**15-112 Darkroom**

**Name: Tony Tao**

**Mentor: Alan Hsu**

**Description:**

Darkroom aims to be a light photo editing software that allows users to blur, sharpen, and denoise images with customizable profiles and kernel sizes. Additional features may include adjusting brightness, tweaking saturation, changing contrast, changing hue, cropping, resizing, as well as automatically carrying out functions on specified objects such as faces.

**Competitive Analysis:**

The competitive landscape for a product such as darkroom is quite varied. Many products such as Adobe Photoshop do give users a lot of control over different functions but have a very steep learning curve. Darkroom aims to be a photo editing tool that delivers the some of the same functionality as lighter photo editing tools such as Adobe Lightroom in a streamlined, easy to use package.

**Structural Plan:**

As shown in my storyboard, my project will be organized using a menu system that will allow me to keep the main screen as clean as possible. There will be a tool bar on the left side, and the image will automatically be fitted into the space on the right, regardless of actual image resolution. Again, this is supposed to be a light photo editing tool and meant for quick touch ups, so the interface will not be as sophisticated as some of the other tools on the market.

**Algorithmic Plan:**

1. Image convolution – I am using 5 nested for loops for image convolution, the outer to loop through each of the red, green, and blue channels, the next to loop through the rows of pixels, the next to loop through the columns of pixels, and the last two to loop through each drow and dcol so that I can go through the kernel.
2. Gaussian Kernel – I am using the two dimensional gaussian distribution to generate a gaussian kernel to do gaussian blurring
3. Bilateral Filter – I am using two weights to take into both the spatial distance and the photometric distance between a neighboring pixel and the central pixel. Each distance is also passed into a gaussian model to create a gaussian distribution.
4. Haar Cascades – I am using opencv’s detect multiscale with a frontal face cascade to detect facial features in images. Opencv’s detectmultiscale compares the features of an image with over 180,000 features in the xml file that was provided by their haar cascade. Before and after using opencv’s method, I need to do some image preprocessing and data postprocessing to make sure that I can accurately get the faces. Some of this processing includes using masks, thresholds, and blurs to make the faces easier to detect.
5. Text Detection – I am using pytesseract’s image\_to\_data function which creates a dataframe based on the text data detected. To more easily get text data, I use cv2’s filter2D function with a ddepth of -1 and a Laplacian kernel/matrix/filter which gets the edges of the text to make it easier to detect.

**Project Timeline:**

**Version Control Plan:**

I am currently using github to version control. I usually commit as often as I remember to, so that means multiple times per day. I can commit changes directly from vscode. <https://github.com/tonytao728/termProject>

Graphical user interface

Description automatically generated

**Module List:**

OpenCV

Pytesseract

Pillow

Numpy

Tkinter

Os

Math

Time