Connor Lowe

1573616

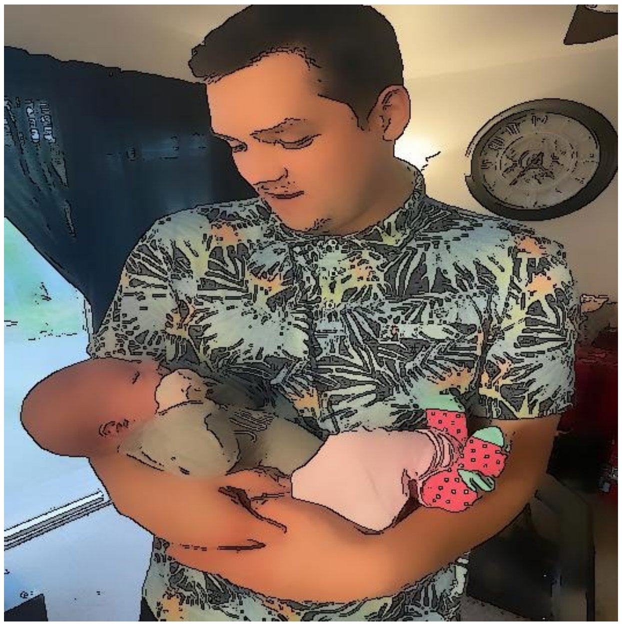
EE 440 Final Project

Cartoon Effect

Effect I wanted to achieve - Cartoon

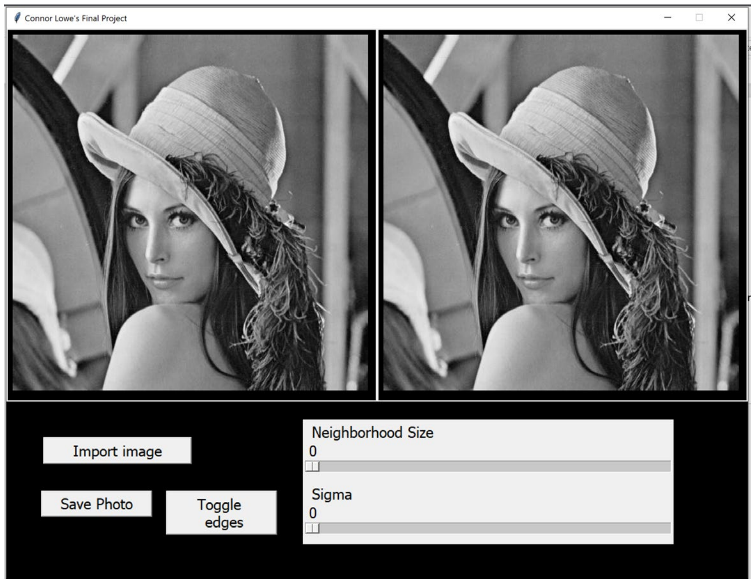
My goal for this final project was to implement a cartoon filter. I planned to do this by implementing a bilateral filter and then enhancing the edges of the image. This kind of effect was always interesting to me, and I was excited to implement it on my own photos. An example of the output I wanted to achieve is below.

I overall did achieve my goal of the cartoon effect and have included an example of my output below.



Implementation

After you open my jupyter notebook and run the code cell the below app will pop up.



I have implemented three different Tkinter buttons.

The first button 'Import image' allows a user to search for an input any image they want. This was done in my browsefunc callback function which calls the askopenfilename function. To deal with different sized images, I rescaled the input image to the max height of the GUI. The scaling is not uniform so the image may become squished or stretched horizontally during this scaling. I chose not to include uniform image scaling, because I though it contributed positively to the cartoon output.

The second button I implemented was the 'Save Photo' button which allows the user to save the output cartoon image to somewhere on the computer. This was done in my savePhoto callback function which calls the Pil.Image.save function. The only file you can save to is a .jpeg file, so for a user to save the output properly only the .jpeg or no extension should be used, if another extension is used the image will not save correctly.

The last button I implemented was a toggle switch 'Toggle edges' that adds the edges of the image into the output to make the cartoon look more distinctive. Toggle edges is not enabled by default, and you can tell when it is, because the button will stay pressed. The edges are implemented using a canny edge detector through cv2, due to the efficiency of the code, and the need for the GUI to run in real time.

The two sliders on the bottom right control parameters for the bilateral filter used to smooth the image while also preserving the edges. The neighborhood size is the diameter of each pixel used in filtering, and the sigma value the farther away colors will be mixed to result in larger areas or semi-equal color. This was implemented also using cv2 due to the constraints on the project needing to run in real time. I did experiment with other convolution filters and my own implementation and have included that above, but my code was not efficient enough to run decently in the GUI. This is something I would fix if time allowed.

How to run the GUI

To run the GUI you will need to install anaconda and be able to run a jupyter notebook file, and then run the code cell in the jupyter notebook file. There are also some dependencies on packages previously used in this class but should run the same way as a regular jupyter notebook file.

What I learned

Overall, in this project I learned how to apply a multitude of filters to images to produce a desired result. I was able to experiment with applying convolutional filters to produce effects such as smoothing with a gaussian distribution, and preforming a weighted average with a bilateral filter, which performs a smoothing and preserves the edges. Canny edge detection was also an interesting thing to experiment with, which smooths an image, finds the edges, and preforms thinning to detect a wide range of edges. In the future I would like to implement the code further myself and use techniques such as broadcasting in numpy arrays to make my code run in real time. Another aspect of this project that really sparked thought was the implementation of the GUI and how Tkinter works. In the future I would love to add functionality and make the GUI more visually pleasing as well. I would also like to try more specialized filters such as high boost filtering.