International Conference on Recent Trends in engineering & Technology - 2013(ICRTET'2013)
Special Issue of International Journal of Electronics, Communication & Soft Computing Science & Engineering, ISSN: 2277-9477

Image Data Extraction for Developing Standalone Application Programs

Amol Bharat Ranadive **Prof. S. R. Ganorkar**

Abstract— **Many complex applications can be developed in image processing as once the data is available which represents the image pixel matrix. Additional key attributes that define dimension and resolution of image become necessary for any further processing. For a standalone programming in image processing application the first step to extract vital information and this sometime becomes difficult. A set of readymade library in tool environment is used and that leads to hindering any possibility to develop an application program. A standalone application program is expected to work without such dependencies of tool environment. The proposed methodology here is an attempt to address for extraction of the image data along with relevant information about an image file through a raw programming method. This is to empower engineers in electronics and computer science for their research and development of image processing applications. Those applications are intended to run in standalone mode, without any dependency of readymade tool. The present research assumes that C is most preferred programming language for developing applications for desktop or an embedded system.**

KeyWords—**image processing, standalone image application, BMP file programming**

I. INTRODUCTION

Through a variety of picture formats, we find the storage format of the picture to show the two-dimensional code. By analyzing the BMP file formats, we have a support mechanism generate image data into a text file algorithm and a function. Two-dimensional code is a new and developing code which has gradually entered people's life [1]. General combination of two-dimensional code components generated data can be generated on-line network of two-dimensional picture code. Mainly researches a type of image information acquisition system, including the establishment of system image acquisition tasks, the encoding and storage of captured images, and other treatment methods [2]. Typically mathematical software tools such as MatLab and SciLab are better with following objectives.

- Data Exploration,Acquisition,Analyzing &Visualization
- Engineering drawing and Scientific graphics
- Analyzing of algorithmic designing and development
- Mathematical functions and Computational functions
- Simulating problems prototyping and modeling
- Application development programming using GUI building environment

However, going ahead for any standalone application, be it on PC or an embedded system/gadget it become really difficult by achieving this through programming [4]. The proposed methodology here is an attempt to address for extraction of the image data along with relevant information about an image file through a raw programming method. This is to empower engineers in electronics and computer science for their research and development of image processing applications. Those applications are intended to run in standalone mode. A digitized image is described by an N x M matrix of pixel values are nonnegative scalars, that indicate the light intensity of the picture element at (i,j) represented by the pixel[3, 5, 6] ref figure-1.

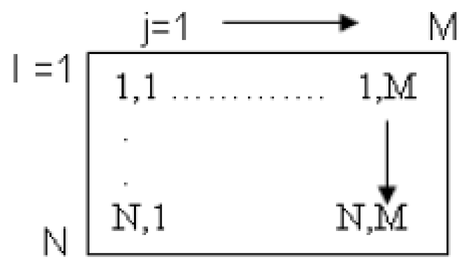


Fig-1: Image model

II. DIGITAL IMAGE FILE STRUCTURE

When we see a picture on our monitor or use our digital camera (or scanner), the image we are viewing or dealing with is not continuous like a pencil drawing – it is made up of many small elements next to each other. When we have enough elements, we get the illusion of a picture or image.

A digital image is a numeric representation (normally binary) of a two-dimensional image. Depending on whether the image resolution is fixed, it may be of vector or raster type. Without qualifications, the term "digital image" usually refers to raster images also called bitmap images.

Image file formats are standardized means of organizing and storing digital images. Image files are composed of digital data in one of these formats that can be rasterised for use on a computer display or printer. An image file format may store data in uncompressed, compressed, or vector formats. Once rasterised an image becomes a grid of pixels, each of which has a number of bits to designate its colour equal to the colour depth of the device displaying it.

ISBN No.: 978-81-926080-0-6/22-24 Feb 2013 ©2013 SNJB's KBJ CoE

BMP files are made of millions and millions of dots called ‘pixels’, with different colours and arrangements to come up with an image or pattern. It might an 8-bit, 16-bit or 24-bit image.

There is need for information to be gathered about the bitmap file format. Microsoft defined the bitmap file format. In hopes of making this a popular file format, Microsoft made structures so that the information can be extracted easily from the bytes of a file. Bitmaps have two headers that contain the information needed to extract and display the image. The first 14 bytes of the file make up the first header, which can be read into a structure called the BitmapFileHeader. The information contained in this structure can be seen in Table 1.

Table 1: Bitmap File Header Structure Items

Item	**Name**	**Size Description**
bfType	2 bytes	ASCII for "B" and "M"
bfSize	4 bytes	Size of the File in bytes
bfReserved1	2 bytes	Reserved Equals 0
bfReserved2	2 bytes	Reserved Equals 0
bfOffBits	4 bytes	Number of Bytes to the Picture Data

Table 2: Bitmap Information Header Structure Items

Item Name	Size	Description
biSize	4 bytes	Size of the Bitmap Info Header in bytes
biWidth	4 bytes	Width of the Bitmap in pixels
biHeight	4 bytes	Height of the Bitmap in pixels
biPlanes	2 bytes	Number of Planes (Equals 1)
biBitCount	2 bytes	Number of Bits Per Pixel (1,4,8,16,24, or 32)
biCompression	4 bytes	Type of Compression (0,1,2)
biSizeImage	4 bytes	Size of the Picture Data in bytes
biXPelsPerMeter	4 bytes	Horizontal Resolution (Pixels/Meter)
biYPelsPerMeter	4 bytes	Vertical Resolution (Pixels/Meter)
biClrUsed	4 bytes	Number of Actual Colours Used
biClrImportant	4 bytes	Number of Important Colours (0 = all)

The next header in the bitmap can be read into a structure called the BitmapInfoHeader. The length of this header is determined by the first four bytes in the header, the biSize field. Generally, the length of the header is 40 bytes long. Table 2 shows the items that are contained in this structure.

An optional colourmap follows the BitmapInfoHeader in the bitmap image file. Each colour in the colourmap is four bytes long and is contained in a RGBQUAD structure. Table 3 shows the items within the RGBQUAD structure. The program determines whether or not a colourmap exists by examining the bfOffbits field in the BitmapFileHeader. If this field is greater than the total size of the two headers, then a colourmap is present.

Table 3: RGBQUAD Structure Items

Item Name	Size	Description
rgbBlue	1 byte	Blue Intensity
rgbGreen	1 byte	Green Intensity
rgbRed	1 byte	Red Intensity
rgbReserved	1 byte	Unused (Equals 0)

The biBitCount member of the BITMAPINFOHEADER structure determines the number of bits that define each pixel and the maximum number of colours in the bitmap. These members can take the any of the following values as depicted in the table-4.

Table-4: Number of bits and maximum colours

Value	Meaning
1	Bitmap is monochrome and the colour table contains two entries. Each bit in the bitmap array represents a pixel.
4	Bitmap has a maximum of 16 colours. Each pixel in the bitmap is represented by a 4-bit index into the colour table. For example, if the first byte in the bitmap is 0x1F, the byte represents two pixels.
8	Bitmap has a maximum of 256 colours. Each pixel in the bitmap is represented by a 1-byte index into the colour table. For example, if the first byte in the bitmap is 0x1F, the first pixel has the colour of the thirty second table entry.
24	Bitmap has a maximum of 2^24 colours. The bmiColours (or bmciColours)

For present scope we are following to bitmap image files. However, the concluding part of the paper also hints on obtaining data from other file formats of image.

A colourmap is used in different ways depending on the biBitCount field in the BitmapInfoHeader. If the biBitCountequals 24, the colourmap is used to list the most important colours of the image, but the colourmap is not actually used by the picture data. If the biBitCount equals 8, each pixel in the picture data is an 8-bit pointer to a colour in the colourmap. A colourmap in an 8-bit bitmap has a maximum 256 colours since the pointer into it can only be 1 byte (2^8 = 256). The biBitCount field can be other values than 8 or 24, but since these are uncommon the current project does not deal with them. Following the colourmap (if a colourmap exists) is the picture data. The picture data also varies depending on the value of biBitCount. If the biBitCount equals 24, then each pixel is represented by three bytes. There is exactly one byte for each intensity: red, green and blue. If the biBitCount equals 8, each pixel is represented by 8- bits, which point to a RGBQUAD structure in the colourmap.

One complication to the picture data is that each line of pixels in a picture, known as a scan line, must start on a LONG boundary. Since a LONG is four bytes in length, a LONG boundary is an address that is evenly divisible by four. The pixels in the image formats being discussed here are either 1 byte (8-bit images) or 3 bytes (24-bit images) long in the picture data. The number of bytes needed to accurately display a scan line may not be divisible by four. Since images can be any number of pixels wide, the end of the scan line may not end on a LONG boundary. In such cases, extra bytes of value zero are padded to the end of the scan line in order to make the scan line end on a LONG boundary. The next scan line can then start on the LONG boundary as well.

However, the red plane in colour image, contents most significant information and is less prone to error during the processing of image. Each pixel in colour image in BMP file is represented by 24-bits. The red plane consists of the most significant byte, carrying maximum information of the image.

The biClrUsed member of the BITMAPINFOHEADER structure specifies the number of colour indexes in the colour table actually used by the bitmap. If the biClrUsed member is set to zero, the bitmap uses the maximum number of colours corresponding to the value of the biBitCount member.

The biCompression member tells whether the image is compressed or not. Windows versions 3.0 and later support run-length encoded (RLE) formats for compressing bitmaps that use 4 bits per pixel and 8 bits per pixel. Uncompressed images are taken for this project and they contain a value 0 in this field.

BMP Files are stored in much the same way that Microsoft Windows internally stores its bitmaps. Peculiarities to note are that:

- The images are stored upside down. The first line in the file is the bottom line of the image and the last line in the file is the first line of the image. This means that if you are computing an image with resolution of 256 x 192, to be displayed in a system where the (0, 0) pixel is the top left, then the first pixel to be computed and written to the BMP file is for pixel (191, 0).
- In 24-bit images RGB values are stored in Blue Green Red order.

III. PROGRAMMING TECHNIQUE AND STRUCTURES

The present discussion refers to programming in C. This is because C is widely used for both programming a desktop application and an embedded system applications. The image processing application that are intended to either run on desktop or on an embedded system need to be developed using C. There is large support of compilers for C language for almost every operating system and processors.

The purpose of this project is to manipulate Bitmap images and construct some statistics out of it. Bitmap are simple to read in C, and easy to understand in terms of structure. Because this image format is simple, it is also quite easy to manipulate them such as modify the colour, reversing them, etc. By following carefully the standard data structure of this file, the data can be stored in an array and modified. The reading (and writing) of the .bmp file is performed using the usual techniques of binary files manipulations. Structures can be used, since they provide a straightforward way to read a bunch of data at once and are easier to manipulate than a collection of fields. Finally, it is recommended to use 24-bits images since they are the easiest to manipulate as each colour (Red Blue Green) channel has been separated.

A struct type is header information stored in a bitmap (BMP) file. BMP, a format invented by Microsoft stores the image using a schema as follows.

The first 14 bytes is reserved for information given by the following struct

```
typedefstruct {
        unsigned short int type;     /* BMP type identifier */
        unsignedint size;            /* size of the
file in bytes*/
        unsigned short int reserved1, reserved2;
        unsignedint offset;          /*      starting
address of the byte */
} HEADER;
```

The next 40 bytes are reserved for a structure as follows.

```
typedefstruct {
        unsignedint size;            /* Header size in bytes
*/
        intwidth,height;             /* Width and height in
pixels */
        unsigned short int planes; /* Number of colour
planes */
        unsigned short int bits;     /* Bits per pixel */
        unsignedint compression;  /* Compression type */
```

unsignedintimagesize; / Image size in bytes */*
intxresolution,yresolution; / Pixels per meter */*
unsignedintncolours; / Number of colours */*
unsignedintimportantcolours; / Important colours */*
} INFOHEADER;

Suppose we are interested in extracting this information from a BMP file. First we need
to read a block of 54 bytes using fread function as follows. Since BMP files are binary
files (recall that there are two types of files, ASCII and Binary) and reading bytes from a
BMP file needs to be done using fread (instead of fscanf for formatted data)

The prototype of the fread function is given by following code snippet.

#include <stdio.h>
*size_tfread(void *ptr, size_t size, size_tnitems, FILE *stream);*

The fread() function reads, into the array pointed to by ptr, up to nitems members whose size is specified by size in bytes, from the stream pointed to by stream.

The BMPHEAD structure uses quite a few long values. In C, literal constants must be expressly casted to long e.g. 0L, 54L. Integer constants must also be expressly casted to long. NEVER assign to a long or do calculations to assign to a long unless the value being assigned is long or explicitly converted to long. The same is true for floats and doubles.

IV. HEX NUMBER REPRESENTATION IN TEXT FILE

The developed program utility is a console-based program used to display the contents of a binary file in hexadecimal format. The rebuild function parses the byte string and writes each byte to the specified output file. Figure 1 shows an example of simple 16X16 size image followed by the hex value that is extracted for the pixels of that image.

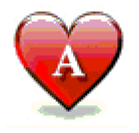
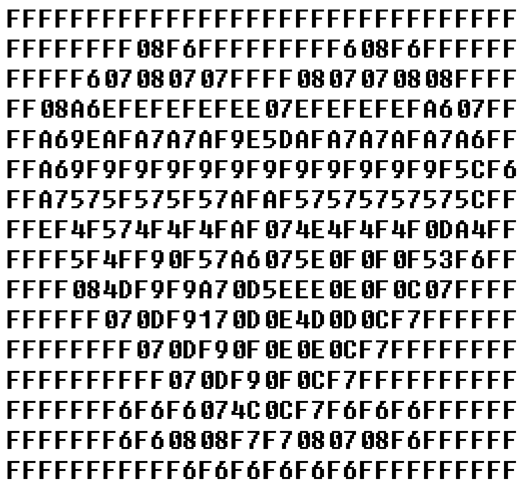


Fig.2 Illustrated image

Hexadecimal pixel values extracted from example image looks as shown in figure-2.



```
FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
FFFFFFFF08F6FFFFFFFFF608F6FFFFFF
FFFFF607080707FFFF0807070808FFFF
FF08A6EFEFEFEFEE07EFEFEFEFA607FF
FFA69EAFA7A7AF9E5DAFA7A7AFA7A6FF
FFA69F9F9F9F9F9F9F9F9F9F9F9F5CF6
FFA7575F575F57AFAF57575757575CFF
FFEF4F574F4F4FAF074E4F4F4F0DA4FF
FFFF5F4FF90F57A6075E0F0F0F53F6FF
FFFF084DF9F9A70D5EEE0E0F0C07FFFF
FFFFFF070DF9170D0E4D0D0CF7FFFFFF
FFFFFFFF070DF90F0E0E0CF7FFFFFFFF
FFFFFFFFFF070DF90F0CF7FFFFFFFFFF
FFFFFFF6F6F6074C0CF7F6F6F6FFFFFF
FFFFFFF6F60808F7F7080708F6FFFFFF
FFFFFFFFFFF6F6F6F6F6F6FFFFFFFFFF
```

Fig-3: Hexadecimal pixel values

Here the two character represent a single byte, where first character is a most significant nibble and the second as a least significant four bits. Very often it is necessary to represent a bitmap with one colour depth onto a device with different colour depth capabilities. Of course if the destination device has better colour than the bitmap then there is no issue since the bitmap can be exactly represented. In the reverse situation where the destination has different and lower capabilities, then the bitmap has to be converted into something that gives the best possible representation. The next sections comprehend upon same.

V. CONVERTING 24 BIT COLOUR TO 8 BIT REPRESENTATION

There is 8 bits allocated to each red, green, and blue component. In each component the value of 0 refers to no contribution of that colour, 255 refer to fully saturated contribution of that colour. Since each component has 256 different states there are a total of 16777216 possible colours.
If R is 8 bits of Red pixel, G is 8 bits of Green pixel, B is 8 bits of Blue pixel from bitmap image of 24 bits resolution. Let P be single 8 bits pixel comprising of red, blue and green for a bit required output.Then the conversion from 24 bit per pixel into a single byte (8 bits) is calculated as follows.

R.7 + R.6	= P.7
R.5 + R.4	= P.6
R.3 + R.2	= P.5
G.7 + G.6	= P.4
G.5 + G.4	= P.3
G.3 + G.2	= P.2
B.7 + B.6 + B.5	= P.1
B.4 + B.3 + B2	= P.0

Unused bits from each colour are redundant. P is combined 8 Bit representation of 256 levels of colours.When this is done, we have each pixel value represented with a 8bit hexadecimal number which can be used for further image processing computations.

VI. CONVERTING 16 BIT COLOUR TO 8 BIT REPRESENTATION

Conversion from a 16bit colour to bit representation can be done by discarding alternate bits from the pixel and just carry forwarding the 8 odd bits.

FUTURE WORK

The present work is carried for uncompressed bitmap image files. However there are Image format: JPG, TIFF, BMP, PNG, JP2K, GIF & RAW etc. Each format follows distinct formats for storage of data, compression of data and Type of information. These different formats of files present a distinct set of challenges to handle and obtain data by uncompressing along with key parameters. This will not be easy to carry out in usual C programming. But with the use of higher level language constructs such as Java this could be done. Java language provides a library of ImageIO class. The methods available in that class is helpful to deal with different image formats and extract the vital attributes such as Width, height, RGB-pixel data of those image files.

VII. RESULTS

All hex dump text files have number bytes per line that are equal to number of pixels in horizontal direction of an image file. A new line feed follow the end of the byte string. All pixel values are non-printable characters will be represented by respective hex code which is represented in two characters, where first character represents most significant nibble and later character represents least nibble. Each byte in the byte string is represented by two characters and must be separated by a single space, with a maximum of 255 bytes per string.

CONCLUSION

The computer's development has entered the post-PC era, more and more embedded devices have permeated our daily lives, and image processing system based on embedded technology can be used for industrial control, intelligent transportation, residential surveillance and other fields. To achieve the design of image information storage part in the system through image coding research.

ACKNOWLEDGMENT

I would like to place a deep sense of gratitude for guidance by Prof S. R. Ganorkar and Dr. Ajay D. Jadhav, (P.G. Program Head) for their insightful comments and constructive suggestions to improve the quality of this work.

REFERENCES

[1] Chunying Kang Coll. of Inf. Sci. & Technol., Heilongjiang Univ., Harbin, China, Research on Generation Algorithm of Bmp Format Two-dimensional Code Image, Volume: 5 Page(s): 433 - 436, 14-16 Aug. 2009
[2] Zhigang Fan and Ricardo L. de Queiroz, Identification of Bitmap Compression History: JPEG Detection and Quantizer Estimation, IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 12, NO. 2, FEBRUARY 2003, PP - 230-235.
[3] Dr T. Meyyappan, SM.Thamarai and N.M.JeyaNachiaban, LOSSLESS DIGITAL IMAGE COMPRESSION METHOD FOR BITMAP IMAGES, The International Journal of Multimedia & Its Applications (IJMA) Vol.3, No.4, November 2011.
[4] Chen Qiu-hong Dept. of Comput. Sci. & Eng., Henan Univ. of Urban Constr., Pingdingshan, China, A new embedded image information processing system design, Volume: 4, 26-28 Feb. 2010 Page(s): 543 - 547.
[5] http://en.wikipedia.org/wiki/Windows-bitmap
[6] http://docs.oracle.com/javase/1.4.2/docs/api/javax/imageio/ImageIO.html
[7] SciLab: http://scilab.in/
[8] http://www.java2s.com/Tutorial/Java/0261__2D-Graphics/0334__ImageIO.htm
[9] Microsoft Corporation, "MSDN: About Bitmaps", http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/bitmaps-99ir.asp?framer=tru, 2007.

AUTHOR'S PROFILE

Amol Bharat Ranadive
Amol Bharat Ranadive was born on March 6; 1977. He has completed is BE in Electronics Engineering from Walchand Institute of Technology, under Shivaji University Kolhapur. He is pursuing M.E.in Electronics Engineering in Digital Systems at SCOE, affiliated to University of Pune.
Email address:amolbharat@rediffmail.com

S. R. Ganorkar
Born on August 6; 1965.He has completed his ME in Adv. Electronics Engineering. His research interests are in Artificial Neural Network and Image Processing. He has 24 years of experience, 13 year in Industrial and 11 years of teaching experience. He is presently working as Associate Professor at E & TC department at Sinhgad College of Engineering, Pune. He has published 12 papers in International journal, 13 papers in International conference and 40 papers in national conference. He is life member of ISTE, New Delhi. He is also a fellow of IETE, New Delhi.
email address: srganorkar.scoe@sinhgad.edu