Image Process HW3

多工所 碩一 傅子容 310553030

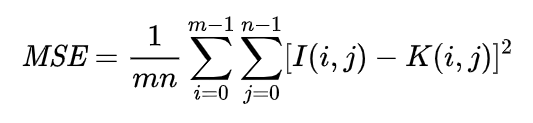
1. **Introduction**

JPEG developed by Joint Photographic Experts Group is widely used in image store. It is published at 1992 and it is a lossy method which means that it’ll sacrifice some trivial information to save image in fewer data. There are many components in JPEG structure to help user achieving this effect, such as different coding way, doing quantization.

In this assignment, I refer to the code on Github and modify it to conduct experiments on some components to show their contributions in the process.

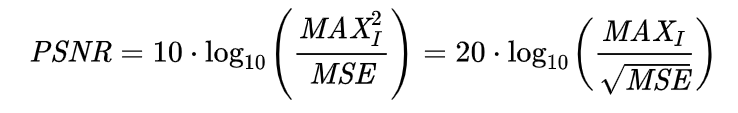
1. J**PEG Encoder/Decoder Method**
   1. Encoder
      1. Convert color space from RGB into YCbCr.
      2. Split image into blocks.
      3. Minus 128 for each element to change data range from [0,255] to [-128,127].
      4. Do DCT on blocks.
      5. Quantize each block with Quantize Tables luminance and chrominance.
      6. Use zigzag represent to arrange each element in the blocks.
      7. Do predictive encode for DC term and run-length encode for AC terms.
      8. Encode with Huffman coding.
      9. Pack all tables, DC term and AC terms.
   2. Decoder
      1. Select tables, DC term and AC terms.
      2. Use Huffman table to decode.
      3. Reverse zigzag back to original represent.
      4. Multiple blocks and Quantize Tables
      5. Do IDCT on each block
      6. Change data range back to [-128,127]
      7. Convert color space from YCbCr back to RGB.
2. **Evaluate**
   1. MSE, Mean-Square Error

MSE shows the difference of each pixel between compression image and original image. It calculates the deviation of decoded compression image. The function is below: I is original and K is after compression.



* 1. PSNR, Peak Signal-to-Noise Ratio

PSNR shows the quality of compression image comparing with the original image. Higher PSNR value means higher quality and fewer deviation. The calculate function is below, and MAX is 255 for color image.



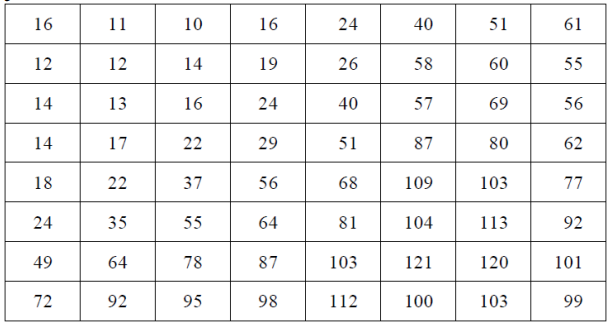
1. **Preprocess**

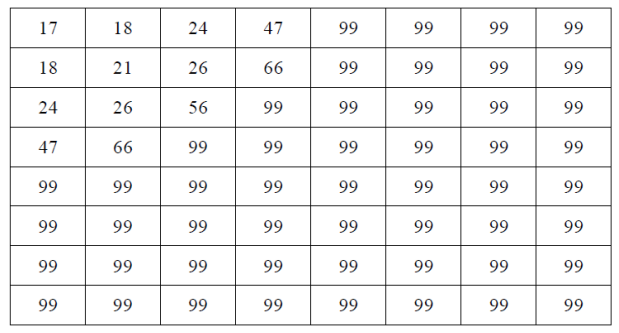
When split image into blocks, it will cause error if image cannot divide into integer. For this situation, I implement zero-padding as preprocess. Using image with zero-padding to encode and remove zero pixels after decoding.

1. **Experiment**
   1. Different Quantize Tables
      1. Different composition

First compare 2 pairs of Quantize Tables: one is supplied by professor and another is found on network. Follows are this two pairs of quantize tables for luminance (up) and chrominance(bottom).

Quantize table 1, supplied by professor





Quantize table 2, found on network

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 |
| 2 | 2 | 2 | 2 | 3 | 4 | 5 | 6 |
| 2 | 2 | 2 | 2 | 4 | 5 | 7 | 9 |
| 2 | 2 | 2 | 4 | 5 | 7 | 9 | 12 |
| 3 | 3 | 4 | 5 | 8 | 10 | 12 | 12 |
| 4 | 4 | 5 | 7 | 10 | 12 | 12 | 12 |
| 5 | 5 | 7 | 9 | 12 | 12 | 12 | 12 |
| 6 | 6 | 9 | 12 | 12 | 12 | 12 | 12 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 3 | 5 | 9 | 13 | 15 | 15 | 15 |
| 3 | 4 | 6 | 11 | 14 | 12 | 12 | 12 |
| 5 | 6 | 9 | 14 | 12 | 12 | 12 | 12 |
| 9 | 11 | 14 | 12 | 12 | 12 | 12 | 12 |
| 13 | 14 | 12 | 12 | 12 | 12 | 12 | 12 |
| 15 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 15 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 15 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |

As the result, we can see how different quantize table affect compression output. Overall, both create a good compression image which very similar to the source image. However, quantize table 1 makes more noises than table 2 when zoom in detail, also uses more space to save image.



1. Original image



1. Using table 2

PSNR=131.07829

MSE=0.13195



1. Using table 1

PSNR=130.98140

MSE=0.13324



The most obvious difference between two table is their element range. To observe how it affects result, I conduct experiment 2, using different scale quantize table.

* + 1. Different scale

I multiple quantize table 2 by 5 different times (0.1,0.5,1,5,10). The results show that the quality is related to the element range of quantize table.

Bigger number in quantize table, make lower quality of compression image. When the elements lower enough, the results will look similar. If the number keep lower, the quality will start ruin again instead of keeping increase.



Table 2,   
x0.1, x0.5  
x1, x5, x10



Table 1,   
x0.1, x0.5  
x1, x5, x10

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table 1 | | | Table 2 | | |
| times | luminance | chrominance | times | luminance | chrominance |
| 0.1 | [1,12] | [1.7,9.9] | 0.1 | [0.2,1.2] | [0.3,1.5] |
| 0.5 | [5,60.5] | [8.5,49.5] | 0.5 | [1,6] | [1.5,7.5] |
| 1 | [10,121] | [17,99] | 1 | [2,12] | [3,15] |
| 5 | [50,605] | [85,495] | 5 | [10,60] | [15,75] |
| 10 | [100,1210] | [170,990] | 10 | [20,120] | [30,150] |

* 1. Different block size

I conduct experiments with different block size: 4,8,16,24(from up to bottom, first is original image), and test different interpolation way to expand quantize table.

Bigger block size brings out more loss and does not such influence consume space, only a bit of fewer.







1. 4x4 block

MSE=0.3145

PSNR= 122.3934



1. 8x8 block

MSE=0.3198

PSNR=122.2230



1. 16x16 block

MSE=0.3242

PSNR=122.0893



1. 24x24 block

MSE=0.3254

PSNR=122.0518

Using cubic interpolation will generate color noise in image and noise increase with block size. On the other hand, linear interpolation and quintic interpolation do not result in this effect.

Cubic interpolation also spent more space.



1. Using cubic interpolation

MSE= 0.3180

PSNR= 122.28137



1. Using linear interpolation

MSE= 0.3254

PSNR= 122.0518



1. Using quintic interpolation

MSE= 0.31180

PSNR= 122.47917

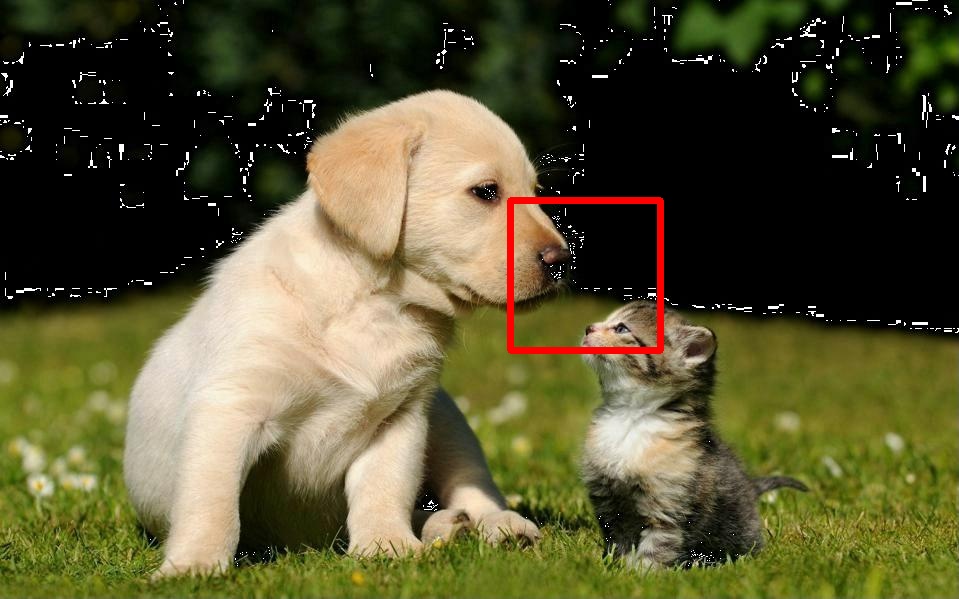


* 1. Whether using predictive coding in DC term

This experiment is trying to prove predictive coding can truly help to save space. These txt file record the code encoded by encoder and jpg file are image recover from code.

It shows that predictive coding can makes code smaller and does not affect the decoding image.

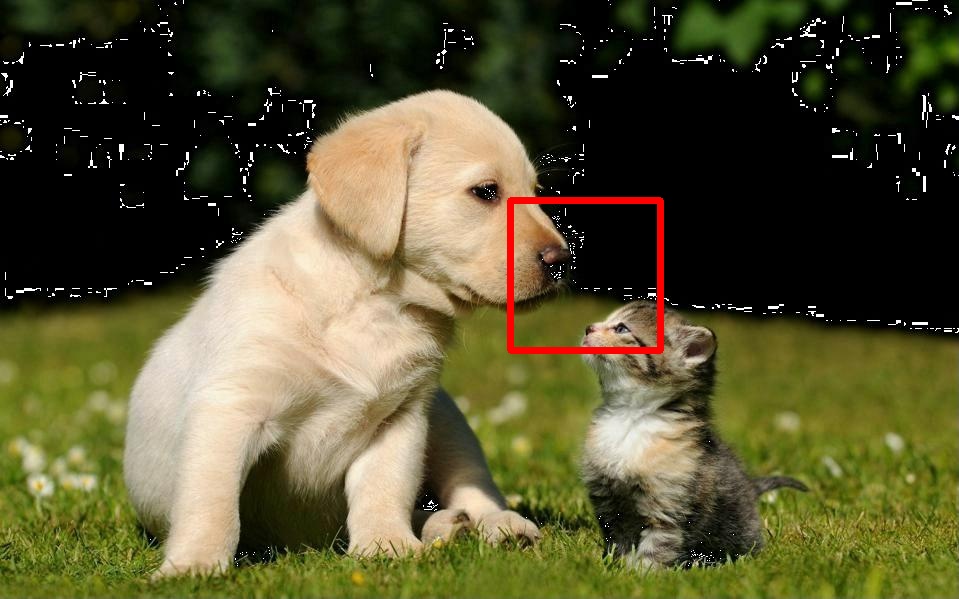




1. With predictive coding

MSE= 0.21476

PSNR= 126.20774



1. W/O predictive coding

MSE= 0.21476

PSNR= 126.20774

1. **Reference**

Github code: <https://github.com/ghallak/jpeg-python>

[Wiki-PSNR](https://zh.wikipedia.org/wiki/%E5%B3%B0%E5%80%BC%E4%BF%A1%E5%99%AA%E6%AF%94)

1. **Code**