Jackie Joyce
CS 4475
Spring 2015
2-10-2015

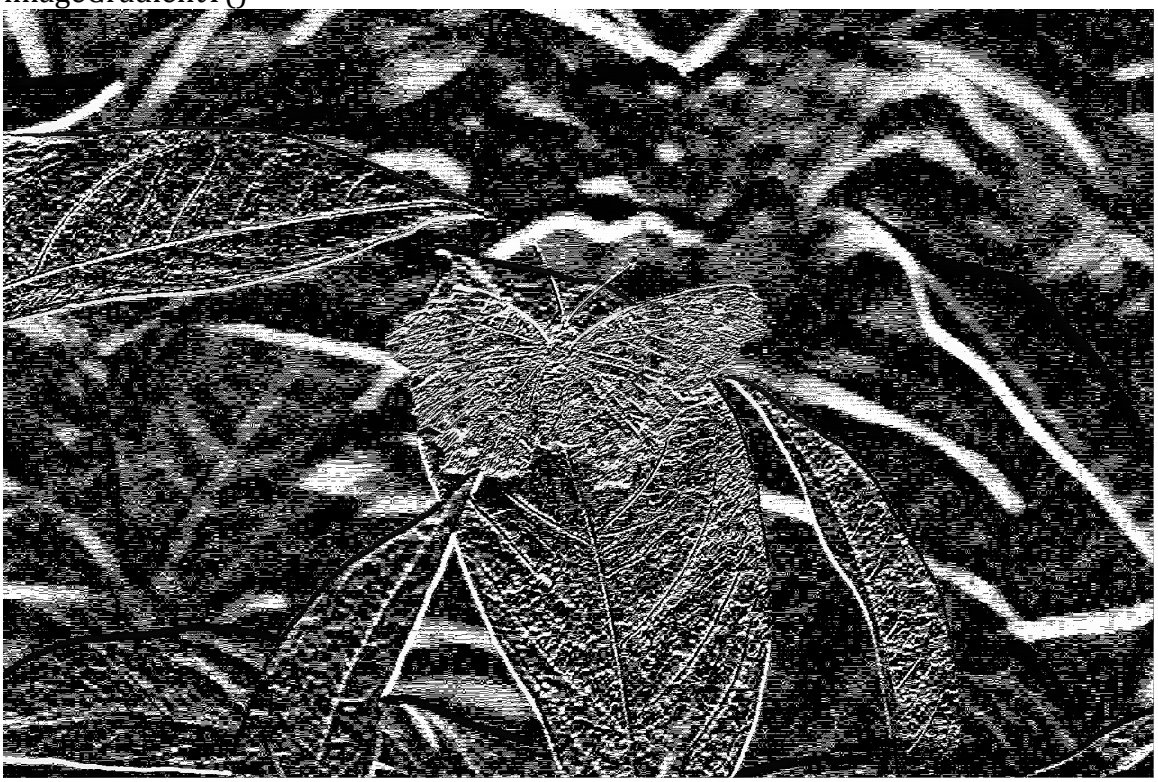
Assignment 5

Methods 1 & 2

For both the imageGradientX() and imageGradientY() methods, I first got the shape of each image. I knew that I would need to use two for loops to reach the pixels in the rows and columns of the numpy arrays. When differentiating in the X direction, I made sure to stay inbounds by subtracting my height by one (since we are comparing columns) and vice versa for differentiating in the Y direction. In this loop, I then simply applied the equation given to us in the assignment. I used cv2.imwrite('image name', image) to check my output. I them returned the images. Below are the two images that I got as outputs:

imageGradientX()

imageGradientY()



Method 3

I was a bit stuck on the computeGradient() method. At first I tried getting the average of the kernel using np.average() and applying a similar function to the first two methods in an attempt to get a blur. This produced some really awesome images, but not what I was looking for! After talking to the professor and Alex in the class I them realized that the kernel is a 3x3 np array of averages! And the average of these times a 3x3 section of the image should be stored and the middle pixel should be set to this value. To do this to every pixel, I created a double for loop and then multiplied and stored the values of the two 3x3 arrays and set the middle pixel to the sum of the stored values. Because of the for loop, the code will iterate through each pixel in the range of width – 1 and height – 1. I got the following image:

The image is not extremely blurry because a 3x3 kernel was applied to a tremendously large image.

Edge Detection

I toyed with the idea of using Photoshop and used the online photo editor photo-kako to create edge detection images, but I wanted to test it out with Python code. I went to the documentation for open CV. Canny and followed their example minus the matplotlib library and got the images below. I did mess around with the matplotlib library a bit and created a few windows to display image comparisons. I was very surprised that I was able to use the Canny function on its own for edge detection. I guess I just like to over complicate simple things. I did a blurred image using the computeGradiant() function and a regular gray scale image of a flower. I wanted to compare and contrast the results to see the effect that burring an image has on its edges.

Image blurred with kernel

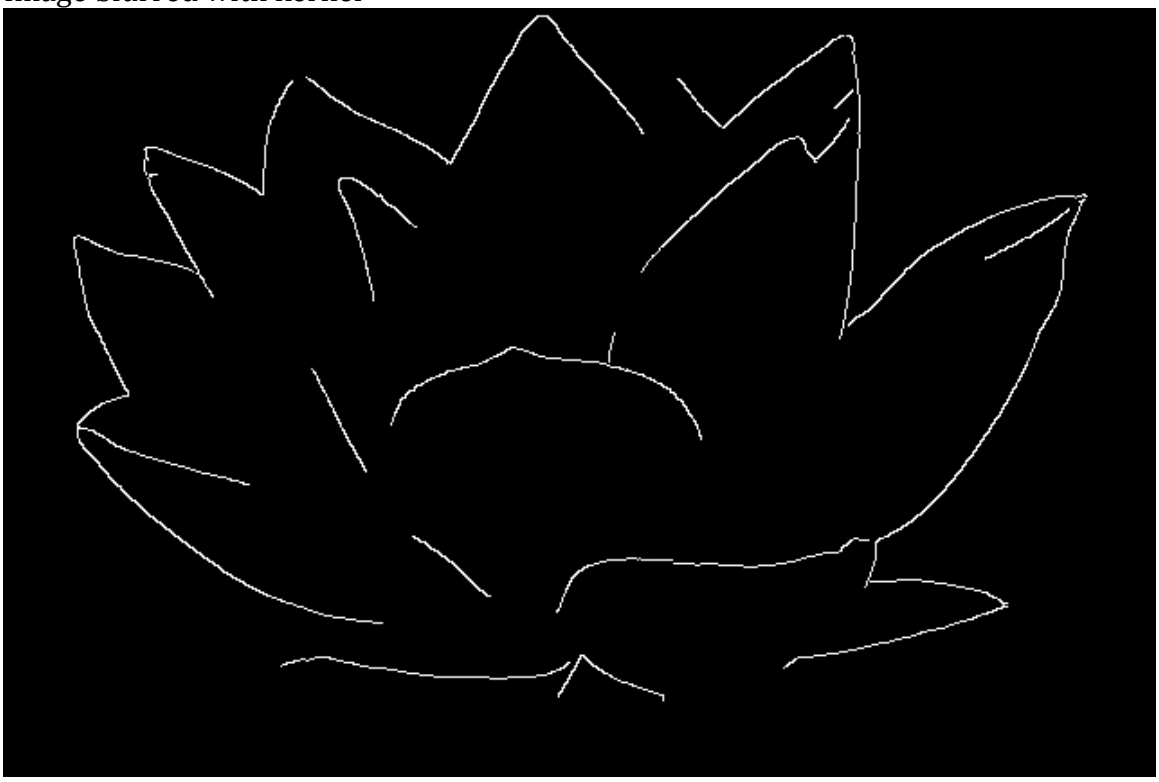
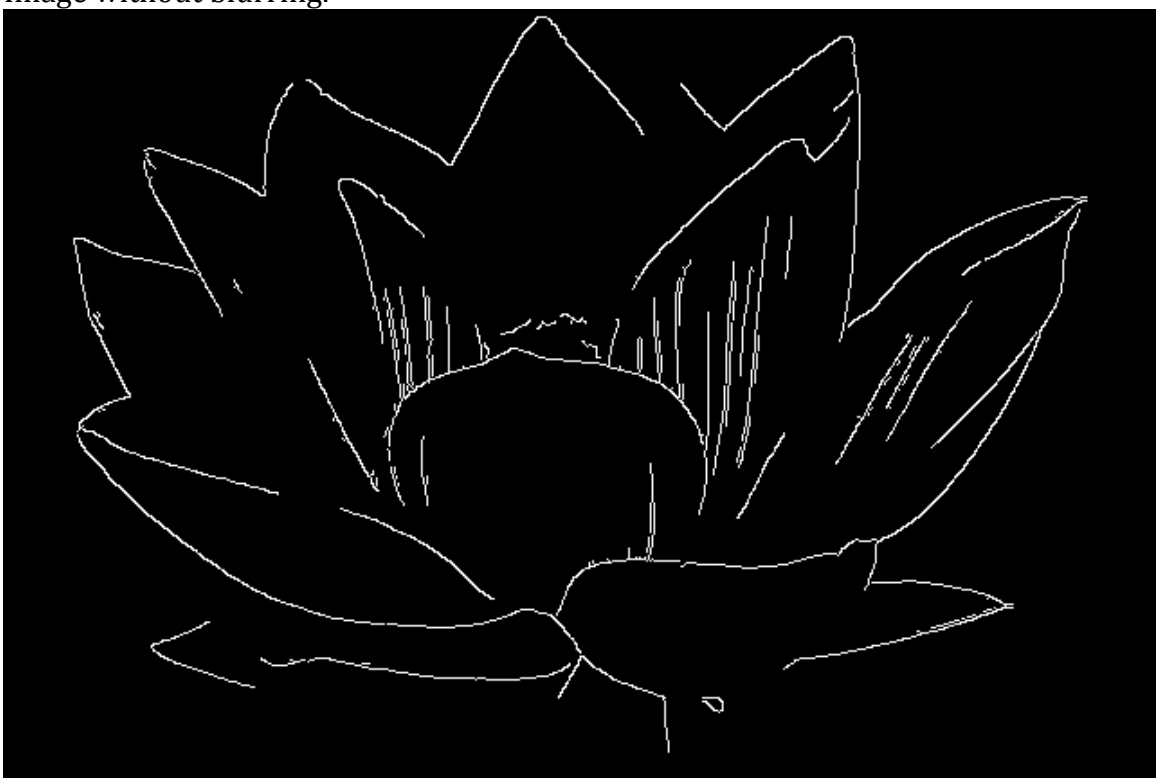


Image without blurring.



Original image

As one can plainly see, there are less pronounced edges in the blurred version and one is left with the “main” edges. The main edges are mostly the outline of the flower against the dark background while in the non-blurred image some of the lines on the petal still appear. I used the code below to create these images:

```
import cv2
import numpy as np
import scipy as sp

img = cv2.imread('flower_gray.jpg',0)
edges = cv2.Canny(img,100,200)

cv2.imwrite('edges 2.jpg', edges)
```