Georgia Tech's Computational Photography Portfolio

Ronak Patel
rpatel88@gatech.edu

Assignment #1: A Photograph is a Photograph

The purpose of this assignment was to explain aspects of a photograph that you were happy or unhappy with and explore what computational photography methods you would could implement on the photograph. The picture I chose Sydney Opera House photo on the right.

Sydney Opera House
Sydney, Australia
Taken with: OnePlus 5t

Assignment #2: Epsilon Photography

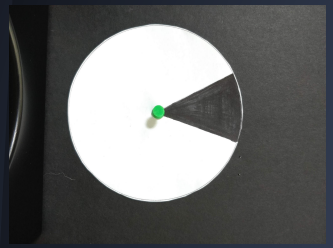


Image 1

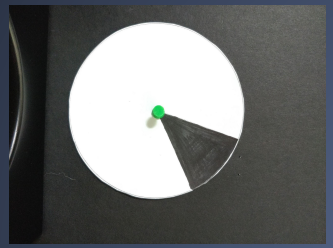


Image 2

Image 3

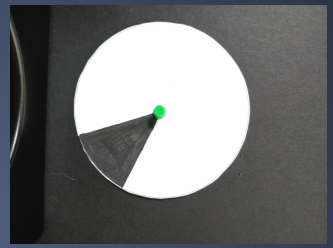


Image 4

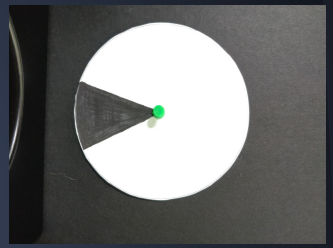
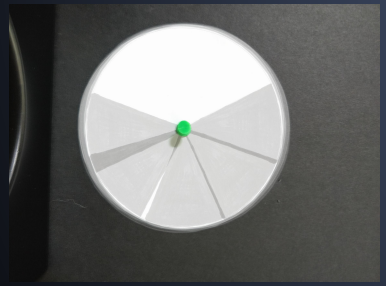


Image 5

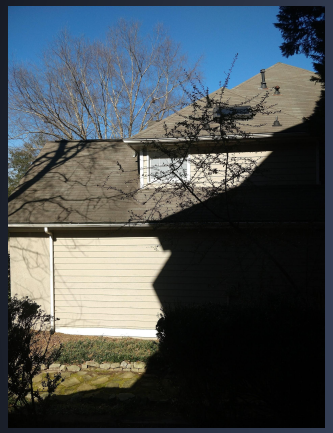


Final Artifact

The purpose of this assignment was to use epsilon photography to create a novel artifact. Here an alpha blend was created using 5 different images. The artifact shows overlapping regions and gaps throughout the rotation of the disk.

Resolution, exposure time, focal length, aperture, and ISO setting remained constant through each image. The only parameter (epsilon) that changed was the rotation of the disk.

Assignment #3: Camera Obscura



The Scene

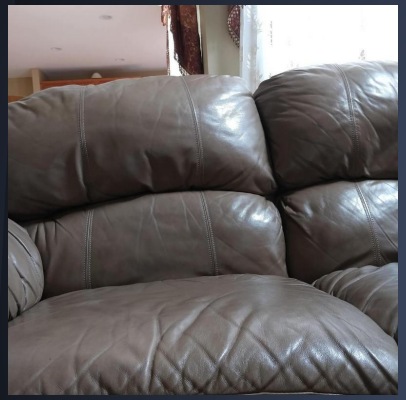
The Set-up

Obscura Image

Aperture: f/1.7
Shutter Speed: 30 sec
ISO: 100

The purpose of this assignment was to create a camera obscura and capture its results. Here, I set up a camera in the basement by creating a pinhole using foil to block out light. The image was projected onto a white sheet and was captured using a shutter speed of 30 seconds.
I was happy with the results here as we can clearly see that it is an inversion of the scene being captured. There was some blurring and the image intensities were lower for the obscura image, but that is to be expected for an in-home set up.

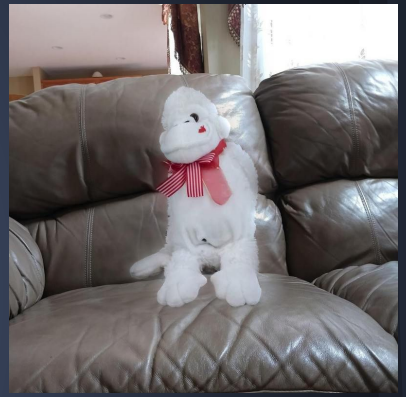
Assignment #4: Blending



Black

Mask

White



Final Artifact

The purpose of this assignment was to produce pyramid blending code and construct a laplacian pyramid blend using two input images and a mask. The mask will only allow portions of the “white” image to be blended into the black image. Here, we can see that the monkey from the white image was masked and blended with the couch in the black image.

The final artifact here was sufficient, but had ghostly features which I was not happy with due to the blending at the edges of the masked white image and the black image.

Assignment #5: Panoramas

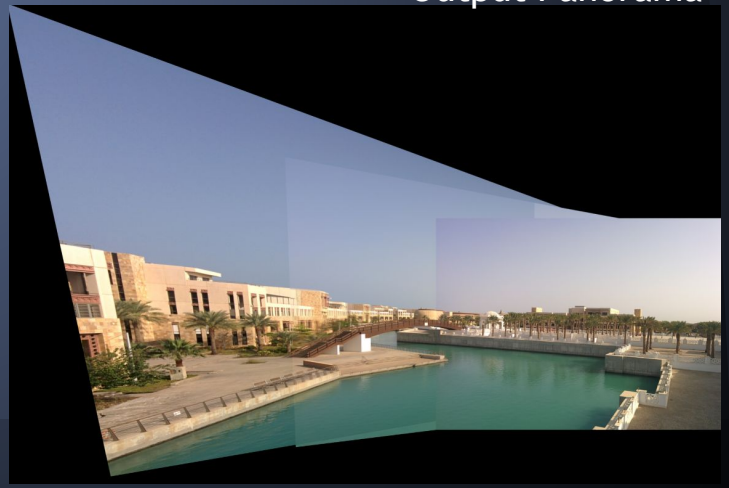
Sample Set Images:


+

+

=

Output Panorama



The purpose of this assignment was to implement panorama code and use it to construct a panorama of a sample image set and our own image set. Matching features between images were found, homography was calculated, the images were warped onto the panorama canvas, and the images were alpha blended together to output the panorama.
The results of the sample set were good when it came to finding homography and warping the images in place on the canvas. However, it can be seen that the simple alpha blend here was not sufficient enough to create a seamless panorama.

Assignment #5: Panoramas

Input images:


+

+

=

Output Panorama

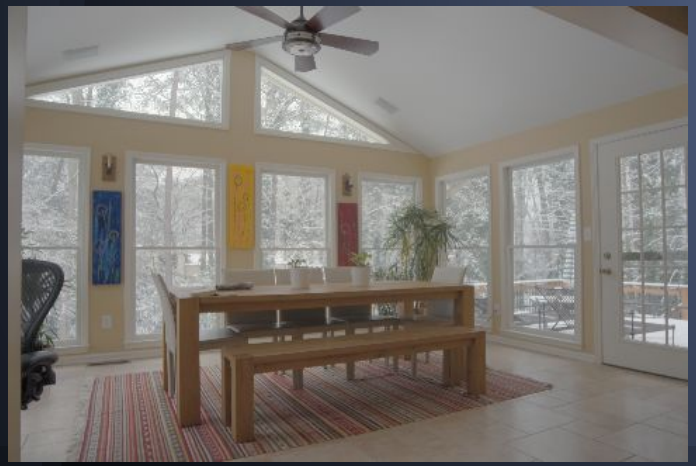
Here is the panorama created with my own input images. Overall, the boundaries between the images were blended well, however ghostly artifacts were formed due to the alpha blending technique used. To enhance both the panoramas generated in this assignment, a more sophisticated pyramid blend would have resulted in a better final output.

Assignment #6: HDR

Input Images:







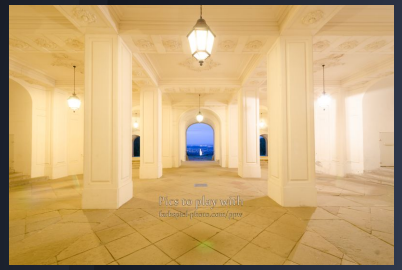
HDR Output

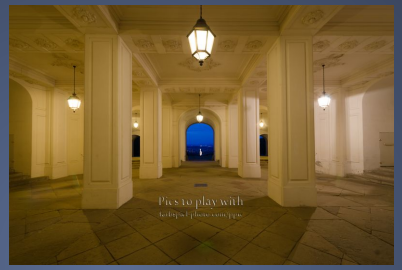
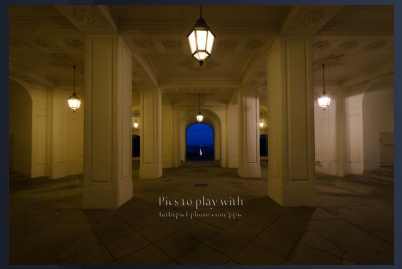
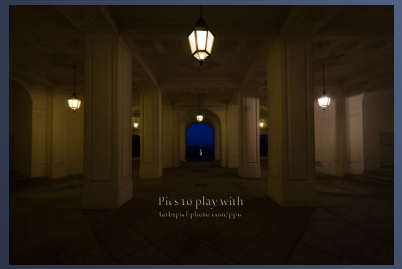
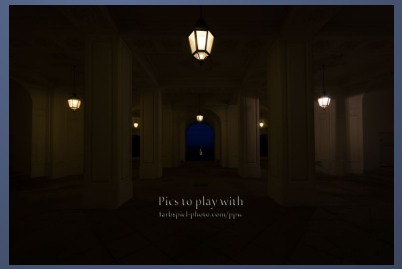
In this assignment, an HDR image was computed programmatically using sets of different images with varying exposure. The HDR image was calculated by weighting the pixels in each of the input photos, sampling pixel intensities, computing a response curve, and finally computing a radiance map to provide a normalized output image.

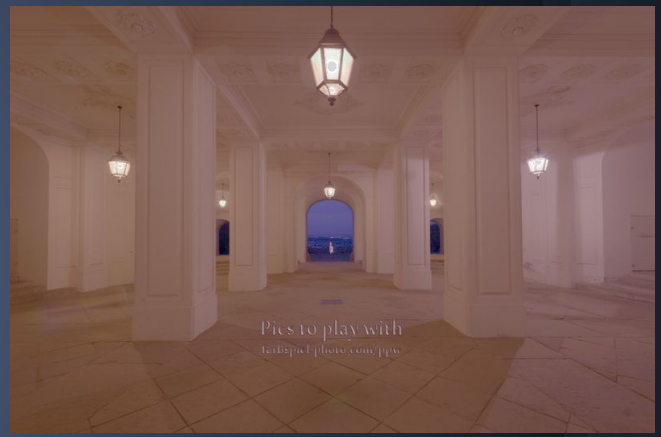
The result shown here demonstrates a successful HDR computation. We can see that the output image utilizes information from each of the input images to show a more “realistic” final output.

Assignment #6: HDR

Inputs:





HDR Output

- Exposure Times: [1s, 2s, 4s, 8s, 15s, 30s]
- ISO: 100
- Aperture: f/8

Another image set was taken with varying exposures to create an HDR image. We can again see that the computation was successful. Details from each image were preserved and the exposures values were calculated to provide a more realistic scene.

Assignment #7: Video Textures

In this assignment, video textures were explored to create a seamless loop in a video. This was done by breaking the video down into a frames and using them to calculate a transition matrices. These matrices were then used to find frames with similar transition points to create a seamless loop. The results for the two texture gifs produced can be found in the links below:

Candle Texture:
https://drive.google.com/open?id=1Hl9Mlb39zfAmQuHWCy8K3X3d17J4S304

Fan Texture:
https://drive.google.com/open?id=1O-onFgoJH6N63n-SOdzGxp2TTKPT0R1u

The results produced from my code was successful in finding transition frames. Although it is noticeable where the gif restarts, it is much better than the original video. To better this output, more frames could have been sampled from the video.

Midterm Project

The purpose of this project was to explore content aware resizing of images. A process called seam carving was implemented in order to reduce or expand image seams with the lowest energy. This project was a replication of the original method outlined by Avidan and Shamir in their paper ““Content Aware Seam Carving”.

The output of the coast image (input image) taken from the paper was replicated with the results below. My output looked identical to the source paper output, indicating successful seam carving.

Figure 1. Input Image

Figure 2. Source Paper Output

Figure 3. Replicated Output

Midterm Project

The seam expansion process was similar to the seam removal process but it was difficult to replicate the averaging mentioned in the paper. Here we can see the result with and without averaging. We can see ghostly artifacts when incorporating averaging so my results were best without averaging.

Original Image

Figures 4 & 5:
Source Paper Output

Figures 6 & 7:
Replication Output
No Averaging

Figures 8 & 9:
Replication Output
With Averaging

Midterm Project

An image of the transport map during the seam carving process of the butterfly image was also included. This map highlights in which order the seams were removed. There is a clear difference from my results and the source paper, this is most likely given by the image energy functions used to compute which seams to remove.

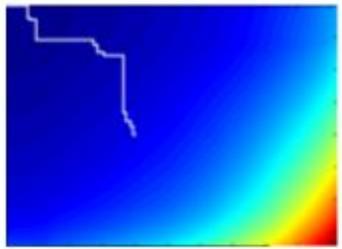


Figure 10. Source Paper T-map

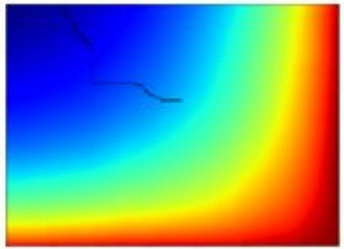


Figure 11. Replication T-map

Figure 12. Original Image (scaled)

Figure 13. Source Paper Output

Figure 13. Replication Output

Final Project: Painterly Rendering

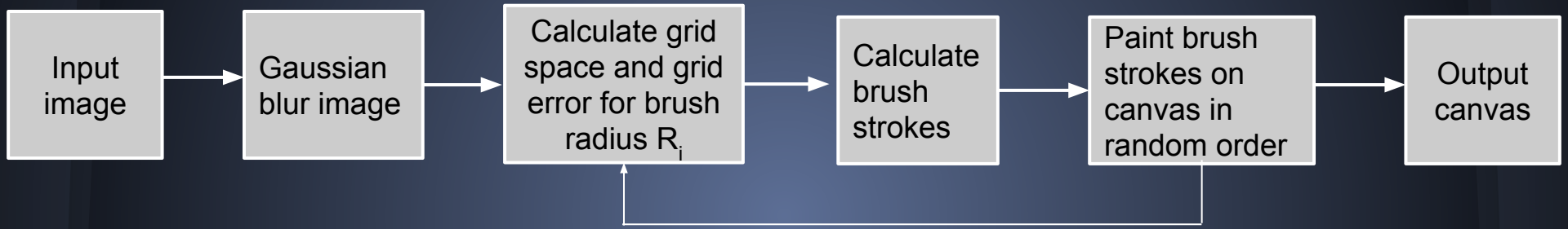
The purpose of this project is to imitate different artistic styles using programmatically calculated curved brush strokes on a canvas. Styles are created by using a series of layer and brush sizes to capture broader and fine details. The goal is to emulate impressionist, expressionist, color wash, and, pointillist styles from the paper “Painterly Rendering with Curved Brush Strokes of Multiple Sizes” by Aaron Hertzmann.

https://www.mrl.nyu.edu/publications/painterly98/hertzmann-siggraph98.pdf

Overall, I was not very happy with the final results. There were clear discontinuities in my output which were a result of unclear information from the paper or misunderstandings from the paper. If I were to revise this project, I would choose a better painting software to formulate the brush strokes and experiment more with the bounds and coordinate system used to paint the brush strokes on the canvas to eliminate white areas (shown in the last slide)

Final Project - Painterly Rendering

A pipeline for the final project can be seen below:



- An image is input into the program with initialized style parameters
- A reference image is created by using a gaussian blur with a kernel size calculated from the standard deviation of the brush stroke radius.
- A grid space is calculated as a function of the grid space parameter and the radius. Using this, a start point for the strokes is calculated by finding where the max error is within a grid space, and passed into the makeSplineStroke() method to calculate splined brush strokes
- Brush strokes are calculated using gradient directions of the reference image
- Brush strokes are painted on the canvas for the radius specified.
- Process is repeated for all other brush stroke radii and the final output canvas is outputted

Final Project - Painterly Rendering

INPUTS

OUTPUTS