ECS 174 – HW#1

Section 1:

I believe I’ve left sufficient notes and comments in my code to explain most of what I am doing. I don’t think any of the code is necessarily complicated to begin with either.

Section 2:

1. By using the *time* library, I am able to time how long it took each function to run.

|  |  |  |  |
| --- | --- | --- | --- |
| Function | Sigma = 2 | Sigma = 4 | Sigma = 8 |
| GaussianBlurImage | 484 | 452 | 413 |
| SeparableGaussianBlurImage | 944 | 905 | 816 |

For GaussianBlurImage, the difference between each sigma seems close to linear. (32s between 2 and 4, and 39 seconds between 4 and 8). So for a sigma of 32, I think a runtime of 330ish is a good estimate. As for SeperableGaussianBlurImage, an exponential trend can be seen. (39 between 2 and 4, and 89 between 4 and 8). This is likely due to the function running 2 filters over the same image. At a sigma of 32, I think 590 is good estimate.

1. [MOIRE]

This is kind of a hard question. Specially since the image gets so small its hard to see artifacts. But I think that to avoid most artifacting, we can use a sigma of 2/scale (so for a scale of 1/8, a sigma of 4 at the least). And to avoid most artifacting, a sigma of 1/scale works best. We can see so in the following examples:

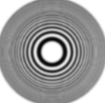
On ¼ the scale, we can see a small amount of aliasing in “sigma 2”, and much less in sigma 4 and 8. However it is also very hard to make out the photo anymore in sigma 8.

On 1/8 scale, aliasing is very apparent in “sigma 2”, and less apparent in “sigma 4” however its still there. And in “sigma 8”, it is very hard to see aliasing, however it is also hard to see the picture

¼ scale: sigma 2 sigma 4 sigma 8

A picture containing electronics

Description automatically generatedA picture containing device, cymbal

Description automatically generated

A close up of a camera lens

Description automatically generated with low confidence1/8 scale: sigma 2 sigma 4 sigma 8

[SEATTLE.JPG]

Seattle was even trickier. Here are 4 photos, one with no blur, and the other three are with a blur using sigma 2, 4, and 8. I’ve provided a zoomed in preview and a normal preview for comparison

[4] Sigma=8

[3] Sigma=4

[2] Sigma = 2

[1] No blur

In [1], it is clear to see that there are issues in both the large preview and the small preview. The large one has no details, and the small one looks almost over exposed.

In [2], the large preview retains a lot of detail, however if we zoom in and compare with the original picture, there are pieces of details that shouldn’t be there, The small preview however looks arguably the best. You can see a lot of details in the small photo, without it being blurry or overexposed

In [3], we have the best balance in terms of details, small preview and large preview.

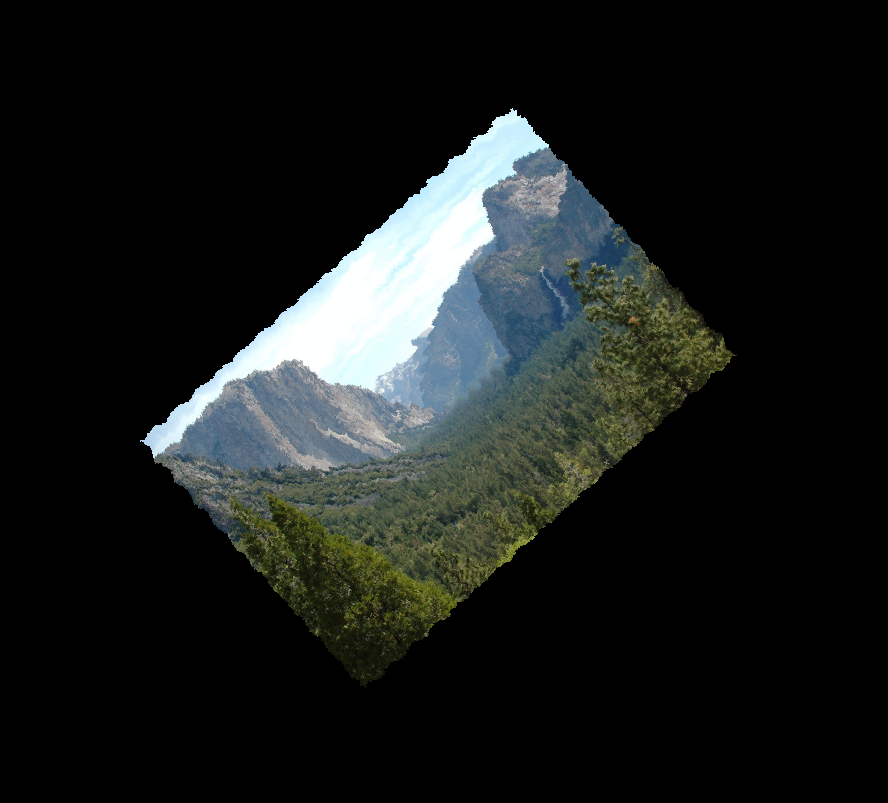
In [4], the small preview is alright but too blurry, and the big picture doesn’t have an issue of introducing new details, however it is also just lacking detail since it is so blurry.

For a complicated picture like Seattle, a sigma of 2/scale seems to fit best, however I think the most optimal way would probably be to use a dynamic sigma. Since it looks like different images demand different requirements

A person climbing a rock

Description automatically generated with medium confidenceIt was actually very hard to see a line that wasn’t visible through the sobel filter as you can see below. The only things I could find were the guy’s foot, and a tiny railing in the background (Yellow arrows). Then there were edges that I can see if I look hard enough, however they are not very apparent. (Blue arrows)

1. A picture containing mountain, nature

   Description automatically generated

20 rotations @ 2 degrees

One 40 degree rotation

The pictures do not look the same. This is likely due to the fact that rotation is a non-linear function. Looking a bit further into the imrotate function, it uses interpolation for rotations. Since the image is getting slightly bigger with slightly more black borders around every rotation, I suspect the interpolation with the black edges is what is causing the squiggles.