Work of JPEG and ZIP and compression test of ZIP

DP1-1 Eric

Contents

1 Preliminary

1.1 Introduction to BitMap Image(BMP)

A image format which storage every pixel's color directly. Coommonly we use one byte per color so 24 bits per pixel.

1.2 Introduction to Joint Photographic Experts Group Image(JPG)

JPEG(.jpg or .jpeg) is an image format developed by Joint Photographic Experts Group. It can efficiently compress the image thus get a relative high quality image as the compress rate is high. This compression is lossy compression. [1]

1.3 Introduction to Huffman Coding

Huffman Coding is a compression method which use shorter binary to represent character with higher frequency. In this code method, a element's code can't be another element's code's prefix. Otherwise, this will lead to not only one decode after encode. This method works in the following way:

Calculate frequency and generate binary tree.

Generate dictionary according to the binary tree.

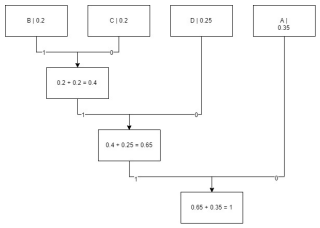
Use the dictionary to encode / decode the string.

[6]
Let's take string $ABDCBAADDAADBCCCBDAA$ as an example.
Calculate frequency

Character	Times	Frequency
A	7	0.35
B	4	0.2
C	4	0.2
D	5	0.25

Table 1: Frequency

Build binary tree



tree.jpg

Figure 1: Binary tree

Generate dictionary

Character	Code
B	111
C	110
D	10
A	0

Table 2: Dictionary

Code the string

$ABDCBAADDAADBCCCBDAA$

$\rightarrow$ 01111011011100101000101111101101101111000

2 Reason why .bmp file is always larger than .jpg file

2.1 How JPG works

JPG Image is formed in the following way:

Devide Image: Devide the image into many small parts with resolution of 8×8 pixels.

Colorspace Conversion: For every 8×8 partm we transfer the color format from RGB into YCbCr, as known as MPEG Compression. Here, **Y** means luminance, **Cb** means Chroma Blue, and **Cr** means Chroma Red.
These three quantities are derived by the equations:

$$\mathrm{Y} = \mathrm{K}_R \cdot R + K_G \cdot G + K_B \cdot B$$
$$C_B = \frac{1}{2} \cdot \frac{B-Y}{1-K_B}$$
$$C_R = \frac{1}{2} \cdot \frac{R-Y}{1-K_R}$$

Here, R, G, B are value of R, G, B in RGB Format. K_R, K_G, K_B are defined with values differently. In ITU-R BT.601 standard, they are defined in the following way:

$$\mathrm{K}_R = 0.299$$
$$K_G = 0.587$$
$$K_B = 0.114$$

[3]

Downsampling: Since human eyes are insensitive to the red and blue chroma, compared with brightness, JPG compress Cb and Cr, but don't compress

Y(luminance). So, JPG compress the Cb and Cr channel to **about** $\frac{1}{4}$ of its original size.This compression is lossy.

Color bleeding: For chrominance channels, which is less sensitive to human eyes, can be "bleed", which means 4 pixels (2×2 block) can be averaged into a single color. For some blocks across the sharp edge, the color is bleeded.

Discrete cosine transform:
The, elements in Y, U, V Component are in range (0 255). for each element, minus 128. So the element become in range (-128 127).

Then, use 2-Dimonsion Discrete cosine transform (showed in the fumular below), we can transfer the 8×8 matrix into a new 8×8 matrix whose (0, 0) element shows the DC Component, and the other elements show the AC Component. This transfer is reverseable, so when in decoding, we can just use the reverse transfer to decode the image.

$$G_{u,v} = \frac{1}{4}\alpha(u)\alpha(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g_{x,y}\cos[\frac{(2x+1)u\pi}{16}]\cos[\frac{(2y+1)v\pi}{16}]$$

g is the 8×8 matrix before transfer, G is the 8×8 matrix after transfer.

$$\alpha(x) = \begin{cases} \frac{1}{\sqrt{8}}(x=0) \\ \frac{1}{2}(x \neq 0) \end{cases} \tag{1}$$

$$u, v \in \{0, 1, 2, 3, 4, 5, 6, 7\}$$

x, y is the index in the original matrix

Also, the DC Component is much larger than AC Component in magnitude, so we can then do the transfer below.

Quantization: Quantization matrix Q is a 8×8 matrix which is included in the JPG Code system.
Then, let M be the matrix whose each element equals to the correspond element in matrix G devided by the Quantization matrix.

$$M_{u,v} = \frac{G_{u,v}}{Q_{u,v}}$$

Here, $A_{u,v}$ means the u^{th} row, v^{th} calcumum element of matrix A.
And then round the elements in matrix M. We'll find that the right-down part of the matrix all become 0.

Compression: However, when expend, the zeros are not togather, which is difficult to compress. So, we expend the matrix with zig-zag like scan like this to make the zeros stay together:

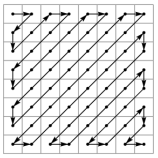


Figure 2: How to scan throught the matrix zigzagly

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Figure 3: example matrix

For exmple, the 8×8 matrix is like this:

This encoded into:
$-26, -3, 0, -3, -2, -6, 2, -4, 1, -3, \cdots$

Then, use Huffman enocoding to compress the data, since a lot of zeros are stay together.
[1] [2] [4]

Decoding is a reverse process of the encoding process

2.2 Why JPG is samller than BMP in size

Bitmap image record every pixel's RGB information. For 24-bit RGB(commonly used), every pixel will take up 24bits. So for a $m \times n$ pixels image, beside the head information and other things, the data will take up $24mn$ bits.

However, for JPG image, the size depends on the downsampling method and the matrix after quantization. The Huffman compression reduce the size by avoiding repeating the same element (here is 0). So JPG is smaller than bit map image.

3 How compression software works

3.1 Algorithm ZIP Compression use

The compression method ZIP use has two part: LZ Compression and Huffman Compression.

LZ: This coding method work in the way below:

For a data, the LZ compressor firstly scan through the data with a sliding window. For the first time it meet a piece of binary, it storage it in the conmmon way. The second time it meet the same binary string, it storega it in the form: (distance, length). For example, for the ascii string below:

abcdabcdabcd

in binary, it is:

011000010110001001100011011001000110000101100010011000110110010001100001011000100110001101100

we define the sliding window of length 4.

So, the binary string above is encoded into:

$$01100001(32,8)(64,8)01100010(32,8)(64,8)01100011(32,8)(64,8)01100100(32,8)(64,8)$$

For **Huffman Compression** Use Huffman compression to compress the distances to make it smaller in size.

[5]

Table 3: Test files

File name	File size	File type	Detail
test_0	1 byte	plain text	
test_1	1,518 bytes	plain text	highly repeated
test_2	36,440 bytes	plain text	highly repeated
test_3.bmp	6,220,854 bytes	BMP Image	
test_4.jpg	342,538 bytes	JPEG Image	the jpg format of the bmp image above

Table 4: Test results

File name	Size before compress	Size after compress	Compression ratio
test_0	1 byte	111 bytes	11100 %
test_1	1,518 bytes	136 bytes	9.0 %
test_2	36,440 bytes	252 bytes	0.69 %
test_3.bmp	6,220,854 bytes	693,909 bytes	11.2 %
test_4.jpg	342,538 bytes	311,022 bytes	90.8 %

4 Result(compression ratio)of the compression

Test software:

Windows Compressed(ziped) Folder

Conclusion of the test:

This test successfully tested that:

JPG image is much smaller than bmp image. in other words, this means the jpg is Compressed. Also, the hight compression ratio of JPG shows that it has been commpressd and can't be compressed a lot anymore.

ZIP compress is quite efficient for long / highly repeated file. For short / less repreted file, it may even make the file bigger.

References

[1] 追梦辅导班. "JPEG格式." baike.baidu.com. , 6 Nov. 2019. Web. 17 Nov. 2019.

[2] McAnlis, Colt. "How JPG Works." freecodecamp 26 Apr. 2016. Web. 17 Nov. 2019.
https : //www.freecodecamp.org/news/how − jpg − works − a4dbd2316f35/.y8yys5lh0.

[3] TIM邓肯. "RGB与YCbCr." CSDN. 26 Nov. 2018. Web. 17 Nov. 2019.
https : //blog.csdn.net/hjhjhx26364/article/details/84548911.

[4] "JPEG." Wikipedia. 14 Nov. 2019. Web. 17 Nov. 2019.
https : //en.wikipedia.org/wiki/JPEG.

[5] 天健胡马灵越鸟. "ZIP压缩算法原理解析." CSDN. 28 May. 2019. Web. 17 Nov. 2019.
https : //blog.csdn.net/pansaky/article/details/90641343.

[6] xgf415. "霍夫曼编码." CSDN. 22 Sep. 2016. Web. 17 Nov. 2019.
https : //blog.csdn.net/xgf415/article/details/52628073.