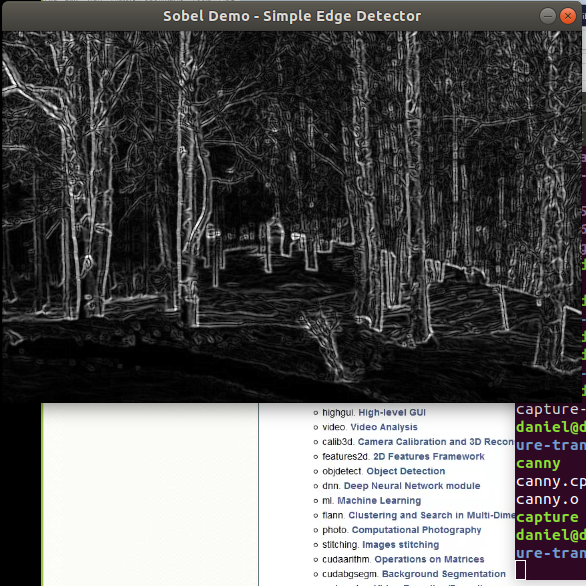
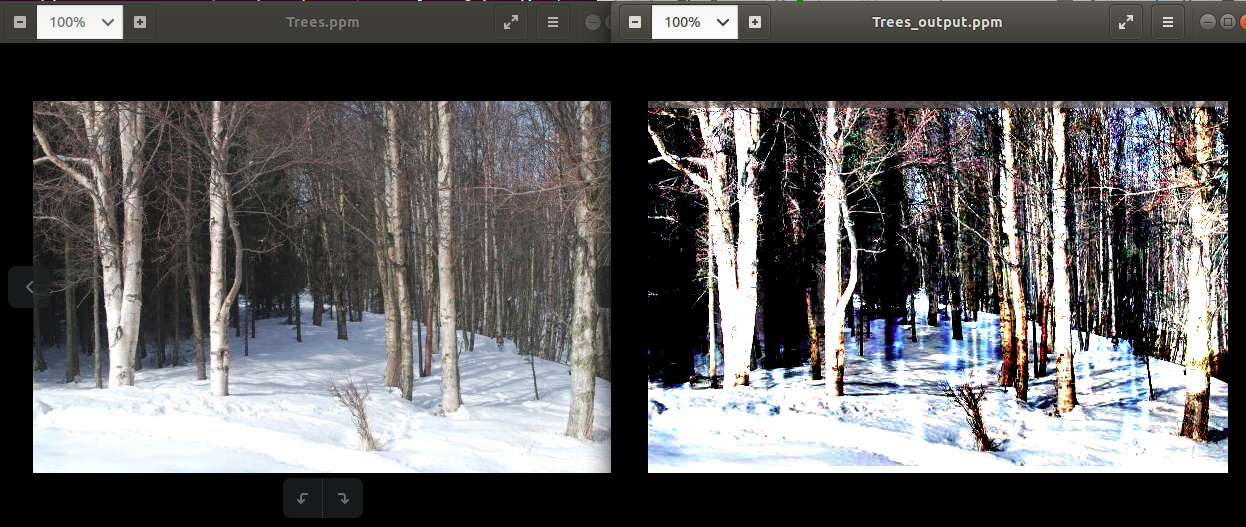
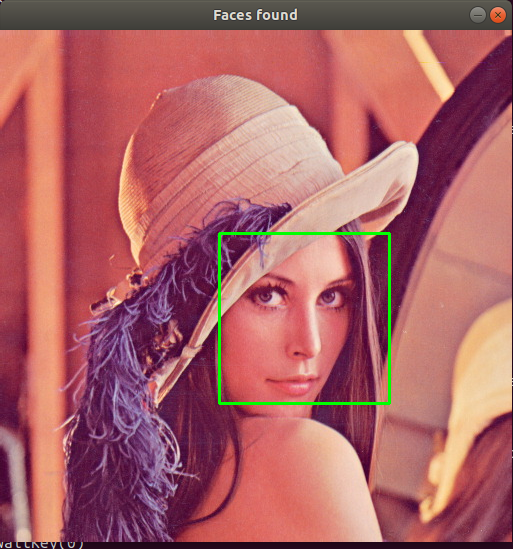
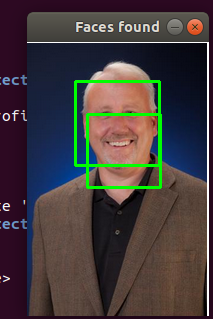
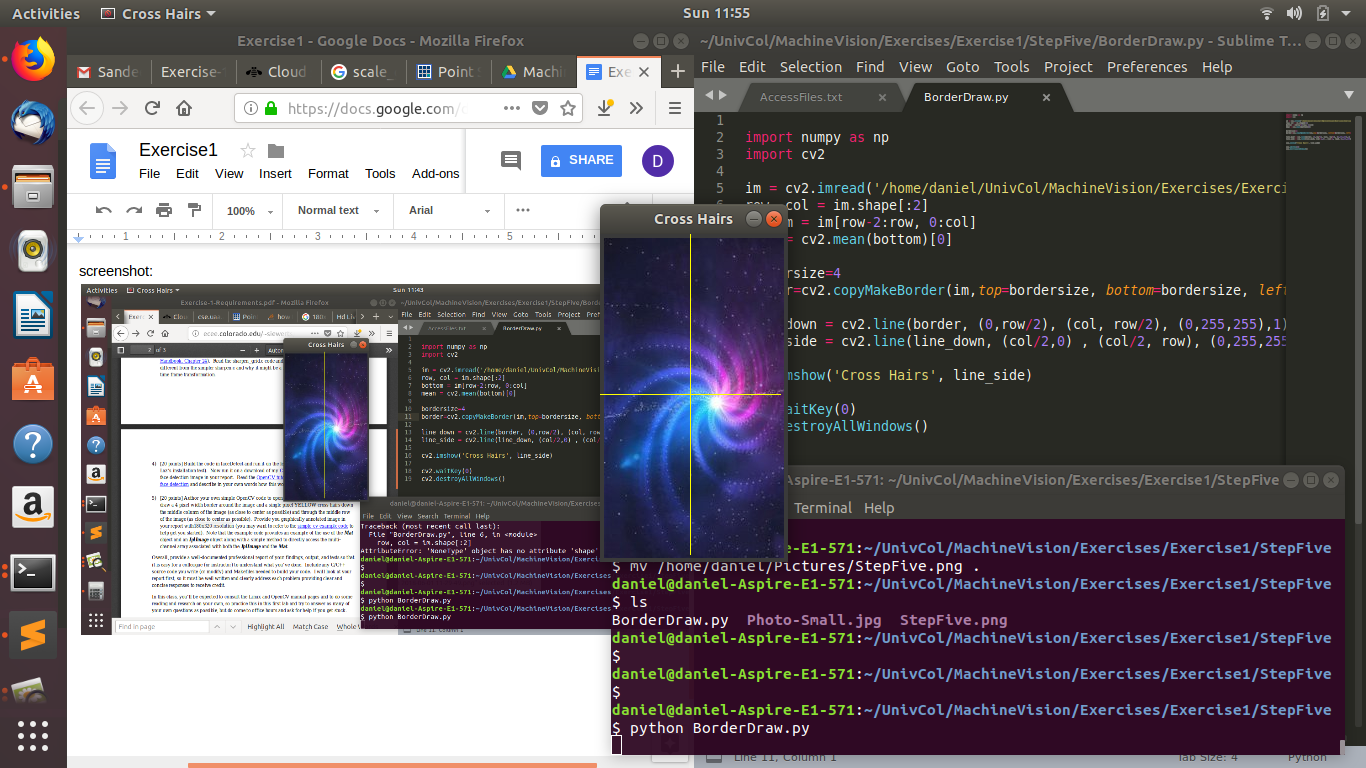
Exercise 1

Daniel Lindberg

06/01/2018

1. “Explore video analytics in the cloud “ by Sam B. Siewert discusses the field of “video analytics.” Which in turn is a broad definition that involves analysing digital video content. Analysation has many fields to it such as image acquisition , computer vision, machine vision...etc. With technology growing there are more ways to do video analytics, some involving processing of these set images in more powerful cloud infrastructure in addition to some of the mobile elements you can embed into these devices. Some of these cloud processes are able to use data-centric systems that can use intelligent inferencing and parse scenes at a much more rapid pace. There have been multiple real life applications of doing video analytics. Technologies such as augmented reality which help give the user video ques to better interact with the environment, automation in cars, and face detection to help depict certain feelings/emotions. Video analytics has been used in applications like augmented reality to help out with safety improvement, and almost for full operability in future autonomous vehicle designs. These augmented reality solutions and autonomous vehicles have the ability to detect certain scenarios and ques and react at a faster pace than the human eye can, examples such as today’s autopilot in cars making sure that you stay within the lines on the road.
2. Screenshot of the Sobel Trees run :  
     
   Description of the Sobel dataset:  
   The first thing that the sobel image processing application does is does a gaussian blur on the image. Gaussian blur is not strictly one algorithm but you have to specify the algorithm. These blurs are meant to be put into a place to reduce the image noise and reduce detail. The function that the gaussian blur does is reduces the size. Then it convers the image into from the Red Green Blue scale to Gray. It then at this point it calculates the derivatives in x and y directions, meaning it is going to detect the gradient edge detection. This is done by noticing the intensity changes in an image. Since it is in grayscale, it notices massive jumps in the equalized gray pixel number. It then applies the approximate gradient to the image from the Scharr and Sobel method to the already existing gaussian blur and gray-scaled image. Then it shows it.
3. Below is the output of the shaprened image on the right hand side of the normal image:  
     
   Obviously the sharpen\_grid.c code in its initial form was producing rows without the colorization. I believed it to be an incremental issue.So I doubled the amount of the H and W Sllices. My changes are in red below:  
     
   #define IMG\_H\_SLICE 2\*(IMG\_HEIGHT/NUM\_ROW\_THREADS)  
   #define IMG\_W\_SLICE 2\*(IMG\_WIDTH/NUM\_COL\_THREADS)  
     
   Through “The Scientist and Engineer’s Guide to Digital Signal Processing” by Steven W. Smith, he defines the point spread function. Steven Smith describes it as the first layer of the retina transforming an image represented as a pattern of light into an image presented as a pattern of nerve impulses. Then the second layer of the retina processes this neural image and passes it to the third layer, the fibers forming the optic nerve. Steven Smith describes specifically the point spread function as the function that the sensing material does to affect and spread out the point of light before it reaches the silicon.   
     
   The PSF function creates this optical illusion called mach bands. The mach bands as you see in the code below:   
   #define K 4.0  
   FLOAT PSF[9] = {-K/8.0, -K/8.0, -K/8.0, -K/8.0, K+1.0, -K/8.0, -K/8.0, -K/8.0, -K/8.0};  
   This PSF array are multipliers applied to each pixel square region. They almost act as a enhancer on the value of the pixels.nThese mach bands will attempt to detect vast changes in the brightness of certain parts of an image and then alter the image to have the dark region before the change appear darker and the lighter region before the light sections look brighter. Based off of the images you see above you can see that the bright color of the tree’s bark appears brighter in the right image, however it’s edges which go to a darker region will appear more dark. This is what is producing the sharpening effect.   
     
   There are a number of reasons why the Sharpen\_grid.c code may be better than the sharpen code. The sharpen\_grid.c code makes uses of threaded capabilities to generate a grid ppm (the output is written in R[i][j] , G[i][j], B[i][j] sequentially) . Whereas the other source code doesn’t use any pthreads, instead applied psf and generates a one dimensional output which is then written to the output ppm file.
4. Below is the screenshot of the face detect sample code I got form the OpenCV download 3.4.1. This is not the same one as the example-code in the exercise 1 step 2, that is because the one would not compile with the designated download of opencv that we were told to work with (3.4.1).   
   Lena:   
   Sam Siewert Faculty Picture: 

Now the code that is attached in the example-code files of Exercise 1 question 2. Uses opencv libraries and specifically uses a method called the Haar classifier. The haar classifier is a machine learning based approach. It attempts to extract Haar features from each image. Each window is placed on the picture to calculate features. Features to give an example are such as the region around the eyes in most face photos being darker than the area then the nose and cheeks. OpenCV has a number of functions to help out with the feature detection. They have detectMultiScale which is a general function to detect these features. It takes a scale factor which is a function that compensates for when a face is closer to the camera and appears bigger, the detectMultiscale also takes in the grayscale image. The detectMultiScale function returns the a list of positions, i.e. a rectangle in which the face detect will draw the selected figure. So the Haar method has an xml classifier set of images which helps classifies features on faces. These features are determined by matching certain shapes with darker regions and typically much lighter regions of the face.

1. Attached is a screenshot of the code I wrote. It is written in python. To run it all you need to do is run the attached code below and type “python BorderDraw.py”, you may have to change the physical path that I have within the code to match you directory. Here is the screenshot:  
     
   Python Code:
2. import numpy as np
3. import cv2
4. im = cv2.imread('Photo-Small.jpg')
5. row, col = im.shape[:2]
6. bottom = im[row-2:row, 0:col]
7. mean = cv2.mean(bottom)[0]
8. bordersize=4
9. border=cv2.copyMakeBorder(im,top=bordersize, bottom=bordersize, left=bordersize, right=bordersize, borderType=cv2.BORDER\_CONSTANT, value=[mean,mean,mean])
10. line\_down = cv2.line(border, (0,row/2), (col, row/2), (0,255,255),1)
11. line\_side = cv2.line(line\_down, (col/2,0) , (col/2, row), (0,255,255), 1)
12. cv2.imshow('Cross Hairs', line\_side)
13. cv2.waitKey(0)
14. cv2.destroyAllWindows()