Nick Thompson

Dr. Morago

CSC-340-001

13 October 2019

Programming Assignment 2 – Harris Corners

Assignment, Equations Used, and Harris Corners Algorithm Description

This program loads an image and extracts Harris Corners using the following equations:

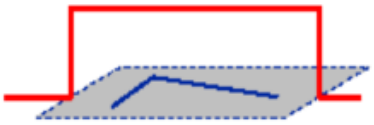
$$E(u,v) = \sum_{x,y} \underbrace{w(x,y)}_{\text{window function}} \underbrace{[I(x+u,y+v)}_{\text{shifted intensity}} - \underbrace{I(x,y)}_{\text{intensity}}]^2$$

$$E(u,v) \approx \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix}$$

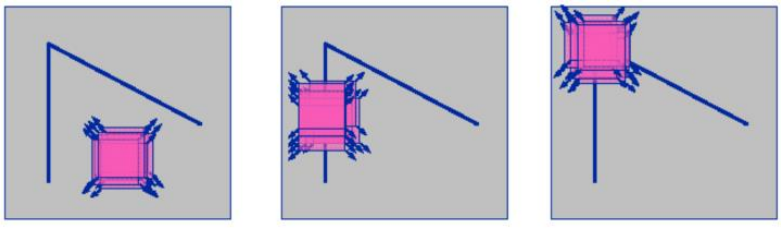
$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

The window function is a sliding window that looks one pixel in every direction (as long each pixel is in the bounds of the image) as it slides through the image. Each value inside the window is weighted the same.

Window function $w(x,y)$ =



1 in window, 0 outside



"flat" region: no change in all directions

"edge": no change along the edge direction

"corner": significant change in all directions

Note: The above three blue triangles correspond the blue triangle shown in the window function.

The following equation is then used to store the cornerness value of each pixel with k=0.05:

$$R = det(M) - k(trace(M))^2$$

$$det(M) = \lambda_1 \lambda_2$$
$$trace(M) = \lambda_1 + \lambda_2$$
λ_1 and λ_2 are the eigen values of M

These values are stored in a matrix with the same dimensions as the original image. They are stored such that each index contains a cornerness value which directly corresponds to the index of the pixel in the original image. These cornerness values are then passed to functions to draw circles on the image by either drawing corners with values above a certain threshold, drawing the top n number of corners in the image, or drawing the top n number of corners in mxm sized neighborhoods.

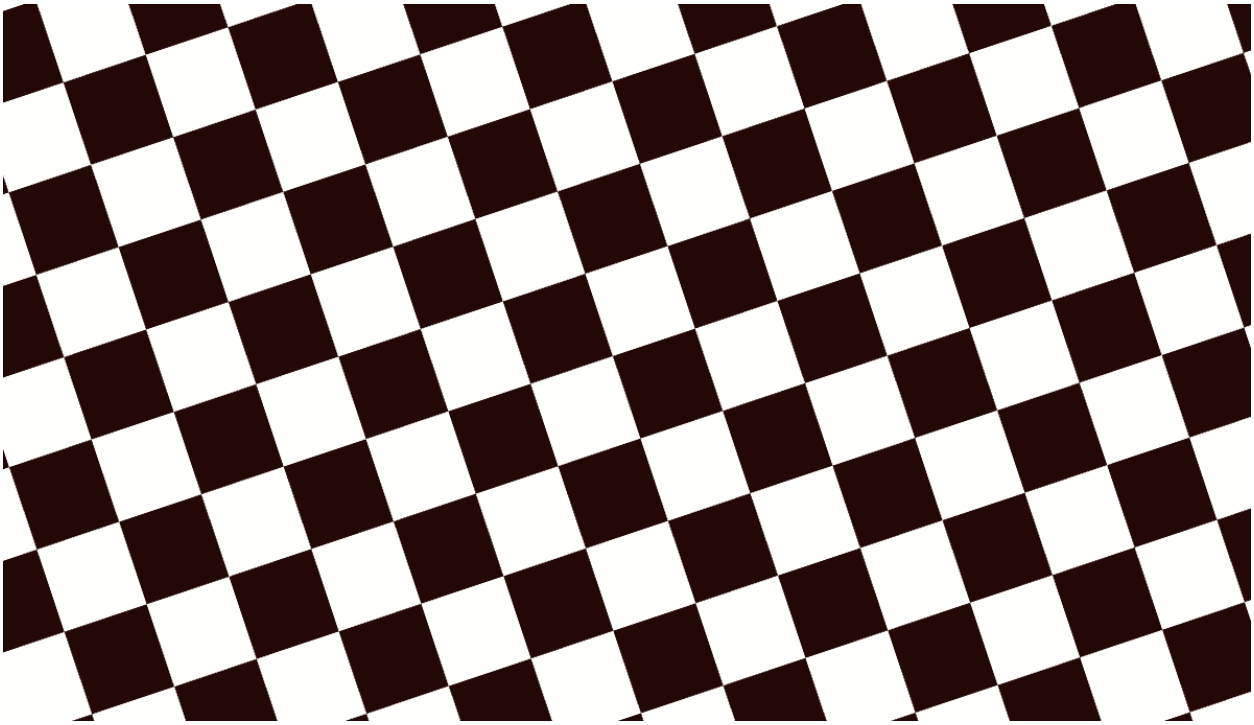


Figure 1.0: Original checkerboard image

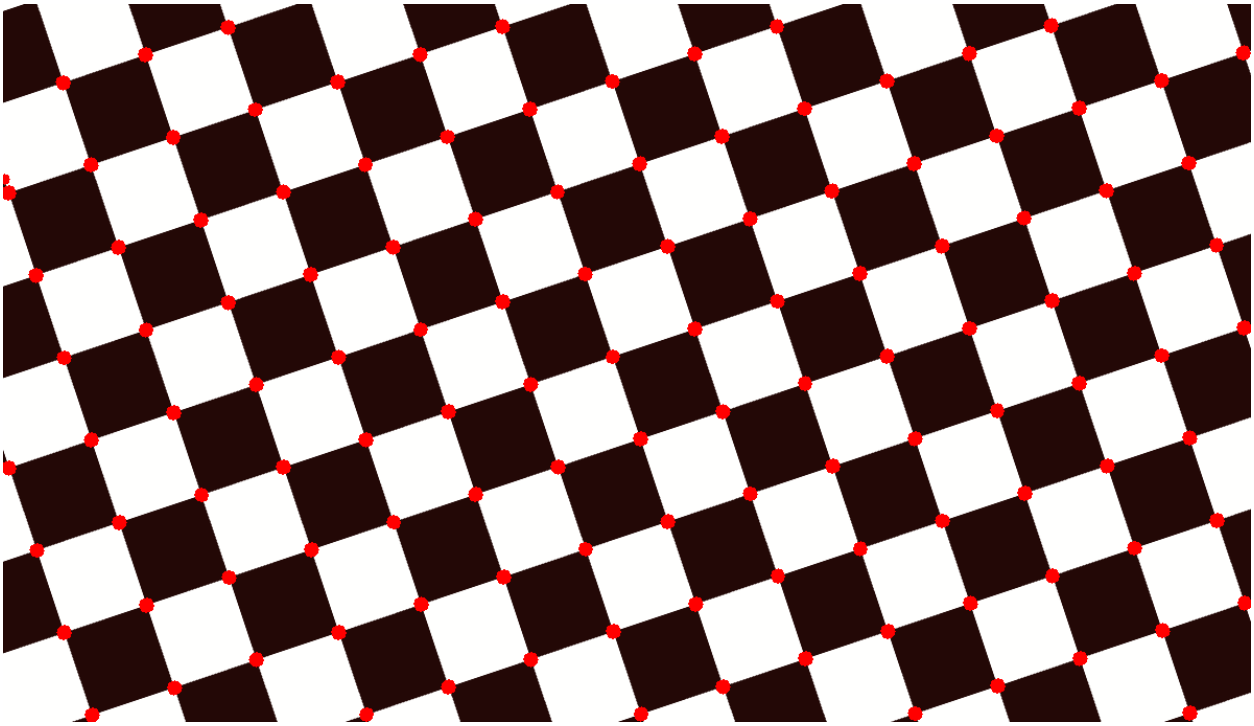


Figure 1.1: Checkerboard image with top 20% of corners drawn.

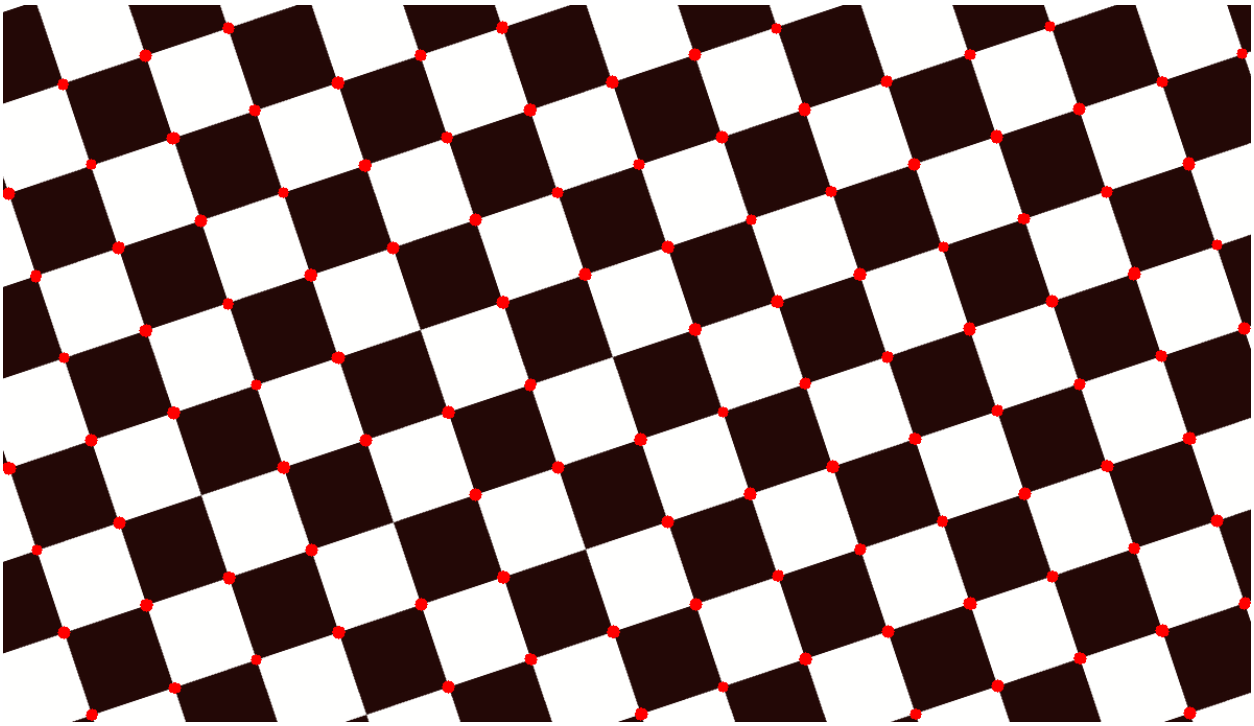


Figure 1.2: Checkerboard image with top 500 corners drawn.

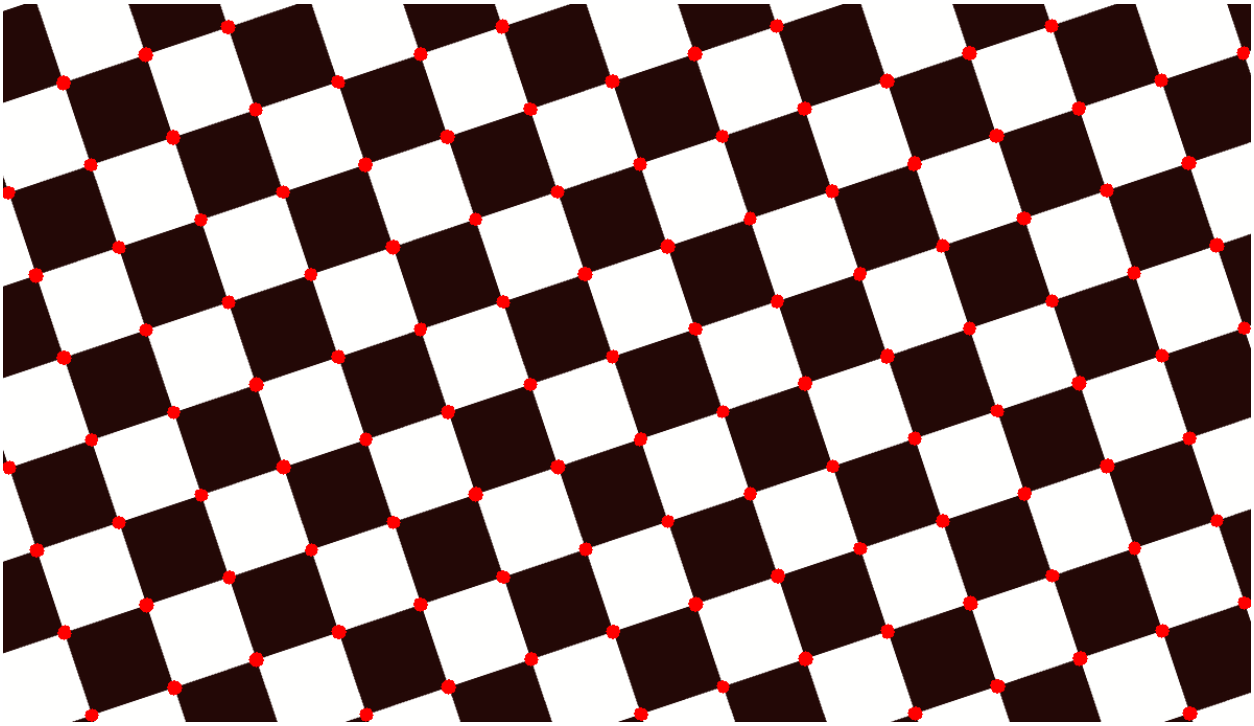


Figure 1.3: Checkerboard image with top 15 corners of 100x100 neighborhoods drawn.

Observations

Because of the uniformity of figure 1.0, drawing the top percentage of corners seems to be the best approach. The top n number of corners and the top percentile of corners seem to yield similar results; however, the top percentile seems more scalable and seems to perform well on most images with little modification. The top n number of corners requires much more trial and error to choose a value that will suit the image.

Figure 2.0: Original woman standing in a tunnel image.

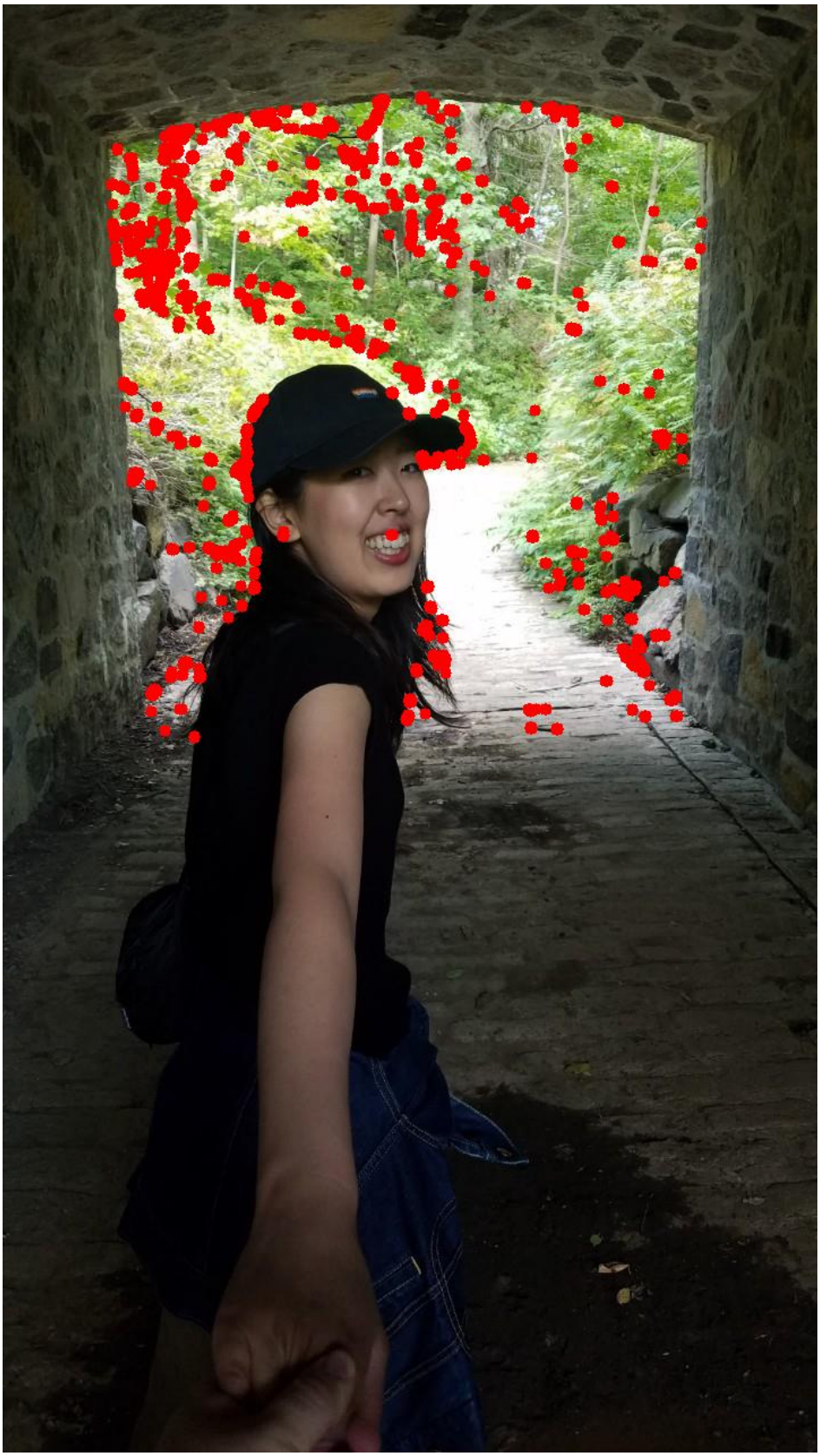


Figure 2.1: Woman standing in a tunnel image with top 10% of corners drawn.

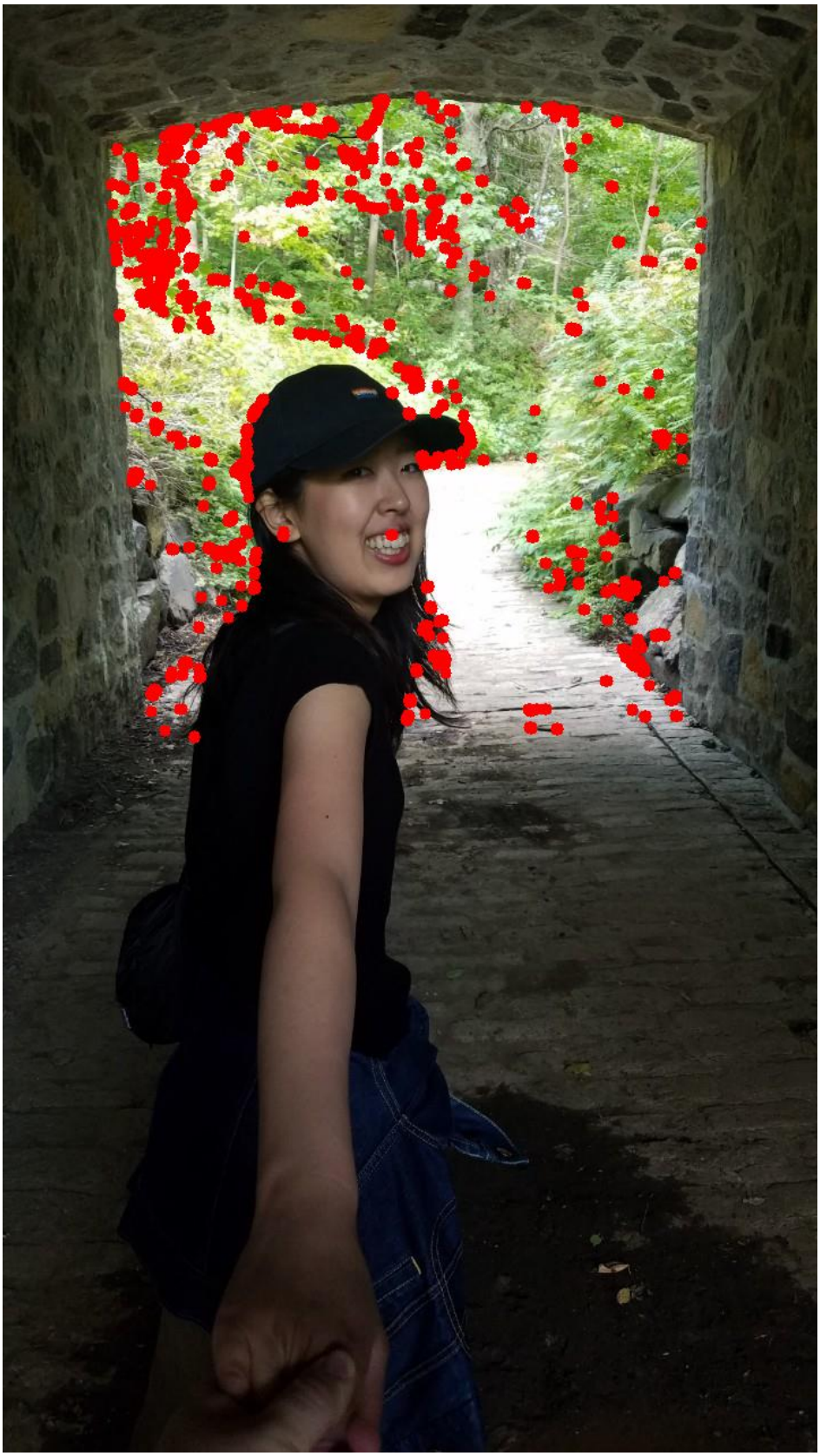


Figure 2.2 Woman standing in a tunnel image with top 2000 corners drawn.

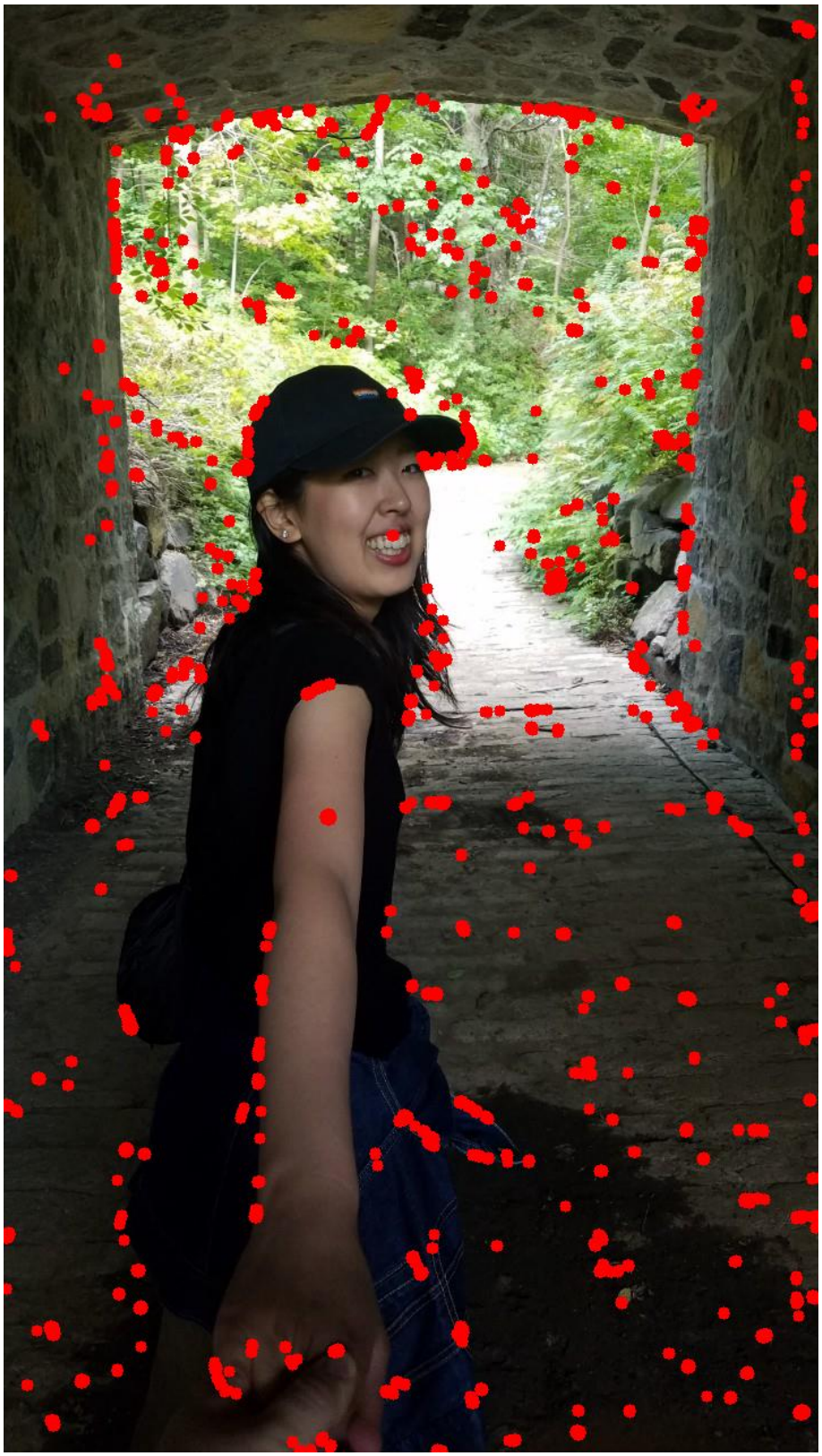


Figure 2.2: Woman standing in a tunnel image with top 25 corners of 100x100 neighborhoods drawn.

Observations (continued)

As shown in figures 2.0, 2.1, 2.2, and 2.3, drawing corners on the top percentile of corners in an image is not always the best approach. Both drawing corners on the top percentile and drawing corners on the top n number struggled to handle the severe lighting change and only drew corners where the lighting changed; the darker parts of the image seemed to be ignored entirely. The top n number of corners in mxm neighborhoods seems to perform well in these circumstances and can ignore the lighting changes. I have also noticed a lot of noise in the images that are saved as ".jpg;" however, I am unsure if most of the noise is due to the image itself or due to being a "jpg." I suspect it is a combination of both. Binary images, such as the checkboard image, have a noticeable decrease in quality of the corners found when they are stored as a "jpg." Here are the results from some of the experiments (Note: *Figure 6* is the only image below that is stored as a ".png." *Figures 3, 4, 5, and 7* are stored as ".jpg."

Results

Figure 3: Dogs playing poker image with top 10 corners of 50x50 neighborhoods drawn.

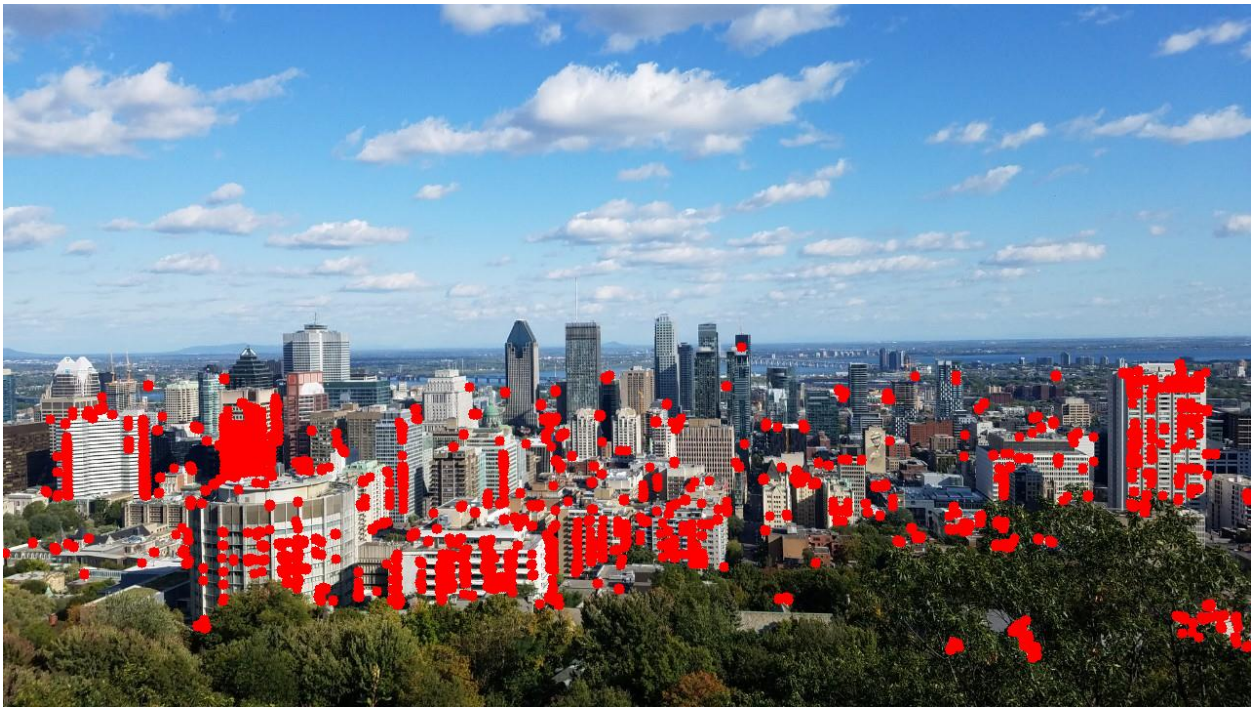


Figure 4: Montreal Cityscape with top 15% of corners drawn.

Figure 5: Quebec City with top 5000 corners drawn.

Figure 6: Cow image with top 50 corners of 100x100 neighborhoods drawn

Figure 7: Brooklyn Bridge image with top 50 corners of 100x100 neighborhoods drawn

Bonus

In addition to the original assignment, I visualized the range of cornerness values computed for an image in both grayscale and RGB. The grayscale image was very simple, but the RGB image was very difficult. Issues I encountered throughout my project will be listed below, including the issues I encounter with the RGB image.

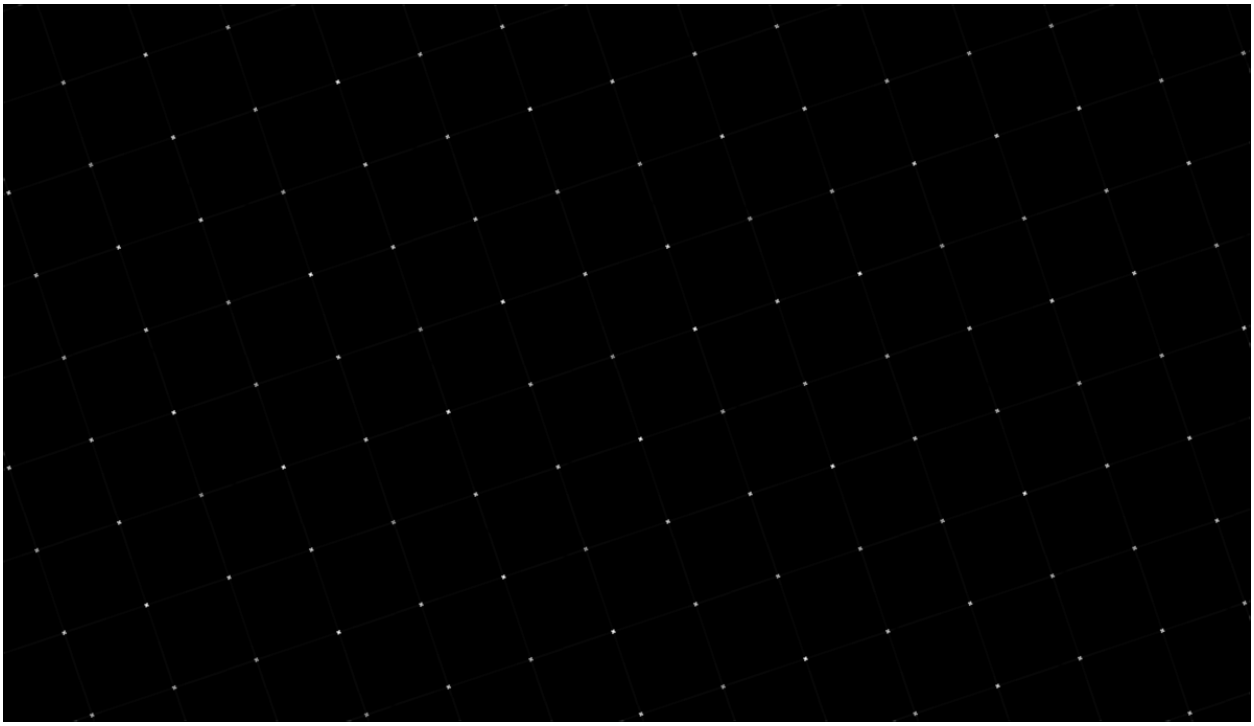


Figure 8.0: Grayscale checkerboard image

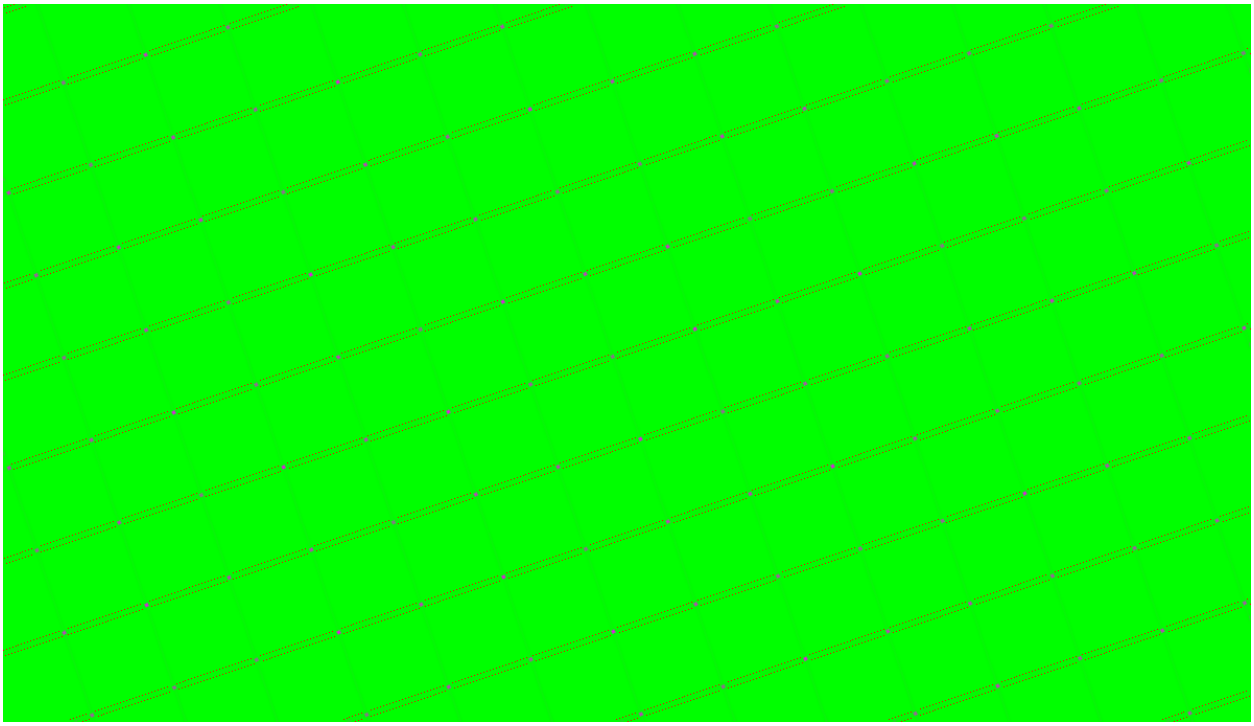


Figure 8.1: RGB checkerboard image

Figure 9.0: Grayscale cow image

Figure 9.1: RGB cow image

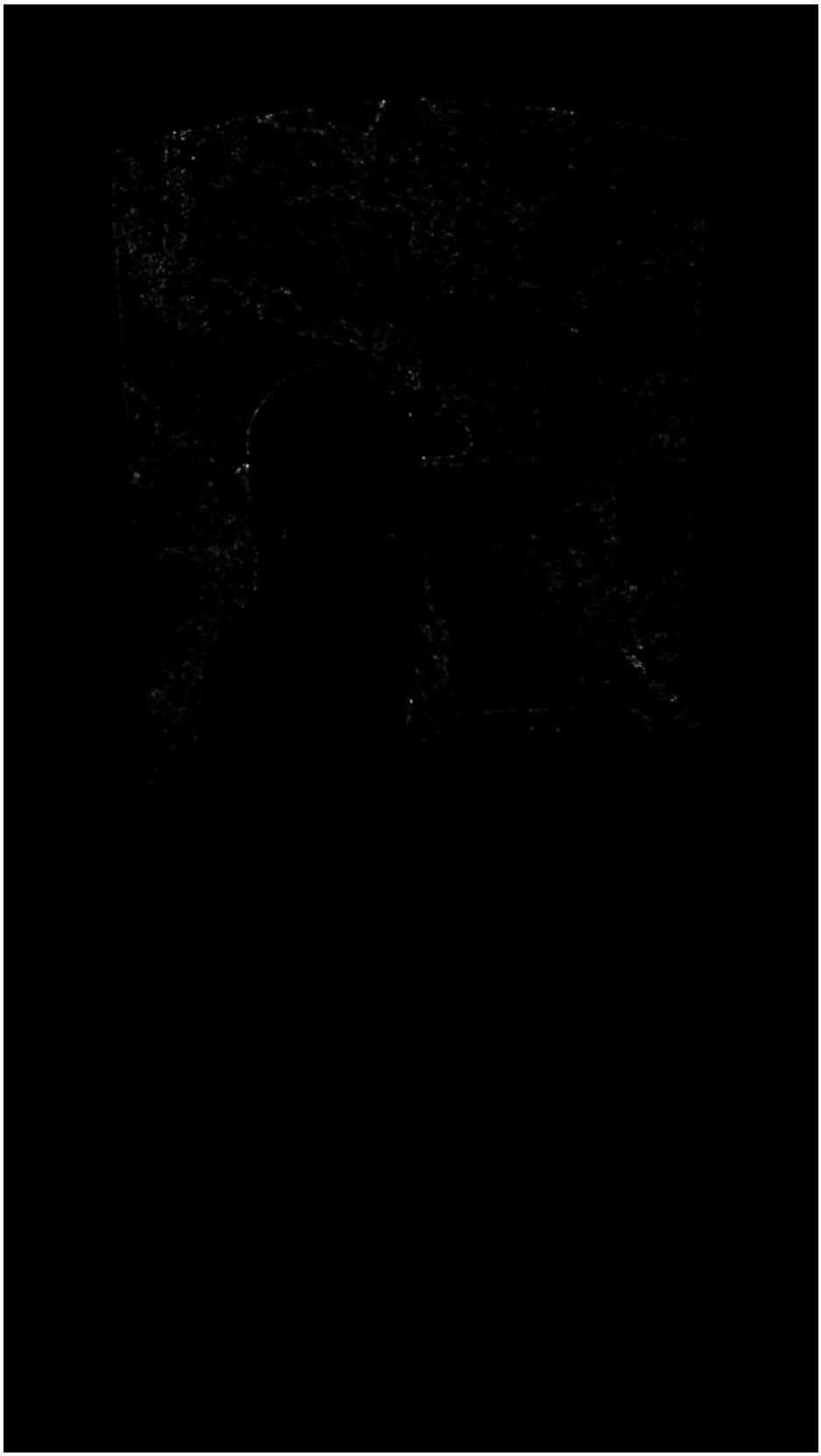


Figure 10.0: Grayscale woman standing in a tunnel image

Figure 10.1: RGB woman standing in a tunnel image

Issues Encountered

I encountered many issues along the way. The first issue I encountered was figuring out how to implement the Harris Corners algorithm. After I did that, I had the problem of drawing circles on the grayscale image which caused my circles to be gray instead of red. I also had minor issues implementing each circle drawing method. The top percentile method was the easiest among the three, and I only had problems implementing the formula to calculate the threshold. The top n number of corners method was slightly harder and took a few tries to be able to calculate the top n number of corners in one pass. The top n number of corners in mxm neighborhoods was difficult conceptually; I had much trouble figuring out a way to iterate through mxm submatrices without sliding the window (like I did in the window function). Eventually, I settled on a quadruply nested loop with the two outer loops controlling the neighborhood and the two inner loops iterating over each pixel in that neighborhood. I also had some trouble with getting the colored bonus question to display the correct colors.