Ryan Holloway
300570174

CGRA 352 - Assignment 3 - Report

All the code is written within one .cpp file. I only included that one .cpp file, since the entire project is stored in a very large folder. To get the code to run, you can copy paste all the text within the .cpp (or .txt) file into any IDE that is currently running C++ with OpenCV. I provided the light-field images as well. The only change that needs to be made to the code is on line 66, where the file path is defined for the location of the light-field images, as this will be specific to where the images are stored on the machine that is running the code. Be sure to use forward-slashes, and not backslashes between each directory.

Functions:

float get_pixel_4d()
This function returns the Vec3b pixel value within the ST image that is stored in the UV array of ST images. The row and column give access to the corresponding ST image, while the t, and s, value identify the exact pixel to return within that ST image.

float calc_distance()
This function does a simple two-dimensional distance formula calculation which is used to determine whether a pixel is within the UV image's aperture, as defined by the current aperture radius setting. The function returns the distance value, so the check for if the pixel is within the aperture radius can be computed later on in the code.

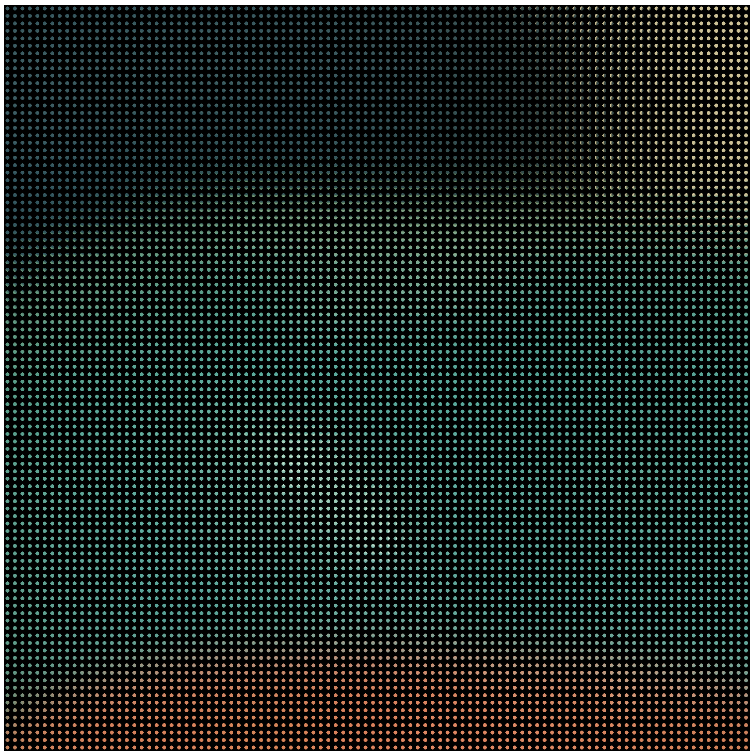
Mat micro_image()
This function iterates through every UV image and determines if its provided UV position is within the current aperture radius; and if it is, then finds the pixel located at the current ST position within that UV image and places that pixel on a corresponding 17x17 'micro image' that stores that light ray data.

Aperture Section
For the core, I created two vectors; one to store the image sensor data with the aperture radius set to 40 and the other with the aperture radius set to 75. These Mat's are then used to generate an image 1700x1700 pixels, consisting of 100x100 micro images, each 17x17 pixels in size.

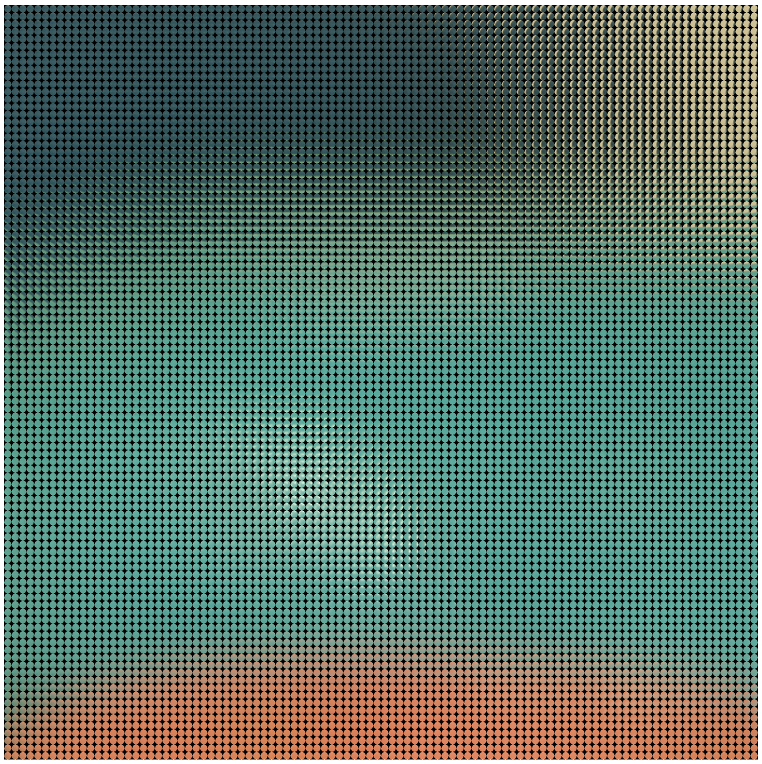
Focus Section
This section of the code produces the result of virtually shifting the focus plane by adjusting the alpha value. Additionally, the depth of field can be adjusted by altering the

aperture radius value. Both variables are located at the beginning of the ‘Focus Section’. The code normalizes all positions; S’T’, ST, and UV then iterates through all of the S’T’ pixels (1024x512) and calculates each resultant pixel’s color value by averaging all the valid light rays within the aperture’s radius. This generates the output of the final 1024x512 image with the custom virtual focal plane and aperture radius.

Results:
Aperture Radius = 40

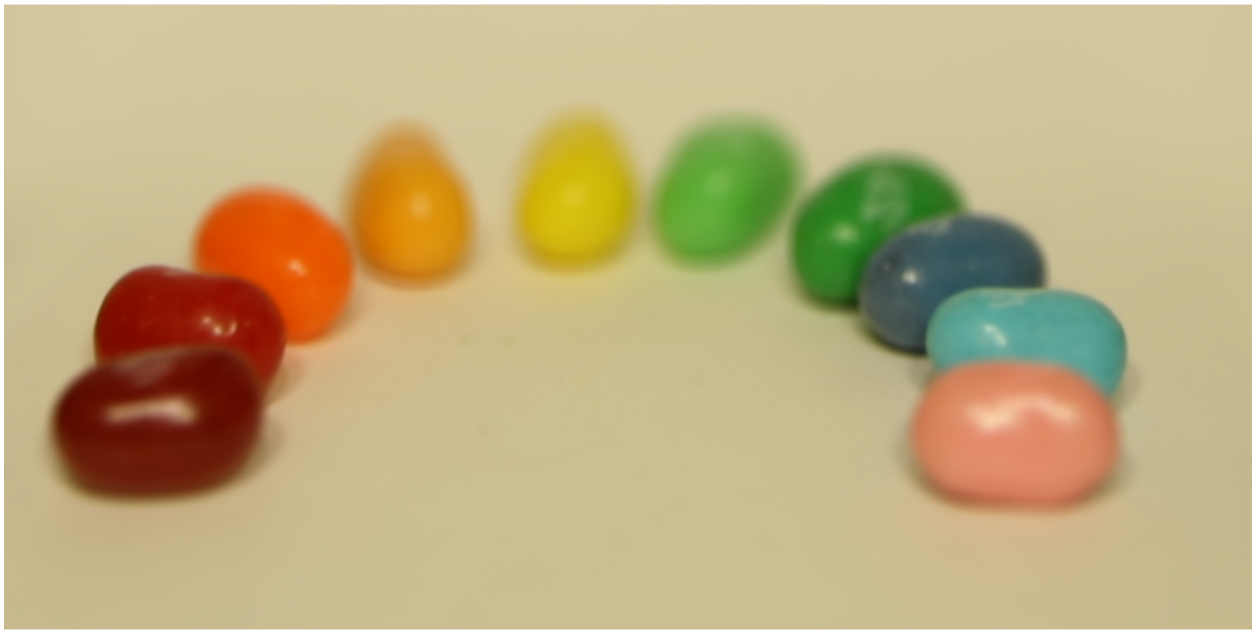


Aperture Radius = 75

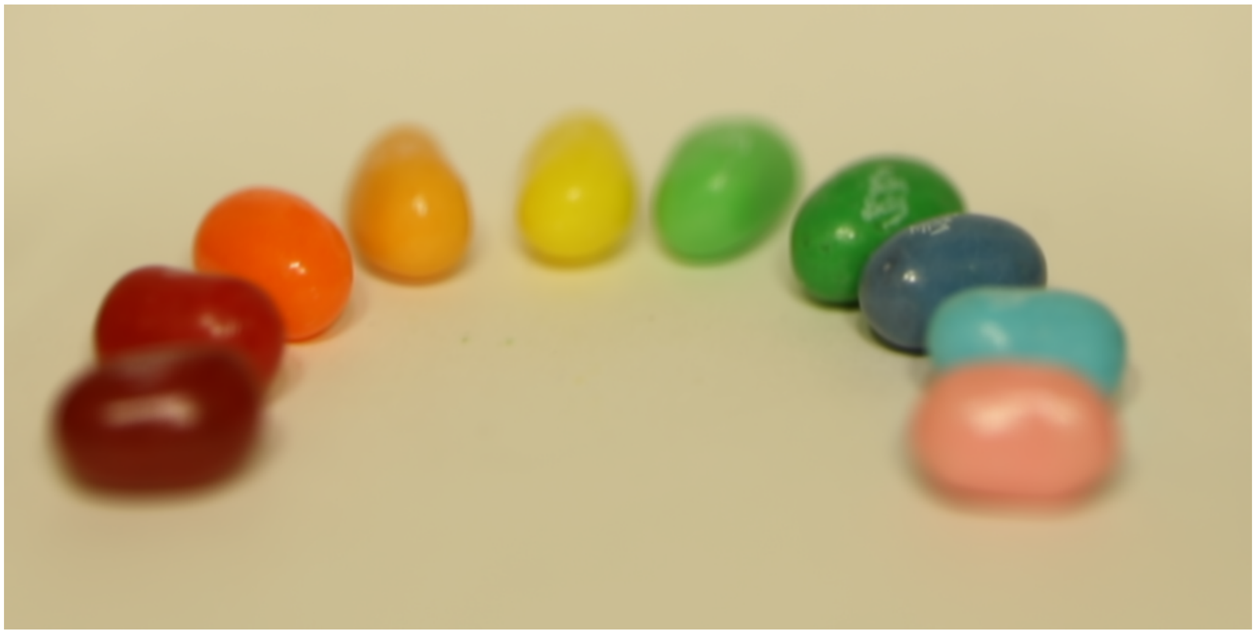


Alpha = 1.05

Alpha = 1.2



Alpha = 1.4



Alpha = 1.6

Alpha = 1.8