Julianne Delia
Computer Vision
2/15/19

HW 2 Report

Problem 1:

1.1) To load the images, I used misc.imread.

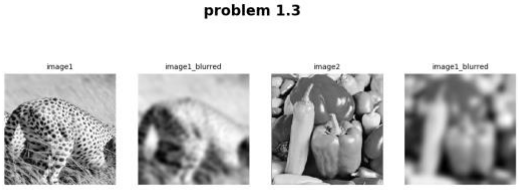
```
image1 = np.float64(misc.imread('cheetah.png', flatten=1, mode='F'))
image2 = np.float64(misc.imread('peppers.png', flatten=1, mode='F'))
```

1.2) To apply the Gaussian blur to each image, I used the code we learned in class called ndimage.gaussian_filter.

```
blur_im1 = ndimage.gaussian_filter(image1, 7)
blur_im2 = ndimage.gaussian_filter(image2, 7)
```

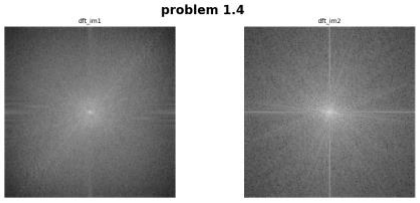
1.3) To display both images, I used the code that we learned in class that is a part of the matplot function, plt.show(). The plt.subplot arranges the plot, the title function gives the image a title, and the imshow function presents the image.

```
plt.figure(1, figsize=(15, 5))
plt.suptitle('problem 1.3', fontsize=20, fontweight='bold')

plt.subplot(1, 4, 1)
plt.title('image1', fontsize=10)
plt.imshow(image1, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(1, 4, 2)
plt.title('image1_blurred', fontsize=10)
plt.imshow(blur_im1, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(1, 4, 3)
plt.title('image2', fontsize=10)
plt.imshow(image2, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(1, 4, 4)
plt.title('image1_blurred', fontsize=10)
plt.imshow(blur_im2, cmap=plt.cm.gray)
plt.axis('off')
plt.show()
```



1.4) In order to compute the DFT of both images, I used a function called fft and fftshift. This is the numpy function for a discrete fourier transform (dft). I found this action on the docs.scipy.org website as well as looking through the lecture notes on how to use this. This function allows me to compute the DFT of each image.

```
dft_im1 = np.fft.fft2(image1)
dft_im1_shift = np.fft.fftshift(dft_im1)
mag_spect_im1 = np.log(np.abs(dft_im1_shift))

dft_im2 = np.fft.fft2(image2)
dft_im2_shift = np.fft.fftshift(dft_im2)
mag_spect_im2 = np.log(np.abs(dft_im2_shift))
```



1.5) Again, I used the same functions that were used in 1.4 in order to display the images. This is code that Professor Xiao showed us how to use.

```
plt.figure(1, figsize=(15, 5))
plt.suptitle('problem 1.4', fontsize=20, fontweight='bold')

plt.subplot(1, 2, 1)
plt.title('dft_im1', fontsize=10)
plt.imshow(mag_spect_im1, cmap=plt.cm.gray)
plt.axis('off')

plt.subplot(1, 2, 2)
plt.title('dft_im2', fontsize=10)
plt.imshow(mag_spect_im2, cmap=plt.cm.gray)
plt.axis('off')

plt.show()
```

Problem 2:

2.1) In order to load the color image, I used misc.imread, like I did in the first problem.

```
# convert color image to luminance
im3 = np.float64(misc.imread('lowcontrast.jpg', flatten=1))
```

2.2) In order to compute the histogram, I applied what we learned in class and used the numpy function numpy.histogram.

```
# Compute histogram
frequencies, bins = numpy.histogram(im3, bins=numpy.arange(-0.5, 255.1, 0.5))
intensities = bins[1:bins.size]
```

In order to find he cumulative distribution and CDF, I created a variable called "cdf" and used np.cumsum (cumulative sum) to find the cumulative distribution. Then, in order to transfer, I divided the cdf by np.float32(cdf[-1]) and multiplied it by 255.

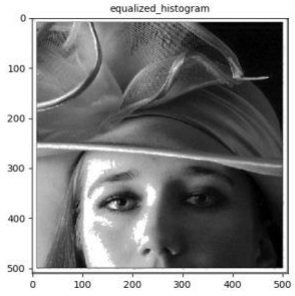
```
# CDF
cdf = np.cumsum(frequencies)

# transfer
intensities_mapping = cdf/np.float32(cdf[-1]) * 255
```

2.3) In order to map the number of pure black and white, I multiplied 0.025 (5%) by cdf([-1]). Then, I broke it down by finding the number of pure black and the number of pure white. I used a while loop for both and set the intensities map of black to 0 and of white to 255.

```
# find the number of pure black and white
number_pure_bw = cdf[-1] * 0.025

# find the number of pure black
num_of_blck = 0
i=0
while (num_of_blck < number_pure_bw):
    num_of_blck += frequencies[i]
    intensities_mapping[i] = 0
    i +=1

# find the number of pure white
num_of_white = 0
j = frequencies.size - 1
while (num_of_white < number_pure_bw):
    num_of_white += frequencies[j]
    intensities_mapping[j] = 255
    j -=1

count = im3.size - (num_of_blck + num_of_white)
```

2.4) Finally, in order to equalize the histogram, I used a while loop that contained a for loop. The while loop incorporated both the results of pure black and pure white in order to create an equalized histogram.

```
# equalize the histogram
tol = count * 0.025
trim = 0
while(trim < tol):
    trim = 0
    count = np.sum(frequencies[i:(j+1)])
    for x in range(i, j+1-i):
        ceiling = count/(j+1-i)
        if frequencies[x] > ceiling:
            trim += frequencies[x] - ceiling
            frequencies[x] = ceiling

intensities_mapping[i:(j+1)] = np.cumsum(frequencies[i:(j+1)]).astype(float)/count*255.0 + 255*(float(num_of_blck)/im3.size)
im3_he = np.interp(im3, intensities, intensities_mapping)

# plot the histogram
plt.figure()
plt.title('equalized_histogram', fontsize=10)
plt.imshow(im3_he, cmap=plt.cm.gray)
plt.show()
```



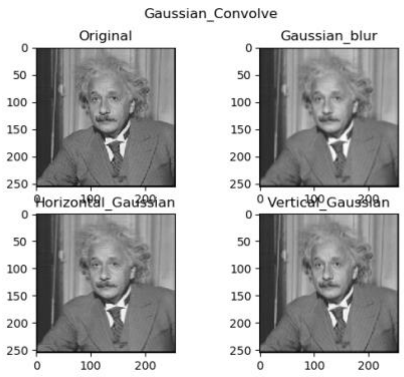
Problem 3:

- Gaussian blur
  - In order to apply the Gaussian kernel, I consulted the notes I took in class during the lecture and was able to find the correct numbers for the array that applies the Gaussian blur. Then, I used the filters.convolve command in order to convolve the image with the Gaussian kernel.

```
# gaussian blur filter
gaus_kern = 1.0/256*numpy.array([[1,4,6,4,1],[4,16,24,16,4],[6,24,36,24,6],[4,16,24,16,4],[1,4,6,4,1]])
gaus_kern_x = numpy.array([1,4,6,4,1])
gaus_kern_y = 1.0/256*numpy.array([1,4,6,4,1])

# gaussian convole
im4_gaus = filters.convolve(im4, gaus_kern, mode="mirror")
im4_gaus_x = filters.convolve1d(im4, gaus_kern_x, axis=1, mode = "mirror")
im4_gaus_y = filters.convolve1d(im4, gaus_kern_y, axis=0, mode = "mirror")
```

  - In order to plot the different outcomes of the Gaussian blur, I again used the matplot function of set_title and imshow. I converted the images to gray scale as well when plotting them.

```
# plot gaussian
fig4_gaussian = plt.figure()
fig4_gaussian.suptitle("Gaussian_Convolve")

im4_plot = fig4_gaussian.add_subplot(2,2,1)
im4_plot.set_title("Original")
im4_plot.imshow(im4, cmap=plt.cm.gray)

# im4_gaus
im4_gaussian_plot = fig4_gaussian.add_subplot(2,2,2)
im4_gaussian_plot.set_title("Gaussian_blur")
im4_gaussian_plot.imshow(im4_gaus, cmap=plt.cm.gray)

# im4_gaus_x
im4_gaussian_x_plot = fig4_gaussian.add_subplot(2,2,3)
im4_gaussian_x_plot.set_title("Horizontal_Gaussian")
im4_gaussian_x_plot.imshow(im4_gaus_x, cmap=plt.cm.gray)

# im4_gaus_y
im4_gaussian_y_plot = fig4_gaussian.add_subplot(2,2,4)
im4_gaussian_y_plot.set_title("Vertical_Gaussian")
im4_gaussian_y_plot.imshow(im4_gaus_y, cmap=plt.cm.gray)
```

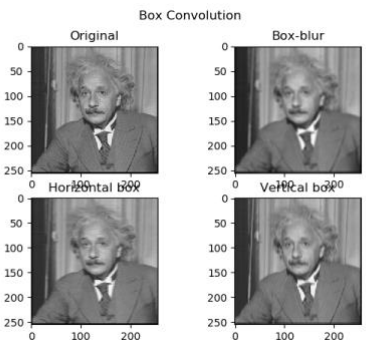


- Box filter
  - Once I completed the Gaussian kernel, I was able to apply what I learned and what I found on scipy in order to apply the box filter to the image. Again, I consulted the code from class in order to convolve the image.

```
# box filter
box_kern = numpy.ones((5,5), dtype=numpy.float32)/25
box_kern_x = numpy.array([1,1,1,1,1])
box_kern_y = numpy.array([1,1,1,1,1])/25.0

# box convolve
im4_box = filters.convolve(im4, box_kern, mode="wrap")
im4_box_x = filters.convolve1d(im4, box_kern_x, axis=1, mode="wrap")
im4_box_y = filters.convolve1d(im4, box_kern_y, axis=0, mode="wrap")
```

  - In order to plot the images, I used the same functions for the box as I did for the Gaussian blur. Once I was able to figure out the first one, the box and sobel filter had similar commands.

```
# plot box
fig4_box = plt.figure()
fig4_box.suptitle("Box Convolution")

im4_plot = fig4_box.add_subplot(2,2,1)
im4_plot.set_title("Original")
im4_plot.imshow(im4, cmap=plt.cm.gray)

# im4_box
im4_box_plot = fig4_box.add_subplot(2,2,2)
im4_box_plot.set_title("Box-blur")
im4_box_plot.imshow(im4_box, cmap=plt.cm.gray)

# im4_box_x
im4_box_x_plot = fig4_box.add_subplot(2,2,3)
im4_box_x_plot.set_title("Horizontal box")
im4_box_x_plot.imshow(im4_box_x, cmap=plt.cm.gray)

# im4_box_y
im4_box_y_plot = fig4_box.add_subplot(2,2,4)
im4_box_y_plot.set_title("Vertical box")
im4_box_y_plot.imshow(im4_box_y, cmap=plt.cm.gray)
```

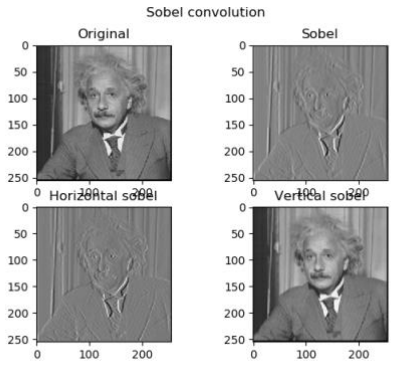


- Sobel filter
    - It took me slightly longer to figure this one out, but once Professor Xiao covered the sobel filter during this week's lecture, I was able to complete this part of the problem.

```
# sobel filter
sobel_kern = numpy.array([[1,2,0,-2,-1],[4,8,0,-8,-4],[6,12,0,-12,-6],[4,8,0,-8,-4],[1,2,0,-2,-1]])
sobel_kern_x = numpy.array([1,2,0,-2,-1])
sobel_kern_y = numpy.array([1,4,6,4,1])

# sobel convolve
im4_sobel = filters.convolve(im4, sobel_kern, mode="nearest")
im4_sobel_x = filters.convolve1d(im4, sobel_kern_x, axis=1, mode="nearest")
im4_sobel_y = filters.convolve1d(im4, sobel_kern_y, axis=0, mode="nearest")
```

    - Once again, I used the same code in order to plot the images. Plotting the images was probably the least difficult part of the assignment because once I knew how to plot one thing, the all came naturally because the code is very similar.

```
# plot sobel
fig4_sobel = plt.figure()
fig4_sobel.suptitle("Sobel convolution")

im4_plot = fig4_sobel.add_subplot(2,2,1)
im4_plot.set_title("Original")
im4_plot.imshow(im4, cmap=plt.cm.gray)

# im4_sobel
im4_sobel_plot = fig4_sobel.add_subplot(2,2,2)
im4_sobel_plot.set_title("Sobel")
im4_sobel_plot.imshow(im4_sobel, cmap=plt.cm.gray)

# im4_sobel_x
im4_sobel_x_plot = fig4_sobel.add_subplot(2,2,3)
im4_sobel_x_plot.set_title("Horizontal sobel")
im4_sobel_x_plot.imshow(im4_sobel_x, cmap=plt.cm.gray)

# im4_sobel_y
im4_sobel_y_plot = fig4_sobel.add_subplot(2,2,4)
im4_sobel_y_plot.set_title("Vertical sobel")
im4_sobel_y_plot.imshow(im4_sobel_y, cmap=plt.cm.gray)

plt.show()
```



Problem 4:

For this problem, I used ndimage.sobel in order to compute the edges of the images. Then I used np.hypot to compute the magnitude. I was stuck on this part for a while until this week's lecture when professor xiao suggested we use hypot. I wasn't sure why my code didn't work for a few days and it was really frustrating. Then, once we went over all of this in the lecture, it all made sense and I was able to figure it out very quickly. After figuring out the magnitude, the rest followed suit easily. All I had to do was plot it. Plotting images in numpy has become a lot easier because we are doing it a lot both in the homework and in class. I think the repetition has been very helpful.