5 years since the original paper on style transfer by Gatys et. al. where they discussed the performance barriers for online and interactive style transfer applications. There have been improvements in technology since then, but there are still challenges for such applications. For real-time video style transfer, we foresee a challenge in minimising latency and utilising GPUs that maximise frame rate of the output video to perform smooth style transfer on live videos.

Possibly, a solution would involve not just improvements in deep learning and GPU technology, but also a mix of other techniques in computer vision.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015.

A. Contributions

Cheong Zhi Xi Desmond worked on setting up the evaluator class to calculate the losses in each iteration, the optimization function, and did general housekeeping and refactoring of the code. Executed post-processing on the images, including both feather blending and normalisation. Set up the html 360° visualiser.

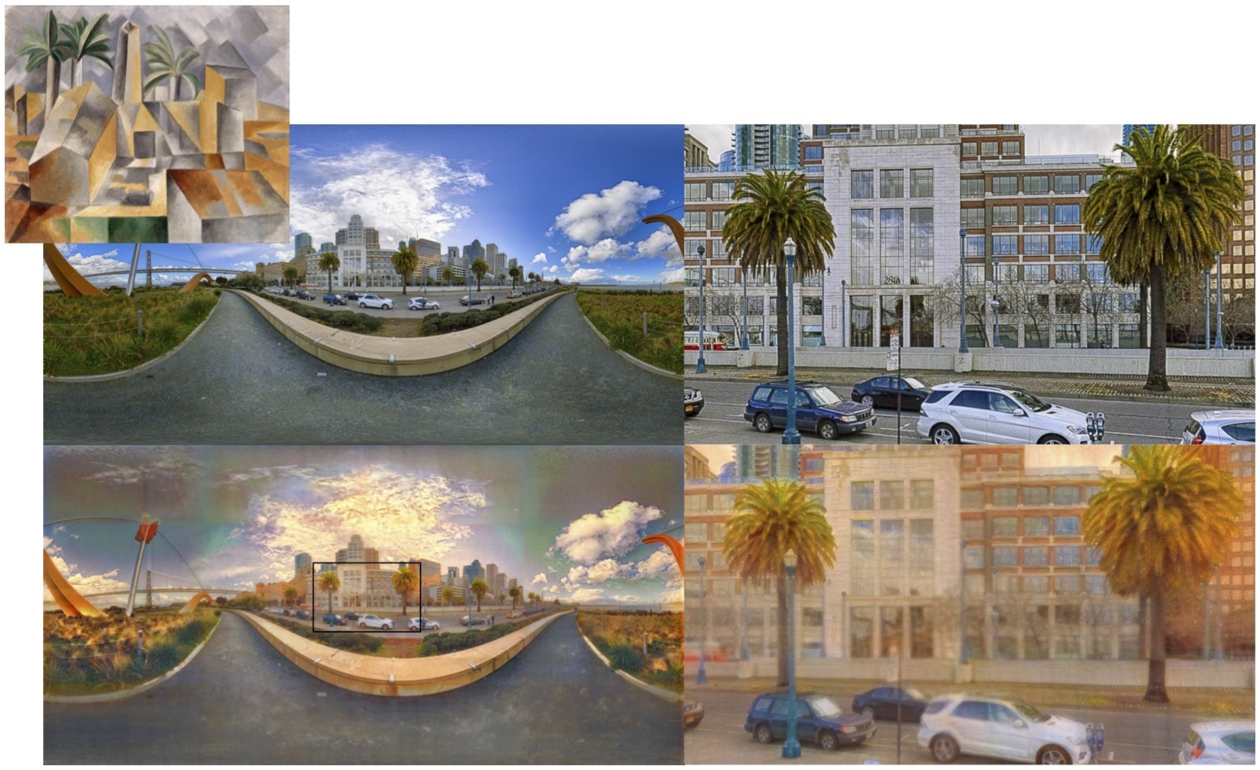
Miku Suga worked on defining the content loss, style loss and total variation loss functions and combining them together into a tensor. Then experimented with preserving high frequency visual information while controlling the images to look natural, helping with tuning hyperparameters. Performed style transfer on own images in **H**.

Jian Cong Loh worked on collecting the images needed and setting up the program, including image preprocessing and setting up the VGG model. He then proceeded to experiment on preserving high frequency visual information while controlling the images to look natural, investigating on resizing and dividing the images into patches. Came up with various post-processing ideas to work on and debugged many of our errors.

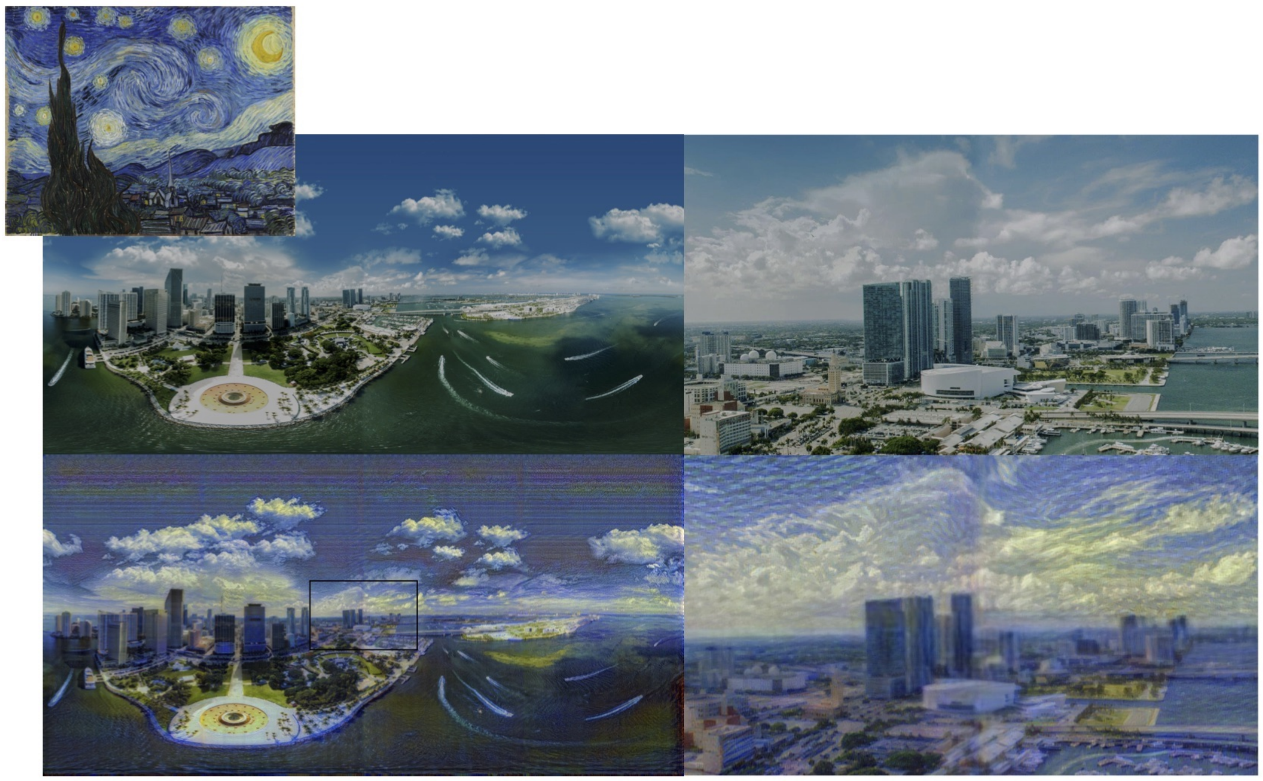
B. Final Outputs

Outputs begin on the following page.

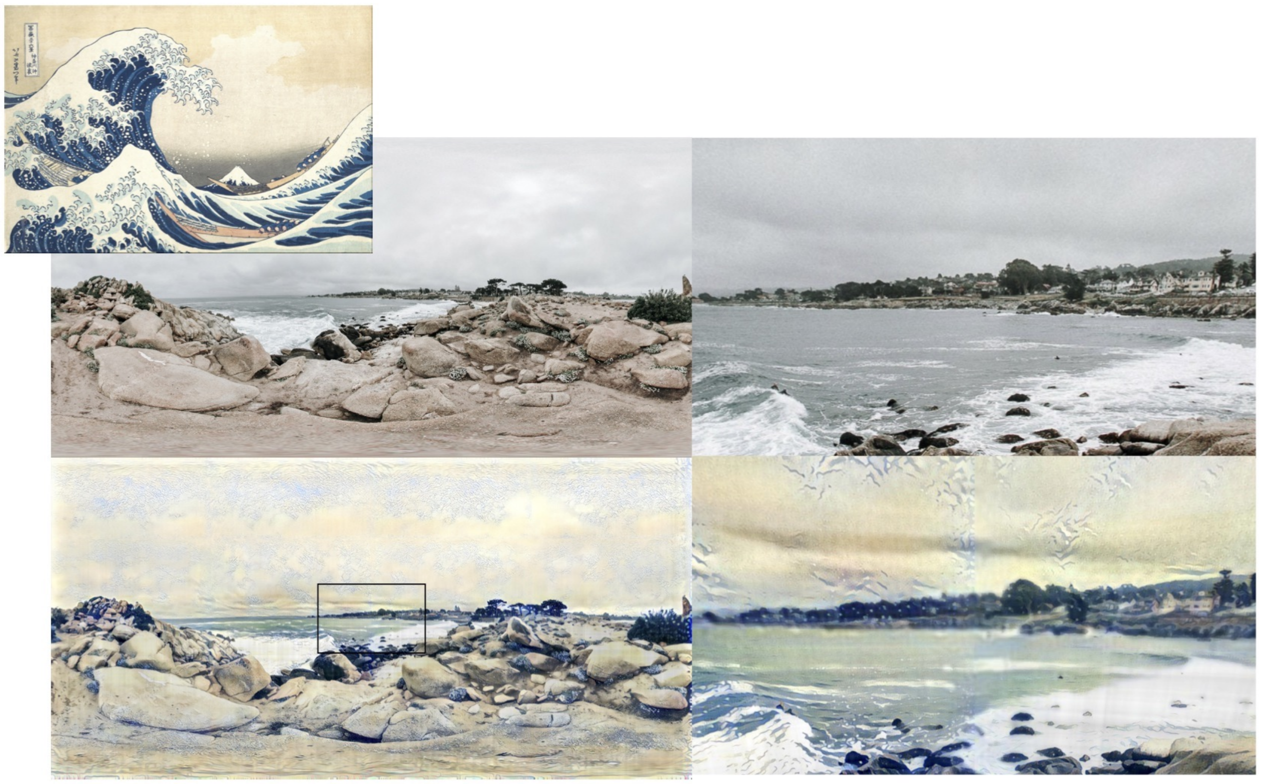
A.



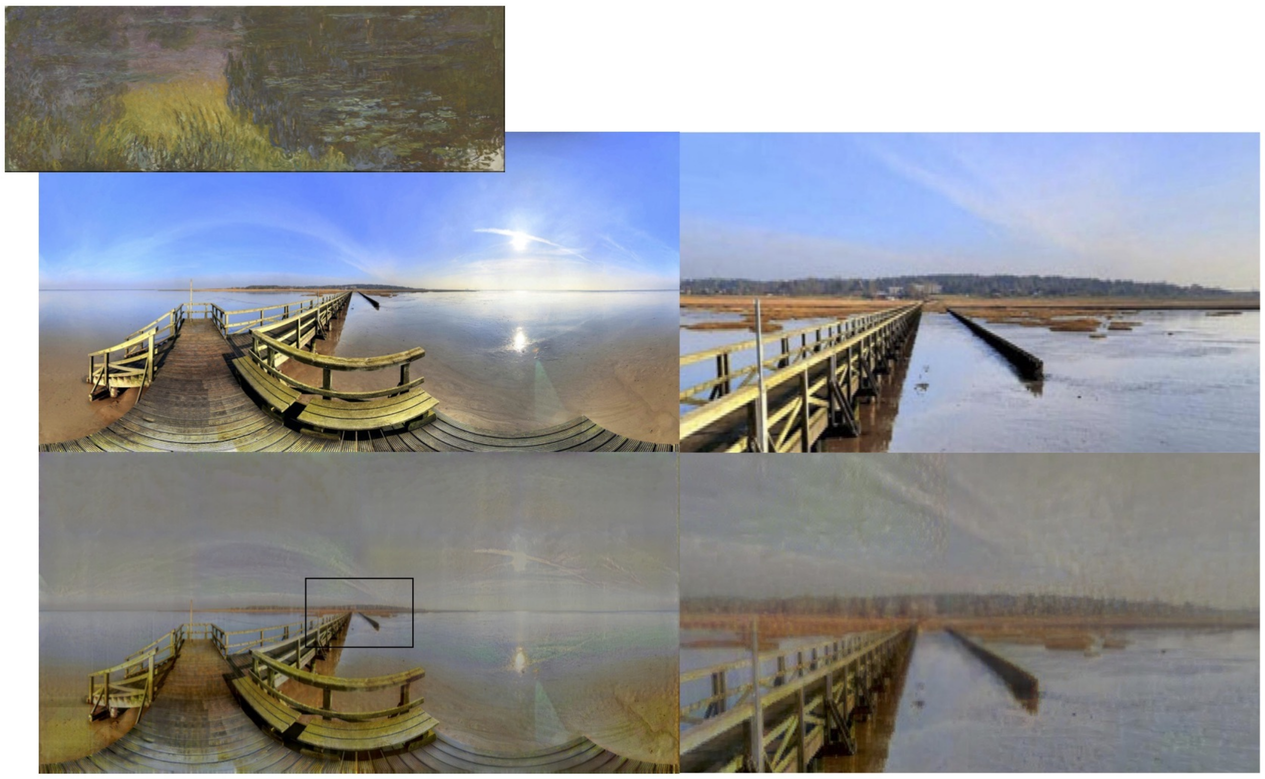
B.



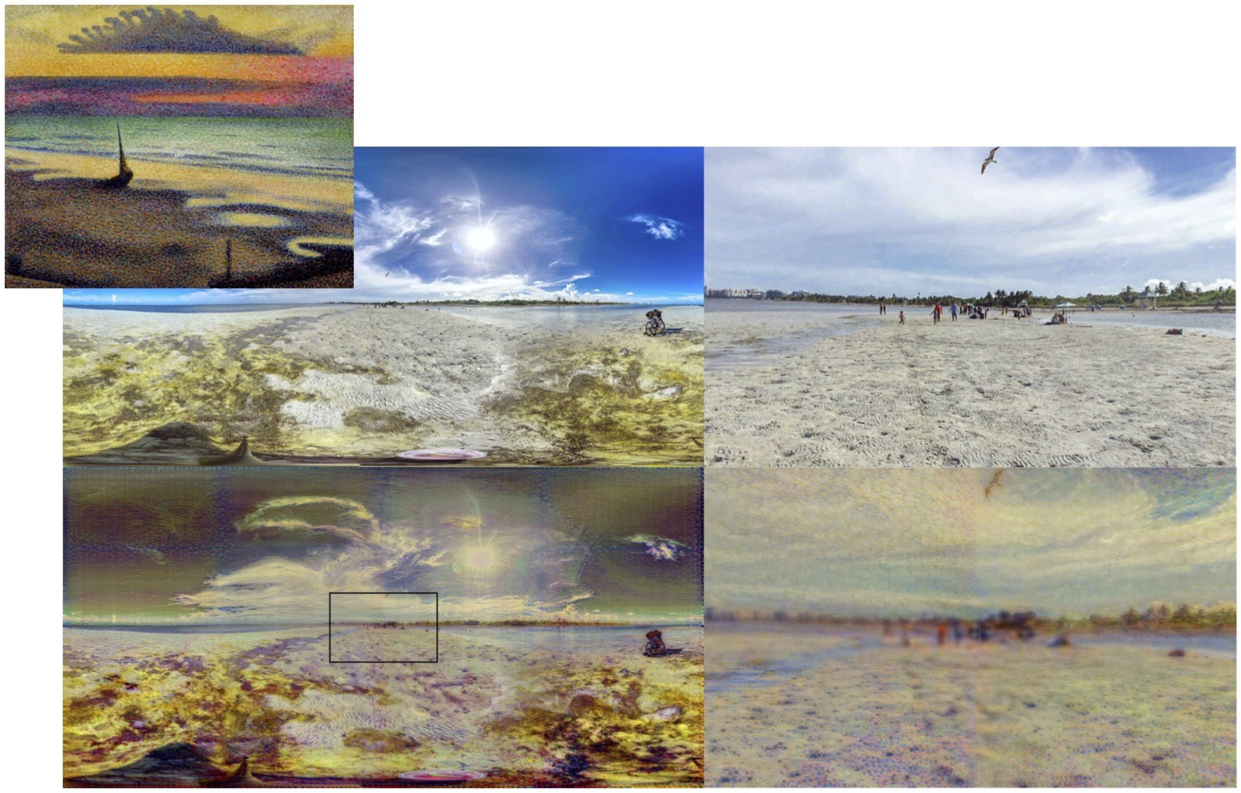
C.



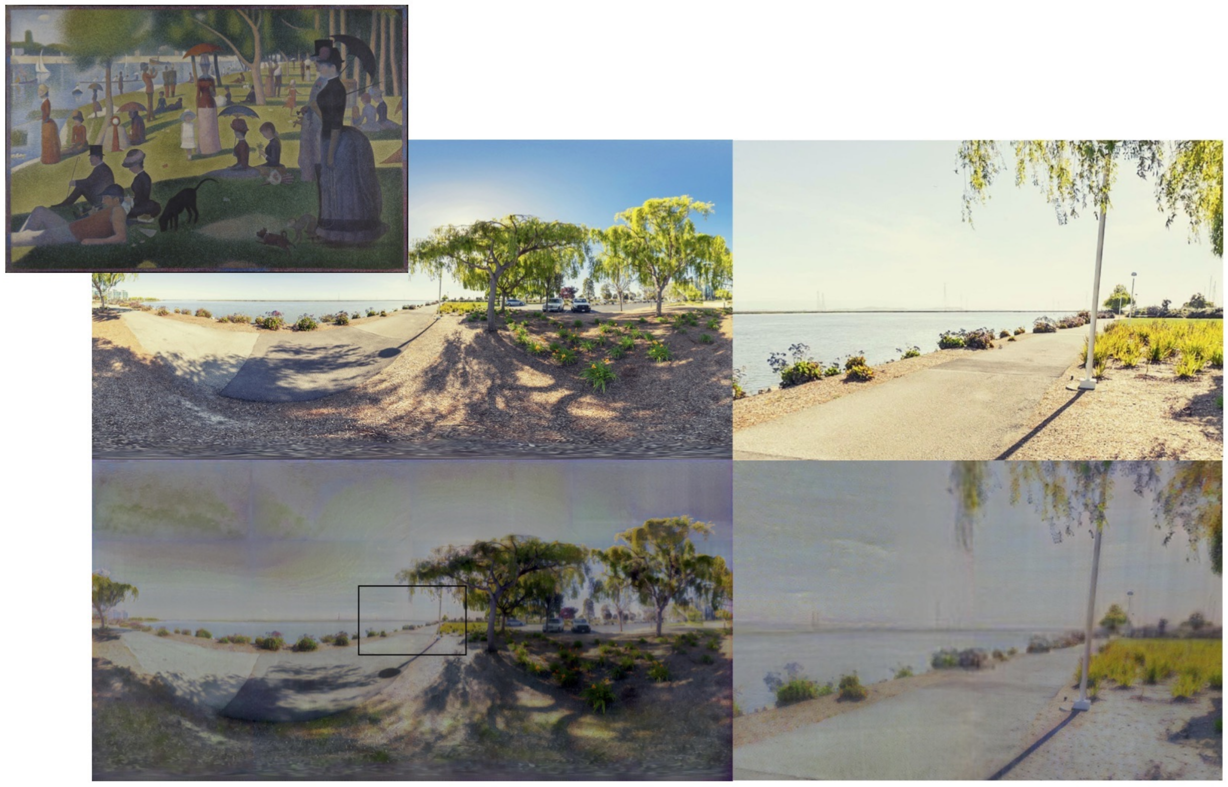
D.



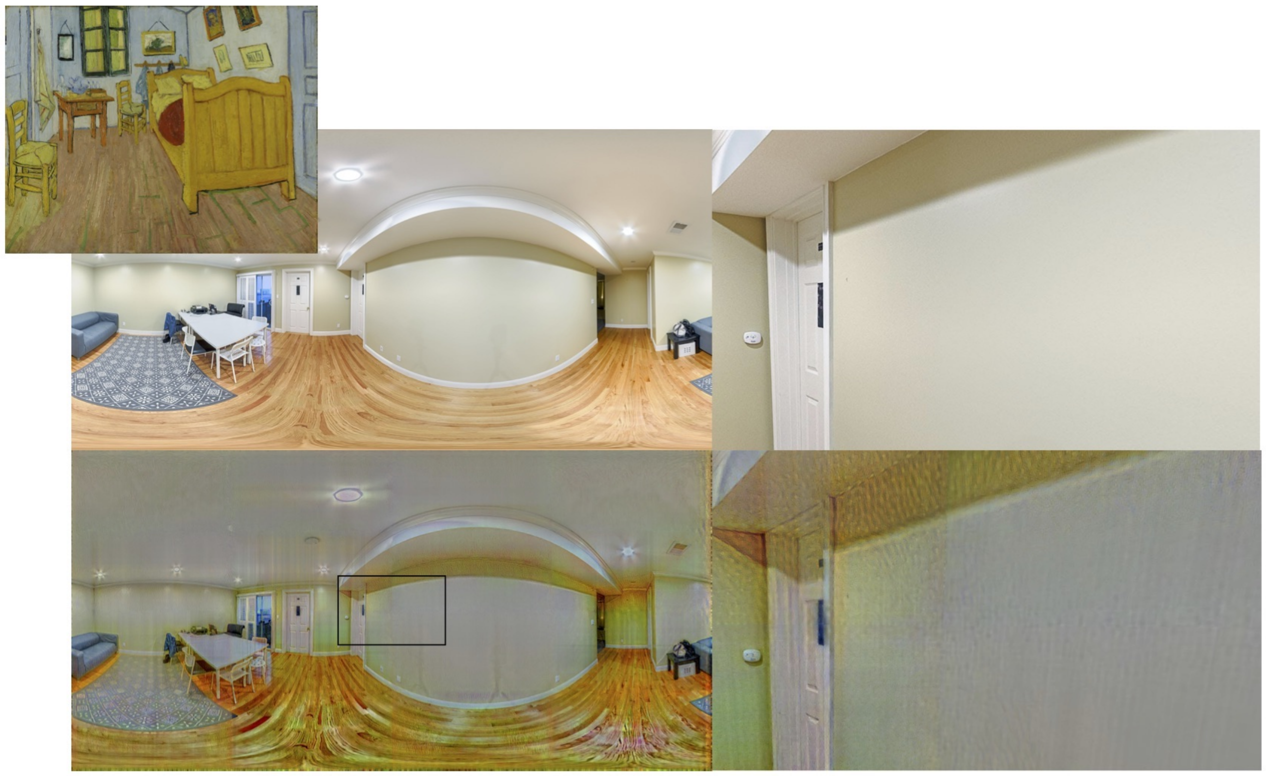
E.



F.



G.



H.

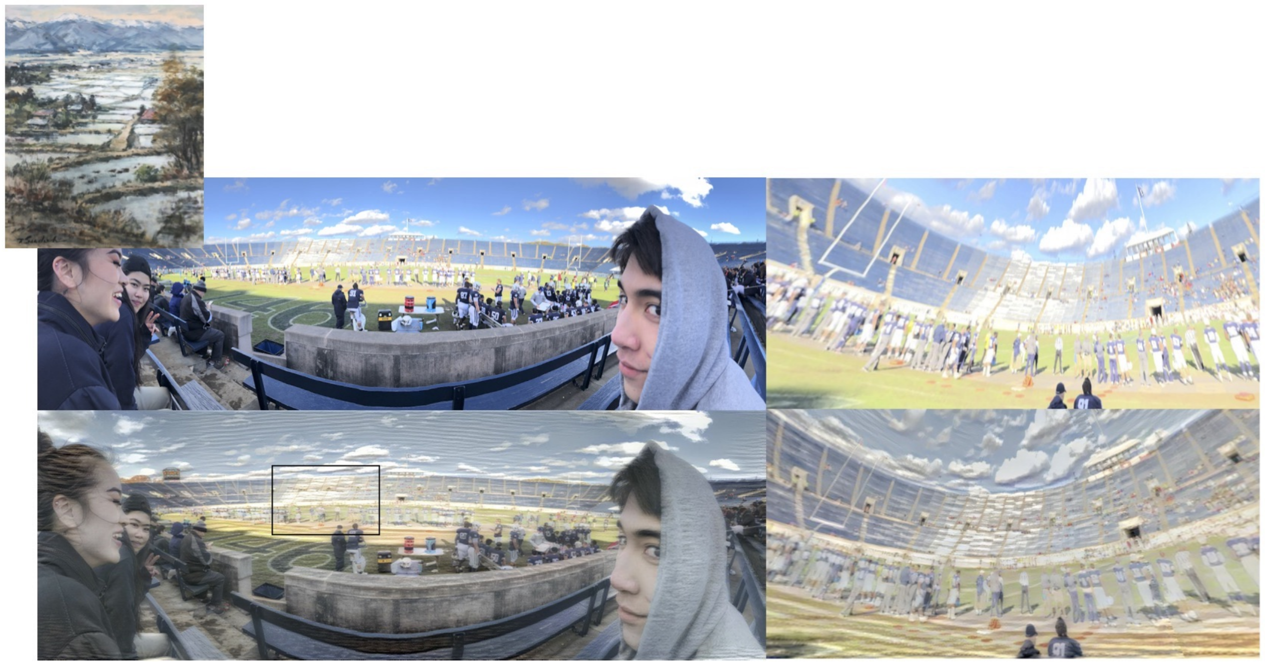


Figure 7. 360° panorama images **A-G** from: Pixexid "360 Panorama Images" pixexid.com/search/360%20panorama. Panorama image **H** taken using an iPhone at one of Brown's football game. Style art: **A.** "Brick Factory at Tortosa" by Pablo Picasso, oil on canvas, 1909 **B.** "The Starry Night" by Vincent van Gogh, oil on canvas, 1889 **C.** "The Great Wave off Kanagawa" by Katsushika Hokusai, color woodblock, 1829-1833 **D.** "The Water Lilies – Setting Sun" by Claude Monet, oil painting, 1914-26 **E.** "Plage a Heist (The Beach at Heist)" by Georges Lemmen, oil on panelmedium, 1891–92 **F.** "A Sunday Afternoon on the Island of La Grande Jatte" by Georges Seurat, oil paint, 1884–1886 **G.** "Bedroom in Arles," by Vincent van Gogh, oil paint, 1888 **H.** "Taue wo matsu suidenn" by T. Sudoh, oil on canvas, 2002