

Multimedia Assignment -E1

Muhammad R. Pourbaba

40214160280404

Professor: Dr. Bastam

Topic: JPEG encoding

1.RGBtoYIQ: A simple color space conversion on a small 8x8 block of Uncompressed portable pixel map(PPM) Gives us YCbCr matrix which is depicted in **Figure 1**:

```
home > mamzi > proj > dct.py ...
1 from tkinter import Tk
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
TypeError: a bytes-like object is required, not
mamzi@Karen:~/proj$ /home/linuxbrew/.linuxbrew
Y component (luminance):
[[150 150 165 165 165 165 165 150]
 [150 180 195 195 210 195 195 165]
 [165 116 90 191 240 225 210 180]
 [165 116 90 213 255 255 225 195]
 [165 210 240 255 255 255 240 195]
 [165 195 225 255 255 254 222 193]
 [165 195 210 225 240 210 90 100]
 [150 165 180 195 195 184 92 99]]
Cb component (blue-difference chrominance):
[[ 42 42 55 55 55 55 55 42]
 [ 42 67 79 79 91 79 79 55]
 [ 55 77 76 76 116 103 91 67]
 [ 55 77 76 115 128 128 103 79]
 [ 55 91 116 128 128 128 116 79]
 [ 55 79 103 128 128 127 103 79]
 [ 55 79 91 103 116 100 76 75]
 [ 42 55 67 79 79 78 76 74]]
Cr component (red-difference chrominance):
[[ 23 23 37 37 37 37 37 23]
 [ 23 52 67 67 82 67 67 37]
 [ 37 201 245 134 112 97 82 52]
 [ 37 201 245 157 128 128 97 67]
 [ 37 82 112 128 128 128 112 67]
 [ 37 67 97 128 128 128 100 70]
 [ 37 67 82 97 112 114 245 224]
 [ 23 37 52 67 67 87 242 220]]
/home/mamzi/proj/import numpy as np.py:49: Use
not be shown
```

Figure 1

```
[250 32 41]]
/home/mamzi/proj/from PIL import Image.py:13: ComplexWarning: Casting com
return np.round(np.fft.fft2(block - 128), decimals=0).astype(int)
DCT Y (Luminance):
[[ 3799 -1122 256 50 -115 50 256 -1122]
 [ -912 877 187 -143 74 174 -144 -396]
 [ 238 43 -282 -38 31 -50 -285 225]
 [ -160 -118 13 53 -15 2 123 149]
 [ -80 60 13 -1 53 -1 13 60]
 [ -160 149 123 2 -15 53 13 -118]
 [ 238 225 -285 -50 31 -38 -282 43]
 [ -912 -396 -144 174 74 -143 187 877]]
DCT I (In-phase Chrominance):
[[ -8998 -275 -843 -426 -83 -426 -843 -275]
 [ -606 -867 -425 124 50 -168 467 1252]
 [ -856 -188 255 202 22 199 387 -93]
 [ -94 142 98 23 -11 77 -20 -185]
 [ -58 40 10 -2 33 -2 10 40]
 [ -94 -185 -20 77 -11 23 98 142]
 [ -856 -93 387 199 22 202 255 -188]
 [ -606 1252 467 -168 50 124 -425 -867]]
DCT Q (Quadrature-phase Chrominance):
[[ -11690 -972 -674 -391 -151 -391 -674 -972]
 [ -1172 -313 -306 31 95 -58 370 987]
 [ -698 -159 75 178 42 165 201 44]
 [ -191 64 106 59 -20 79 57 -91]
 [ -104 75 18 0 70 0 18 75]
 [ -191 -91 57 79 -20 59 106 64]
 [ -698 44 201 165 42 178 75 -159]
 [ -1172 987 370 -58 95 31 -306 -313]]
mamzi@Karen:~/proj$
```

Figure 2

2.DCT: After color space conversion i've done the Discrete Cosine Transform(DCT) on the same image YIQ matrix **Figure 2** , **Figure 3**:

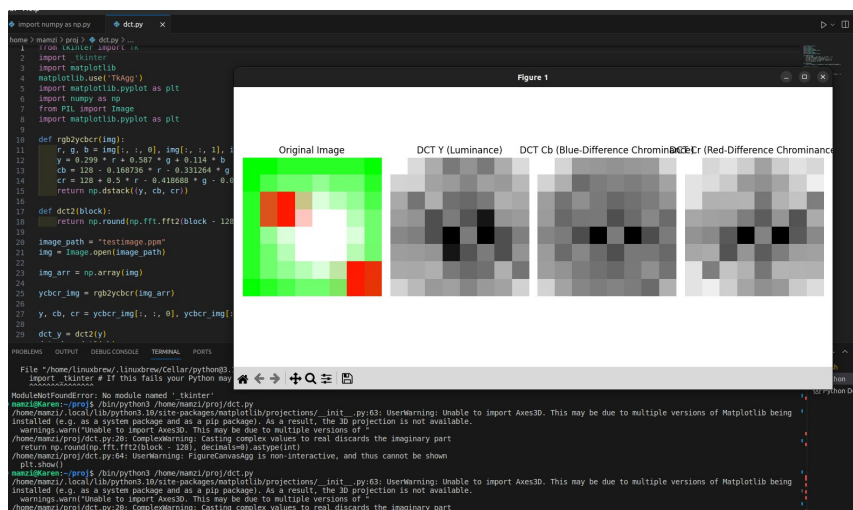


Figure 3

3.Quantization: now we can divide values of arrays to the arbitrary quantization rate (Figure 4)

```

45
46 q_table_iq = np.array([[17, 18, 24, 47, 99, 99, 99, 99],
47                        [18, 21, 26, 66, 99, 99, 99, 99],
48                        [24, 26, 56, 99, 99, 99, 99, 99],
49                        [47, 66, 99, 99, 99, 99, 99, 99],
50                        [99, 99, 99, 99, 99, 99, 99, 99],
51                        [99, 99, 99, 99, 99, 99, 99, 99],
52                        [99, 99, 99, 99, 99, 99, 99, 99],
53                        [99, 99, 99, 99, 99, 99, 99, 99]])
54
55 # Quantize DCT coefficients

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```

/home/mamzi/proj/from PIL import Image.py:13: ComplexWarning: Casting complex numbers to real discards the imaginary part
  return np.round(np.fft.fft2(block - 128), decimals=0).astype(int)
Quantized DCT Y (Luminance):
[[ 237 -102   26    3   -5    1    5  -18]
 [ -76   73   13   -8    3    3   -2  -7]
 [  17    3  -18   -2    1   -1   -4   4]
 [ -11   -7    1    2    0    0    2    2]
 [  -4    3    0    0    1    0    0    1]
 [  -7    4    2    0    0    1    0  -1]
 [   5    4   -4   -1    0    0   -2    0]
 [ -13   -4   -2    2    1   -1    2    9]]
Quantized DCT I (In-phase Chrominance):
[[ -529  -15  -35   -9   -1   -4   -9   -3]
 [ -34  -41  -16    2    1   -2    5   13]
 [ -36   -7    5    2    0    2    4   -1]
 [  -2    2    1    0    0    1    0  -2]
 [  -1    0    0    0    0    0    0    0]
 [  -1   -2    0    1    0    0    1    1]
 [  -9   -1    4    2    0    2    3  -2]
 [  -6   13    5   -2    1    1   -4  -9]]
Quantized DCT Q (Quadrature-phase Chrominance):
[[ -688  -54  -28   -8   -2   -4   -7  -10]
 [ -65  -15  -12    0    1   -1    4   10]
 [ -29  -6    1    2    0    2    2    0]
 [  -4    1    1    1    0    1    1   -1]
 [  -1    1    0    0    1    0    0    1]
 [  -2   -1    1    1    0    1    1    1]
 [  -7    0    2    2    0    2    1  -2]
 [ -12   10    4   -1    1    0   -3  -3]]
mamzi@Karen:~/proj$

```

Figure 4

4. Calculating Huffman table: by making the Huffman tree we are able to generate the Huffman table

5.Run-Length Encoding (RLE): with encoding the arrays using Huffman Table we are actually generating the coefficients which can be send over network(Figure 5).

```
110001100110011100011000110001100011000110001100011000110001010  
● mamzi@Karen:~/proj$ /bin/python3 "/home/mamzi/proj/from PIL import Image.py"  
/home/mamzi/proj/from PIL import Image.py:45: ComplexWarning: Casting complex value to real discards imaginary part  
return np.round(np.fft.fftfreq(block // 128), decimals=0).astype(int)  
Huffman-encoded RLE DCT Y (Luminance):  
11100110001110011100111001100011001110011100110011100111000110011010  
Huffman-encoded RLE DCT I (In-phase Chrominance):  
110011100011001110011100011000110011110011000110001010  
Huffman-encoded RLE DCT Q (Quadrature-phase Chrominance):  
110001100111001110001100011000110001100011000110001100011000110011100111000110001010  
○ mamzi@Karen:~/proj$
```

Figure 5

6.sending over network: so i did setup a virtual machine act as a server to receive the code and decode it . The final comparison between the Original image versus the Processed image is shown in **Figure 6**.

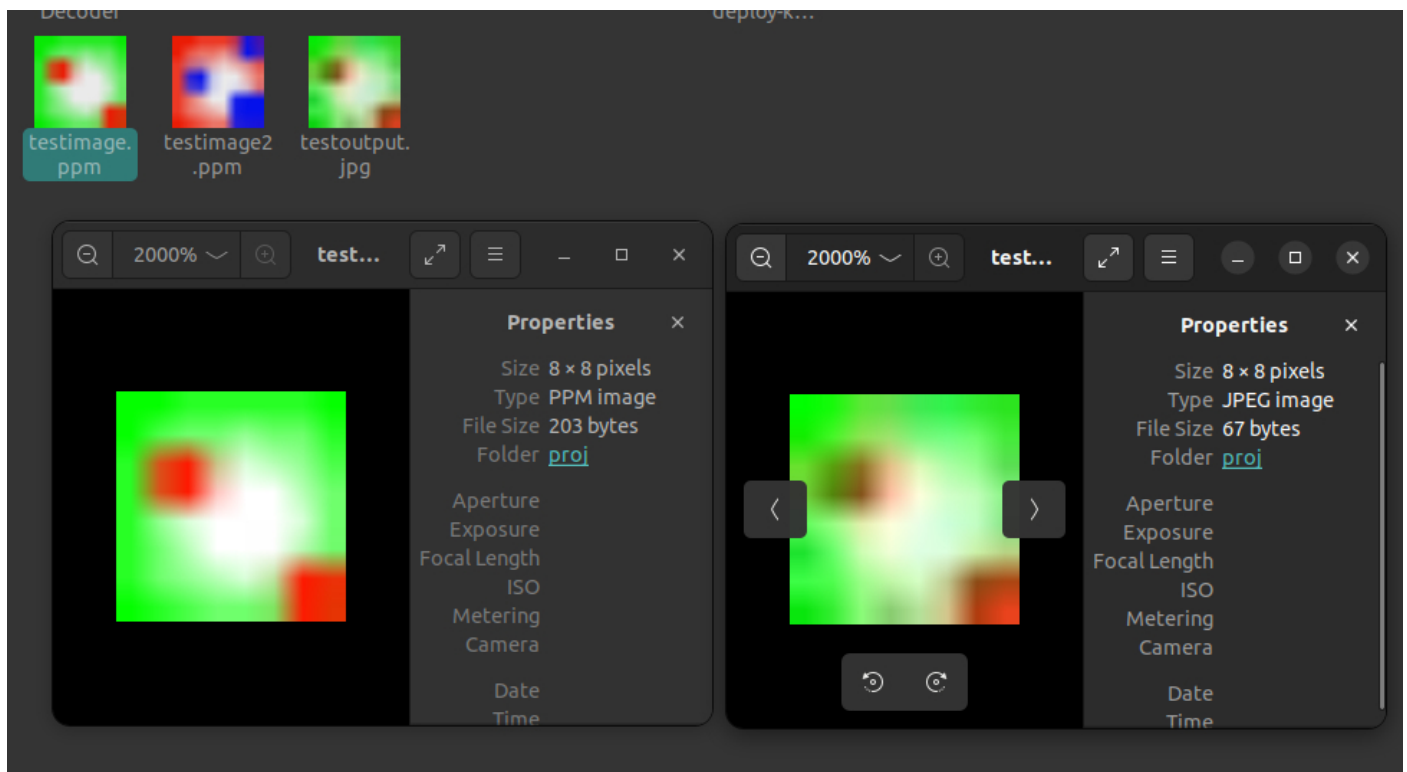


Figure 6

Since the JPEG is a lossy-image-compression algorithm the result of the jpeg process on the image is sensible on the sharpness of the encoded image. It is obvious that the encoded image is less sharper and a bit different from the Original image.

Project files: https://github.com/mopoa/jpeg_compression.git