Introduction to Digital Image Processing

Homework 6

December 6, 2020

Student: Ivan Arevalo
Perm Number: 5613567
Email: ifa@ucsb.com

Department of Electrical and Computer Engineering, UCSB

Problem 1

Figure 1: Separate edge enhancement on the R, G, and B channels

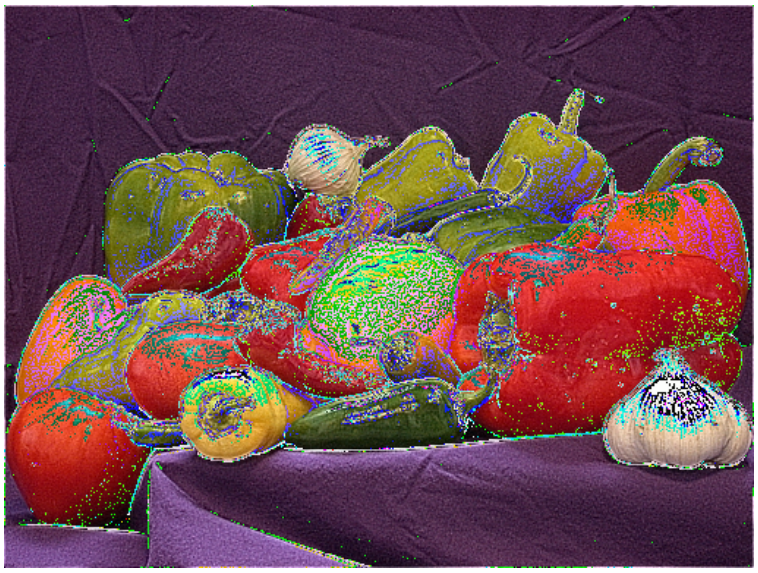

Figure 2: Edge enhancement to the Y channel

We can see that edge enhancement on Y channel distorted the colors less as in the reds are redder and greens are greener, but I suspect I made some mistake in my code that I was't to figure out as imagine the edge enhancement on Y channel should look way better. I payed close attention to data types and run multiple test in order to debug but unfortunetwly couldnt find the root of the problem. I made sure my RGB to YIQ and YIQ to RGB transforms were correct by comparing the original with one obtained by passing it through both transforms and it looked exactly the same. I also looked at the channel-wise edge enhancements and they looked correct. I have attached as reference.

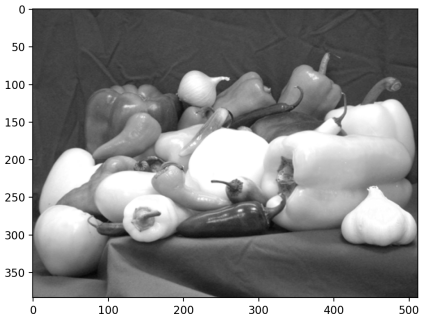

Figure 3: Original red channel

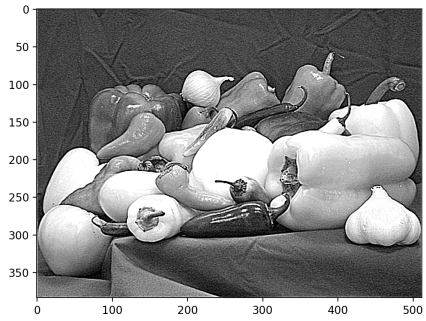

Figure 4: Enhanced red channel

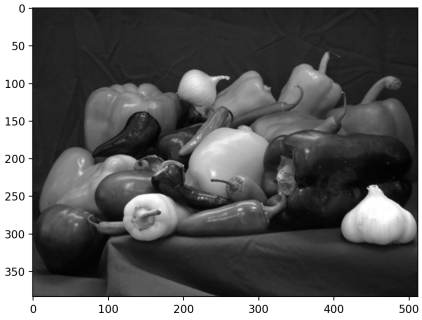

Figure 5: Original green channel

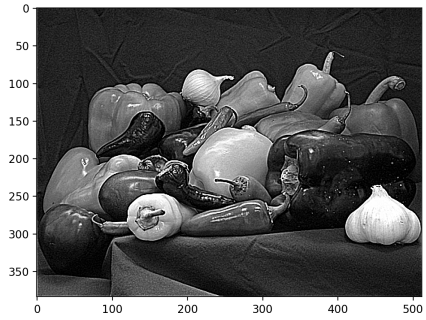

Figure 6: Enhanced green channel

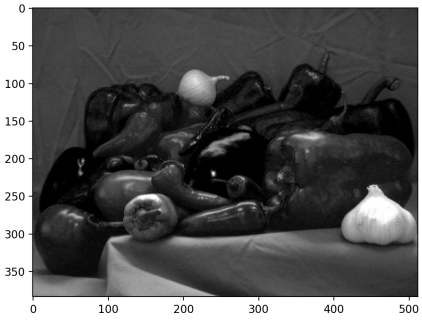

Figure 7: Original blue channel

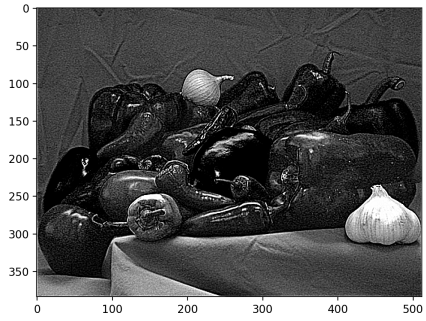

Figure 8: Enhanced blue channel

Problem 2

Figure 9: Output image with 4 pixel approximation

Figure 10: Output image with 8 pixel approximation

The mean square error for the 4 and 8 pixel approximation was 6.35 and 2.21 respectively. We can see that having more pixels in our approximation slightly improved our image and I hypothesis that it can further improve the more pixels we pick for out estimation. Our affine estimations for 4 and 8 pixels estimation are displayed below.

```
[[  1.59955357  -0.52165179  -0.05714286 -42.31205357]
 [ -0.30967262   1.71421131  -0.1047619  -44.86116071]
 [ -0.29449405   0.15037202   1.43809524 -43.45133929]]
```

Figure 11: Linear transform with 4 pixel approximation

```
[[  1.66523185  -0.53458339  -0.10620867 -43.05793825]
 [ -0.33312678   1.67052091  -0.04982558 -42.86881042]
 [ -0.26994947   0.15068173   1.41369862 -43.50271913]]
```

Figure 12: Linear transform with 8 pixel approximation

Edge Enhancement Code

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
from scipy.signal import convolve2d

alpha = 1
impulse = np.array([[0,0,0],[0,1,0],[0,0,0]])
laplace = np.array([[1,1,1],[1,-8,1],[1,1,1]])
sharp_filter = impulse - alpha*laplace #considering diagonal and alpha =1
print(sharp_filter)
img = Image.open("peppers_color.png")
plt.imshow(img, cmap='gray', vmin=0, vmax=255)
plt.show()
img_pxl = np.asarray(img).astype(np.uint8)
output_img = np.zeros(img_pxl.shape)


for i in range(img_pxl.shape[2]):
    plt.imshow(img_pxl[:, :, i], cmap='gray', vmin=0, vmax=255)
    plt.show()
    output_img[:, :, i] = convolve2d(img_pxl[:, :, i], sharp_filter, mode='same')
    plt.imshow(output_img[:, :, i], cmap='gray', vmin=0, vmax=255)
    plt.show()

plt.imshow(output_img.astype(np.uint8), vmin=0, vmax=255)
plt.show()
plt.imsave('peppers enhanced RGB.png', np.round(output_img).astype(np.uint8), vmin=0,
    vmax=255)

YIQ_transform = np.array([[0.299, 0.587, 0.114], [0.596, -0.275, -0.321], [0.212,
    -0.526, 0.311]])
print(YIQ_transform)
yiq_img = np.zeros(img_pxl.shape)

for j in range(img_pxl.shape[0]):
    for i in range(img_pxl.shape[1]):
        yiq_img[j,i,:] = YIQ_transform@img_pxl[j,i,:]

print("Displaying raw YIQ")
plt.imshow(yiq_img.astype(np.uint8), vmin=0, vmax=255)
plt.show()
```

```
yiq_eq = np.copy(yiq_img)
yiq_trial = np.copy(yiq_img)
yiq_eq[:,:,0] = convolve2d(yiq_img[:, :, 0], sharp_filter, mode='same')

print("Displaying Y after filter")
print(yiq_eq.shape)
plt.imshow(yiq_eq[:,:,0].astype(np.uint8), cmap='gray', vmin=0, vmax=255)
plt.show()

RGB_transform = np.array([[1.000, 0.956, 0.602], [1.000, -0.272, -0.647], [1.000,
    -1.108, 1.700]])
print(RGB_transform)
rgb_img = np.zeros(img_pxl.shape)

for j in range(img_pxl.shape[0]):
    for i in range(img_pxl.shape[1]):
        rgb_img[j,i,:] = RGB_transform@yiq_eq[j,i,:]
        yiq_trial[j,i,:] = RGB_transform@yiq_img[j,i,:]

print("Displaying RGB image")
plt.imshow(rgb_img.astype(np.uint8))
plt.imsave('peppers enhanced Y.png', rgb_img.astype(np.uint8))
plt.show()
```

Color Calibration Code

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

def compute_lin_transform(input_img, out_img, num_pxls):
    selected_pxls = []
    M = np.ones([num_pxls, 4])
    t = np.zeros([num_pxls, 3])
    iter = 0
    while len(selected_pxls) < num_pxls:
        j = np.random.randint(0, input_img.shape[0])
        i = np.random.randint(0, input_img.shape[1])
        if (j,i) not in selected_pxls:
            selected_pxls.append((j,i))
            M[iter, :3] = input_img[j, i, :]
            t[iter] = out_img[j, i, :]
            iter +=1

    A = np.zeros([3, 4])
    for i in range(t.shape[1]):
        a =  np.linalg.lstsq(M, t[:,i], rcond=None)[0]
        A[i, :] = a
    print(A)

    return A
```

```
def perform_lin_transform(input, A):
    col_img = np.zeros(out_img_pxl.shape)
    for j in range(in_img_pxl.shape[0]):
        for i in range(in_img_pxl.shape[1]):
            x = np.concatenate((in_img_pxl[j, i, :], np.ones(1)))
            col_img[j, i, :] = A @ x
    return col_img

in_img = Image.open("q2_orig.png")
out_img = Image.open("q2_color_txm.png")
in_img_pxl = np.asarray(in_img)
out_img_pxl = np.asarray(out_img)

A4 = compute_lin_transform(in_img_pxl, out_img_pxl, 4)
col_img4 = perform_lin_transform(in_img_pxl, A4)
print("Displaying Color Calibrated Image")
plt.imshow(col_img4.astype(np.uint8), vmin=0, vmax=255)
plt.show()
# plt.imsave('q2_calibrated_4p.png', col_img4.astype(np.uint8))
MSE4 = ((out_img_pxl - col_img4)**2).mean(axis=None)
print("MSE with 4 pixels: {}".format(MSE4))

A8 = compute_lin_transform(in_img_pxl, out_img_pxl, 8)
col_img8 = perform_lin_transform(in_img_pxl, A8)
print("Displaying Color Calibrated Image")
plt.imshow(col_img8.astype(np.uint8), vmin=0, vmax=255)
plt.show()
# plt.imsave('q2_calibrated_8p.png', col_img8.astype(np.uint8))
MSE8 = ((out_img_pxl - col_img8)**2).mean(axis=None)
print("MSE with 8 pixels: {}".format(MSE8))
```