Research on the Technology of Compressing Images in Developing Android Mobile Application

Zou Shaowu[1], Su Guibin[2]

[1] College of Computer and Information Engineering, Inner Mongolia Normal University, Hohhot 010022, Inner Mongolia, China
[2] College of Network Technology, Inner Mongolia Normal University, Hohhot 010022, Inner Mongolia, China
wangzsw@aliyun.com, ciecsgb@imnu.edu.cn

Abstract - In the Android mobile applications, transferring pictures is the main reason that users consume their network data flow. Although efficient image compression method is an effective way to solve the problem, the image compression method does not be extensively studied on the Android platform. In order to achieve the purpose of quickly and efficiently compressing images in developing Android mobile application, we pre-process image with Android API before compressing image, and accomplish the processing of the pretreatment image's color conversion, FDCT transform, quantization, coding and other operations to achieve the processing of JPEG image compression encoding under the Android platform. The experimental results show that the method can efficiently compress images, and guarantee higher compression ratio and good image quality.

Index Terms - Android, JPEG, image compression

1. Introduction

Android system is favored by lots of mobile application developers as its openness, abundant hardware support and convenient way of development since Google officially released Android 1.0 mobile phone operating system in September, 2008. Many android applications require users to send photos to the remote server, however, with the development and iteration of mobile phone hardware, the phone's camera has a higher pixel and photographs took by mobile phone occupy more storage space of data. If directly sending the photographs in the phone to the remote server via the network, processing these photographs contain large image information will bring great pressure to storage capacity of the memory, bandwidth of the channel and the processing speed of the remote server. To solve these problems simply by increasing the memory's capacity, channel's bandwidth and the remote server's processing speed is unrealistic. Thus, we need to compress these photographs to reduce the data transmission of network traffic, the cost of network resources and accelerate the transmission speed. In this article, we research the Bitmap class of processing image in the Android API and image compression standard of JPEG algorithm, and achieve the purpose of quickly and efficiently compressing images.

2. Related technology

A. *Bitmap class in Android API*

Bitmap as the most important class for processing image in Android API, provides the methods of accessing the information of image, cropping image, rotating image, scaling image, saving the image file in the specified format and other operations. Bitmap class is in the package of android graphics. As the constructor of Bitmap class is private, Bitmap objects can not be created directly. But it can be instantiated through the auxiliary interface for creating Bitmap objects. And BitmapFactory class creates Bitmap objects through JNI interface from various sources, including files, streams, and byte-arrays.

Bitmap class provides several methods for operating bitmap. The method of compression can write a compressed version of the bitmap to the specified outputstream. If this returns true, the bitmap can be reconstructed by passing a corresponding inputstream to BitmapFactory.decodeStream().

B. *The JPEG standard*

The name "JPEG" stands for Joint Photographic Experts Group, the name of the committee that created the JPEG standard and other still picture coding standards[1]. JPEG is a generic standard for the still image compression, which suited for gray-scale images and color images. The degree of compression can be adjusted, allowing a selectable tradeoff between storage size and image quality. The Greater of compression ratio we choose, the lower quality of image we will get; contrary, the smaller of compression ratio we choose, the better quality of image we will get.

JPEG defines a variety of operating modes. There is a basic mode based on 8 × 8 DCT block sequence encoding which divides an image into some 8 × 8 blocks, and processes DCT, quantization and entropy coding from left to right, top to bottom. The encoding flowchart shown in Fig. 1

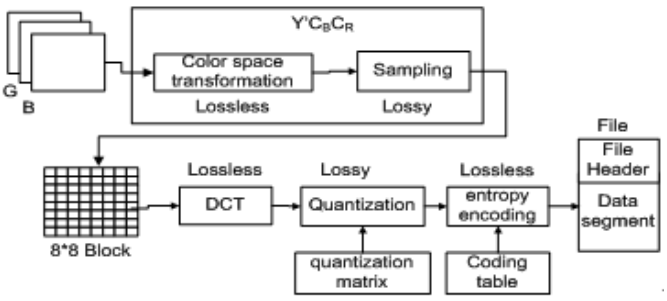


Fig. 1 The encoding flowchart

The main calculation steps of JPEG compression encoding.

1) Color space transform

The image should be converted from RGB into a different color space called Y'C_BC_R before JPEG encoding. The Y' component represents the brightness of a pixel, and the C_B and C_R components represent the chrominance (split into blue and red components). The Y', C_B and C_R components can be encoded separately. The conversion formula[2] is given by:

$$Y = 0.2990R + 0.5870G + 0.1140B$$

$$C_B = -0.1687R - 0.3313G + 0.5000B + 128$$

$$C_R = 0.5000R - 0.4187G + 0.0813B + 128$$

2) Discrete cosine transform(DCT)

Discrete cosine transform is widely used in image compression and is the important mathematical foundation of JPEG, MPEG and other data compression standard[3]. DCT transform image spatial-domain matrix (P matrix) converted to the frequency-domain matrix (T matrix). This conversion process is reversible. The DCT transform schematic diagram shown in Fig. 2:

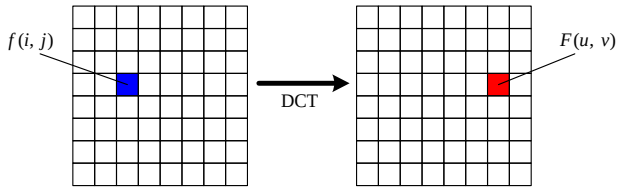


Fig. 2 The DCT transform schematic diagram

The two-dimensional inverse DCT is given by:

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

where

f(x,y) is the reconstructed pixel value at coordinates (x,y)

F(u,v) is the reconstructed approximate coefficient at coordinates (u,v)

when u=0,v=0,C(u), C(v)= $1/\sqrt{2}$

when u>0,v>0,C(u), C(v)=1

3) Quantization

DCT redistributes the energy of spatial and reduces the correlation of image[4]. DCT itself does not achieve the effect of data compression. In order to compress image, the data which is converted needs to be properly quantized. Quantization filters the information which impacts less on the human visual effects to ensure image quality. Quantization can filter out high frequency components and preserve the low-frequency components. Because the human eyes are not sensitive to high-frequency components, so we need to filter out high-frequency components and preserve the low-frequency components. In the quantized 8 * 8 block, the top-left corner entry with the rather large magnitude is the DC coefficient. The remaining 63 coefficients are called the AC coefficients. We need to quantify the 64 DCT coefficients with quantization table recommended by JPEG. The quantized DCT coefficients are computed with

$$Value(i,j) = [T(i,j)]/Q(i,j)$$

where:

$T(i,j)$ is the value of matrix in 8 * 8 block

$Q(i,j)$ is the value of quantization table

The quantization matrix is as follows:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Brightness quantization matrix

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

Chrominance quantization matrix

4) Entropy coding

Entropy coding is a special form of lossless data compression. It involves arranging the image components in a "zigzag" order employing run-length encoding (RLE) algorithm that groups similar frequencies together, inserting length coding zeros, and then using Huffman coding on what is left.

(1)DC coefficient encoding

AS the strong correlation between adjacent blocks in the image, the difference between the two is encoded rather than the actual value. This method called DPCM coding which can improve the compression ratio. The difference formula is given by:

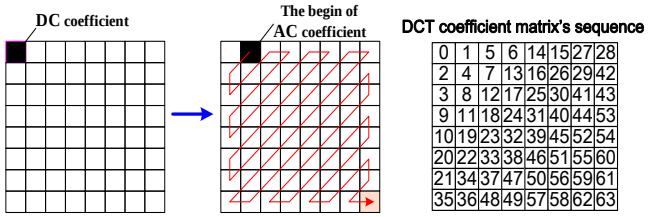
$$DIFF = DC_j - DC_{j-1}$$

There are two symbols for encoding DC coefficients. Symbol 1 is a size (Size); Symbol 2 is the amplitude value (Amplitude). Size is the number of bits required to represent x and Amplitude is the bit-representation of x. The symbols should be encoded by DC encoding table.

(2)AC coefficient encoding

Most of AC coefficients in the DCT coefficient matrix are float close to zero. After rounding, a large number of AC coefficients are zero in each 8 * 8 block. In order to improve encoding efficiency and compression ratio, the AC coefficients which Value of 0 should be stored together in sequence, and the coefficient matrix should be sorted in a "zigzag" order. There are two symbols stand for the AC coefficients with two bytes. The Symbol 1 is step and size (RunLength, Size). It marks the end of block (EOB) with (0,0) and indicates ZRL with the (15,0). When the step length is greater than 15, the number of ZRL should be increased to resolve the situation. But the maximum number of ZRL should not be more than three. The symbol 2 is amplitude values (Amplitude). The symbols should be encoded by AC encoding table.

The zigzag sequence for the quantized coefficients is shown below.



3. Specific Work

A. *The total flow of the compression process*

The flowchart of image compression with Android API is shown in Fig. 3. Firstly, we should put the image into stream object with BitmapFactory in Android API. And then preprocess the resource image by calling the Android application program interface. Pretreatment of the specific work is to scale down the image's height and width for reducing the image size by the Bitmap's method of compress in Android API. After preprocessing the image, preprocessing image's DCT, quantization, entropy encoding by the JPEG standard. Finally, we save the compressed image to the Android phone's SDCard.

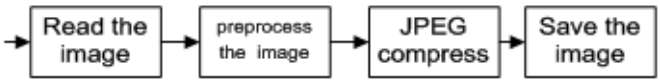


Fig. 3 The simple flowchart of compressing image

B. *The JPEG compression process*

The flowchart of the JPEG compression process is shown in Fig. 4. We should convert the preprocessed image from RGB into Y'C_BC_R and do the 8 × 8 two-dimensional FDCT (forward discrete cosine transform) to the image to generate a frequency coefficient table. Then according to the luminance, chrominance quantization table recommended by JPEG to quantify these frequency coefficients table in order to filter out high-frequency components and preserve the low -frequency components. After quantization, the DC coefficients need to be encoded by the differential coding and AC coefficients need to be encoded by the run-length coding. Then we do entropy coding to the DC coefficient and AC coefficients respectively with Huffman encoding. Finally, we save the compressed documents to the specified location.

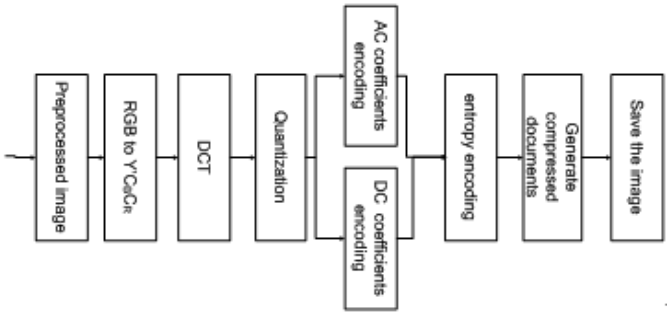


Fig. 4 The flowchart of the JPEG compression

1) *RGB to Y'C_BC_R*

In the module of converting RGB into Y'C_BC_R, we shift appropriately every pixel of the input color image to obtain the values of each pixel's RGB and get the component values of each pixel's Y', C_B and C_R through the conversion formula of RGB to Y'C_BC_R. The processing procedure is shown in Fig. 5.

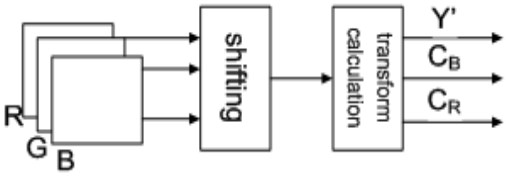


Fig. 5 RGB to Y'C_BC_R

2) *Discrete cosine transform*

If we do the transform calculation to the image according to the formula of two-dimensional DCT, the completion of the 8 × 8 pixel DCT calculation requires 8 × 8 × 8 × 8 = 4096 multiplications and 8 × 8 × 8 × 8 = 4096 additions[5]. To Android system, such a large number of operations will cause great CPU resource consumption and even cause memory overflow. Therefore, the two-dimensional DCT formula needs to be simplified into two formulas of one-dimensional:

$$y(u,v) = \frac{1}{2}\sum_{i=0}^{7} c(u)z(u,i)\cos\frac{(2i+1)u\pi}{16}$$

$$z(u,i) = \frac{1}{2}\sum_{j=0}^{7} c(v)x(i,j)\cos\frac{(2j+1)v\pi}{16}$$

The calculation amount is reduced to 2 × (8 × (8 × 8)) = 1024 multiplications and 2 × (8 × (8 × 8)) = 1024 additions using the two formulas of one-dimensional. Though only one-fourth of calculation amount will take, the operation amount is still too large and the method need to be further optimized. Therefore, according to the law of one-dimensional DCT, we do the one-dimensional line conversion and one-dimensional column conversion to the 8 × 8 pixel block data with AAN method. The number of operations dropped to 29 × 8 × 2 = 464 additions and 5 × 8 × 2 = 80 multiplications, greatly improving the computing speed.

3) *Quantization process*

The quantization initializes an 8*8 quantization matrix according to the luminance and chrominance quantization table[6] recommended by JPEG at first, and set the value of quantization matrix based on quality level. When the quantization table is initialized, quantify the input data which contains an 8*8 pixel block, and finally, output pixel quantization matrix.

4) *The entropy encoding process*

Since AC and DC coefficients need to be encoded in different ways[7], so we process the AC and DC coefficients separately. We encode DC coefficients with DPCM coding and process AC coefficients with run-length encoding. The

current DC coefficient minus the last DC coefficient in front of it. Then we get the DC coefficient difference called △DC and according to △DC lookup the coding table to find the corresponding CODE value. The value of △DC and CODE need to be linked and written to the file. For the AC coefficients encoding, AC coefficient coding is more complex and requires statistical consecutive zeros and find the corresponding CODE values from code table. Fig. 6 is the flowchart of AC coefficient coding:

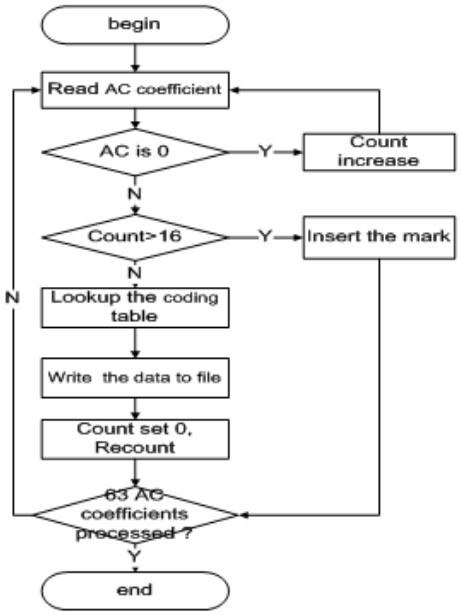


Fig. 6 the flowchart of AC coefficient coding

4. The Experimental Results

The following part is the experimental test results. The test picture size is 2846.72kb. The image can be compressed according to different quality levels. The higher quality levels we choose, the better image quality will be got, but the greater storage space will be occupied for storing the image; contrary, the picture's quality is worse, and the smaller storage space will be occupied.

Original picture quality level: 60 quality level:1

Quality	Astart	Aend	Cstart	Cend	Acost	Ccost	Tcost	Ssize(KB)	Dsize(KB)	Cratio	Effect
1	46.71	48.65	48.65	50.23	1.94s	1.585s	3.525s	2846.72	15.71	181.2	Poor
25	51.51	53.48	53.48	55.07	1.97s	1.595s	3.565s	2846.72	36.62	77.74	Fair
40	35.56	37.54	37.54	39.23	1.975s	1.695s	3.67s	2846.72	49.16	57.91	Fair
60	9.152	11.11	11.11	12.68	1.955s	1.57s	3.525s	2846.72	65.77	43.28	Good
80	19.11	21.06	21.06	22.82	1.95s	1.315s	3.265s	2846.72	103	27.64	Good
100	49.31	51.24	51.24	52.93	1.935s	1.69s	3.625s	2846.72	526	5.41	Excellent

Quality: Image compression quality level
Astart: The start time point of preprocessing image
Aend: The end time point of preprocessing image
Cstart: The start time point of JPEG compression
Cend: The end time point of JPEG compression
Acost: The total cost time of preprocessing image
Ccost: The total cost time of JPEG compression
Tcost: The total cost time of image compression
Ssize: Source image size Dsize: Destination image size
Cratio: compression ratio
Effect: The visual effects of the image compressed

4. Conclusions

In this paper, we pre-process image with Android API before compressing image, and accomplish the processing of the pretreatment image's color conversion, FDCT transform, quantization, coding and other operations to achieve the processing of JPEG image compression encoding under the Android platform. The experimental results shows that the method proposed in this paper reduces the image's capacity and data space, achieves the purpose of to reducing the network traffic flow, speeding up the speed of transporting data and improving network efficiency in mobile application development.

References

[1] Hu Dong. Basic approach and international standard of still image encoding. Beijing: Bejing University of Posts and Telecommunications Press, 2003: pp.1-238.
[2] Qian Penghe. Design and implementation of handwritten information on the mobile terminal based on JPEG compression technology. Shanghai: Shanghai Jiao Tong University, 2007.
[3] Wu Ying. The application of DCT transform in image compression. Computer and Modernization, 2013, (8):pp. 103-106.
[4] Ma Yuanyuan, Yang Feng, Xin Ke, Jiao Fangchao. Research on the compression of JPEG images based on DCT. Computer Technology and Development, 2011, 21(8):pp. 133-136.
[5] Yu Lei, Li Lei, Cui Jianming, Chen Xinhua. FPGA-based design and implementation of JPEG compression coding. Microcomputer & Its Applications, 2012, 31(21):pp. 23-25.
[6] Gong Shengrong, Liu Cunping. Data image processing and analysis. Beijing: Tsinghua University Press, 2008.
[7] Yang Jing. FPGA-based research and implementation of image compression JPEG basic model. Shanghai: Donghua University, 2008.