Jeremy Thomas

Programming Assignment 3: Harris Corners

Project Description:

The assignment was to create a program that loads an image, extracts Harris Corners, and then displays the corners overlaid on the image. There were two methods implemented to find "cornerness" values (response value) that met a certain requirement. The first method found a threshold based on a percentage of the largest response value and only displayed corners that exceeded this threshold. The second method broke the image into several pieces (m) and displayed the largest specified amount of response values (n) within each piece. Several images were processed using the two methods while adjusting the n, m, and percent parameters to achieve the best possible corner display. I accomplished this in my code by applying three functions.

Functions implemented:

The first function calculates the Harris Corner values. It accepts an image and converts it to grayscale. Gradient values for x *(ix)* and y *(iy)* are found by iterating over the pixels in the image and calculating the difference in color between pixels within a certain range on the specific axis (figure 1).

Figure 1

$$xGradient\ (x, y) = image\ (x + 1, y) - image\ (x - 1, y)$$

$$yGradient\ (x, y) = image\ (x, y + 1) - image\ (x, y - 1)$$

The function also calculates the square values of gradient x *(ixix)*, gradient y *(iyiy)*, and the product of gradient x and gradient y *(ixiy)*. These values are stored in arrays and are used to determine the response value of a pixel by summing all square values within a certain window range of the pixel. The summation is found by iterating over the pixels in the image while setting an offset to adjust for the middle pixel within the window. Once the summations of square values are found, the response value can be found with the "cornerness" equation (figure 2).

Figure 2

$$cornerness\ equation = det\ (summation\ matrix) - 0.05\ (trace\ (summation\ matrix))^2$$

The first equation is to find the determinate of the summation matrix. I did this is my program by multiplying the summations of ixix and iyiy and then subtracting $ixix^2$. The second equation finds the trace of the summation matrix by summing the summations of ixix and iyiy. The third equation inserts the determinate and trace into the "cornerness" equation. The response values generated are added to an array and returned from the function.

The second function implemented uses the response values to determine a threshold based on the max response value in the corner value array. The threshold is changed based on a percent parameter given to the function. The function iterates over the corner value array and if the value is greater than the threshold, it marks the pixel with that corner value on the original color image.

The third function implemented breaks the pixels of the image into separate chunks determined by the parameter m. It labels the largest nth pixels determined by the parameter n within each chunk. This is done by iterating over the pixels in an image while stepping to new chunks determined by the width or height of the image divided by m. Each pixel in a chunk is then added to a corner list. The corner list is reverse sorted and is then used to find the top n values in the list. The pixels associated with these values are displayed on the original color image.

Testing:

Figure 3: The image has been processed given a threshold of .50.

Figure 4: The image has been processed by breaking the image into 10 chunks and selecting the 20 largest corner values in each chunk.

Figure5: The image has been processed with a threshold of .90.

Figure 6: The image has been processed with a threshold of .99.

Figure 7: The image has been processed with a threshold of .97.

Figure 8: The image has been processed with a threshold of .97.

Figure 9: The image has been processed by breaking the image into 20 chunks and selecting the 10 largest corner values in each chunk.

Figure 10: The image has been processed by breaking the image into 10 chunks and selecting the 20 largest corner values in each chunk.

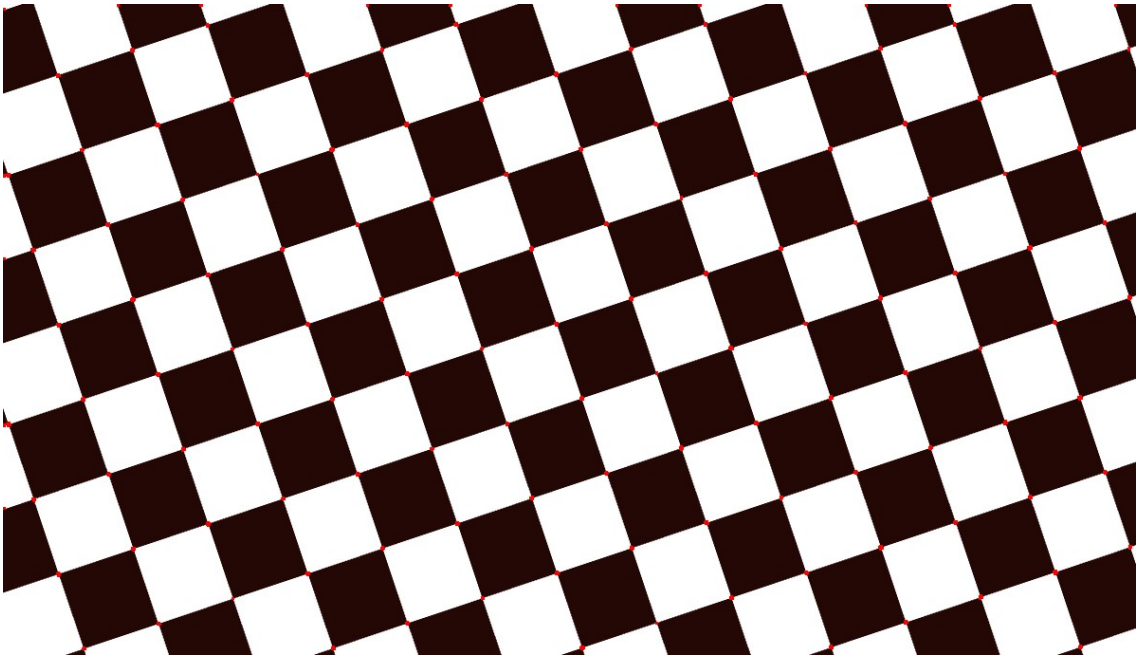

Figure 3

Figure 4

Figure 5

Figure 6

Figure 7

Figure 8

Figure 9

Figure 10

Conclusion:

Images with less rigid displays of corners take a much higher percentage value leaving a much smaller threshold for pixels to be displayed. Examples where corner labeling is based on purely threshold, we stick to areas in which lighting is greater or rigidness of corners is more defined.

When defining corners using the chunk method we find more “corners” further from the concentration of the light and rigidness. This allows for corners to be found in areas where the lighting may be significantly darker.