Ph 21 Homework 3

Emily Springer

February 14, 2020

1 Question 1.

I imported the following image into my local directory.

$1 TRILLION
Apple

2 Question 2.

To open this image in python, I used Python Imaging Library with the following code:

$$im = PIL.Image.open(r"C:/Caltech/Ph21/hw3/image.jpg")$$
$$print(im.format, im.size, im.mode)$$
$$im.show()$$

The ouput for the format, size, and mode were "JPEG (890, 501) RGB"

3 Question 3.

I implemented edge detection using FIND_EDGES. I added the following lines to my previous code to get the edges.

$$imageWithEdges = im.filter(ImageFilter.FIND_EDGES)$$
$$imageWithEdges.show()$$

4 Question 4.

To blur the image, I used ImageFilter's GaussianBlur function by writing

$$im = PIL.Image.open(r"C:/Caltech/Ph21/hw3/image.jpg")$$
$$blurred = im.filter(ImageFilter.GaussianBlur)$$
$$blurred.show()$$

The edge pixels can be treated such as when the magnitude of the rgb vector greatly changes.

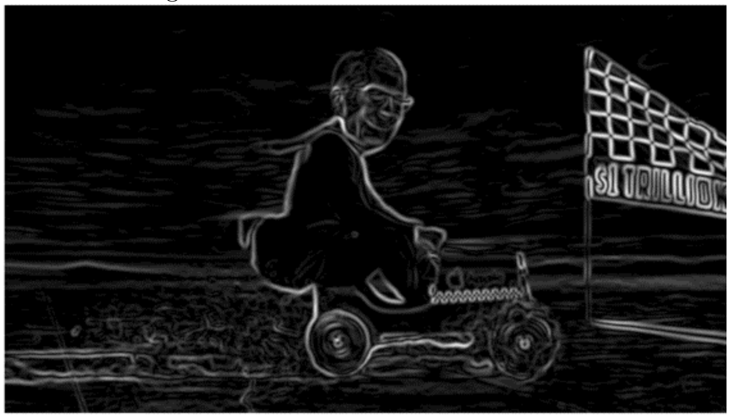
5 Question 5.

I used the first derivative of the Gaussian with a sobel filter to detect the edges. My sobel filter function converted the rgb part of the image to magnitude of black to match the dimension of the transformation matrices. My transformation matrices were

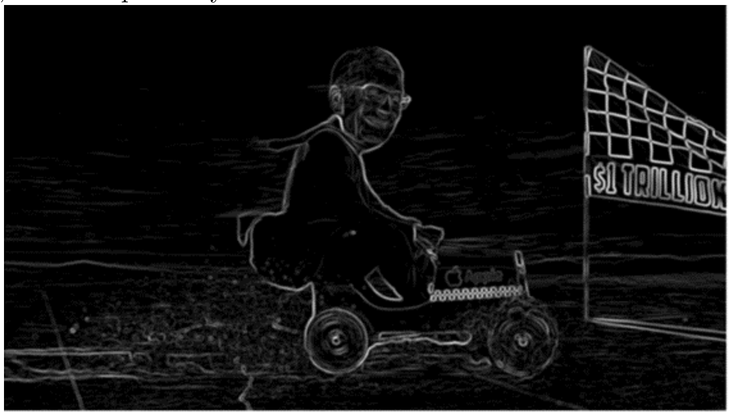
$$Kx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], np.float32)$$
$$Ky = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]], np.float32)$$

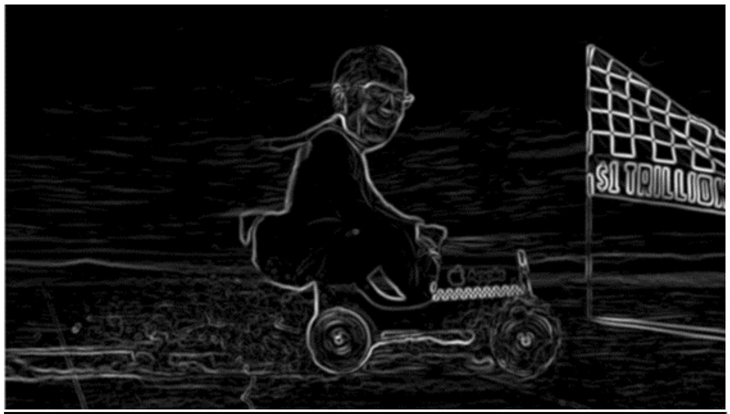
Then, I convolved the image with each k and y matrix before taking the hypotenuse of them to get the magnitude of the edge at each point. I created a cutoff for the magnitude, which I arbitrarily set to 15. I converted the magnitude of the edges to an image using PIL.Image's fromarray function.

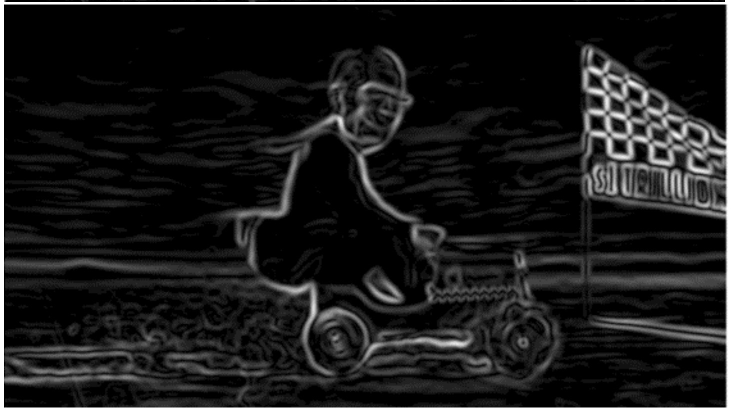
Here is the edge detection with the default blur radius of 2:



I then proceeded to try different blur radii. Here is the edge detection with the default blur radius of 1, 1.5, and 3 respectively:



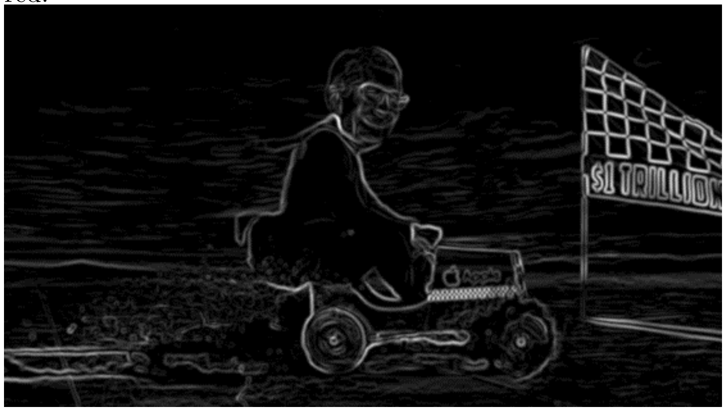




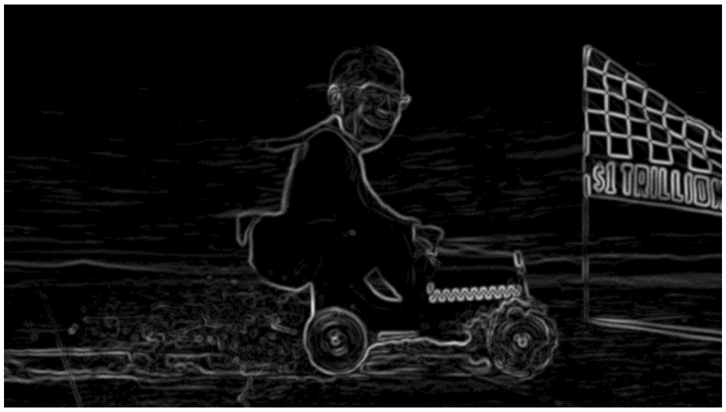
6 Question 6.

I altered the program to only take red, green, or blue components instead of the magnitude of all of them. The result was

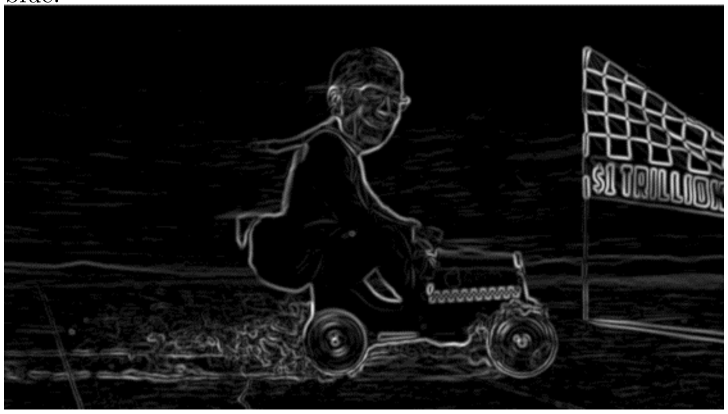
red:



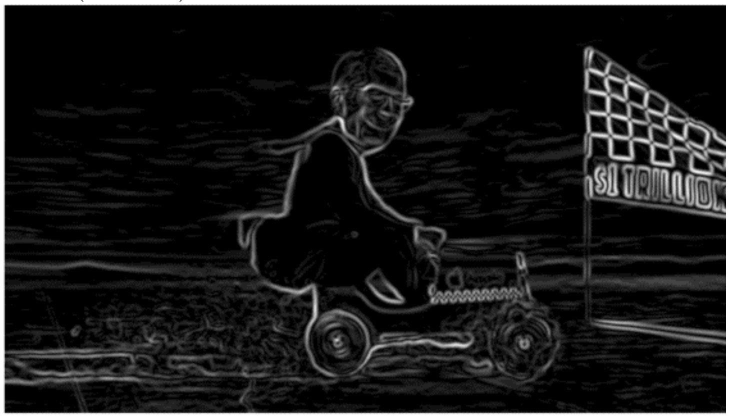
green:



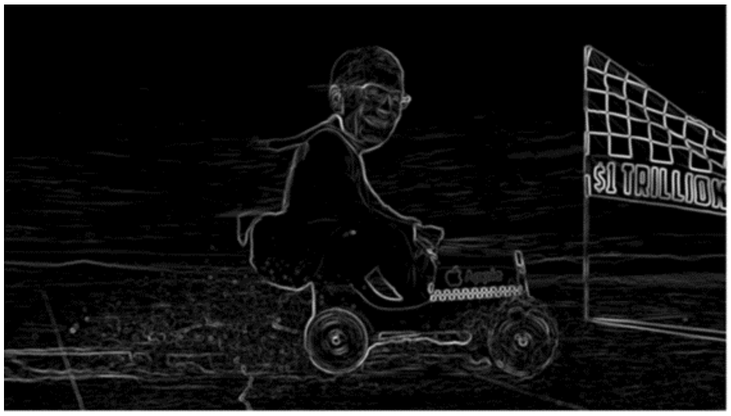
blue:



Here is a comparison of with and without blurring:
with: (radius 2)



without:



The signal-to-noise ratio doesn't greatly change, but blurring the image reduces the noise, increasing the ratio.

Edge detections can be used to defeat CAPTCHAs because they often only require identifying objects in images, which edges would greatly help with. Another use would be AI vision, such as for self driving cars to identify obstacles.

7 Question 7.

Edge detection allows for automated analysis of images. There are many parameters that can be altered to optimize the signal-to-noise ratio.