BMP to JPEG – the Conversion Process

[1]Anjali Anand, [2]Dr. Himanshu Aggarwal
[1] M.Tech Student, [2]Professor
Department of Computer Engineering
Punjabi University, Patiala.
[1]anjalianand_87@yahoo.in, [2]himanshu.pup@gmail.com

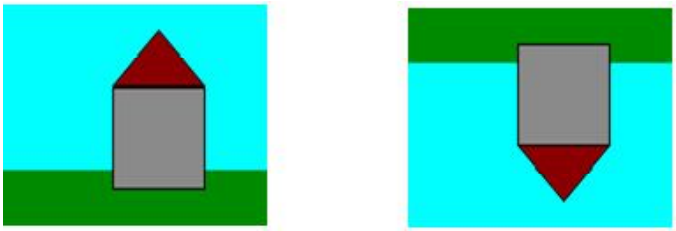
Abstract : *Images of different formats are generated, edited and transmitted on a very regular basis in a vast number of systems today. The BMP format is the native format for the Windows World and vast majority of windows-based applications supports this format. But a BMP image is quite large and voluminous. It becomes cumbersome to move it around in bandwidth constrained systems or where bandwidth is to be conserved for cost purposes such as the World Wide Web. Such scenarios demand use of a conversion technique to a format which is compact, such as JPEG, which compresses the image to a high degree with little loss in perceived quality of the image. This report deals with the steps used for converting an image stored in BMP format to JPEG format.*

1. Introduction

BMP stands for Windows Bitmap which is the native format of the Windows World. JPEG Stands for Joint Photographic Experts Group and is the most widely used format. The BMP file format and JPEG file format along with their file structure are explained below:

2. The BMP File Format

The Microsoft's Windows Bitmap (BMP) File format is a basic file format for digital images in the windows world. This is the native format for the Windows World, so the vast majority of windows-based applications support this format. The BMP format has changed as Windows has grown and changed. There are various versions of BMP files. It is important to know that the rows of pixel data in a BMP file are stored upside down. That means that the topmost row which appears on the screen actually is the lowest row stored in the bitmap [10].



Pixels displayed on screen **Pixels stored in BMP file**

Fig 1: Pixel data is stored upside down in BMP file [10].

BMP File Structure

The Microsoft Windows Bitmap (BMP) file format is a basic file format for digital images in the Microsoft Windows world. This is the native graphics format for the Windows world, so the vast majority of Windows-based software applications support this format. Since BMP was created for Microsoft Windows, it was created for the Intel processors only. Hence, it is all least significant byte first. The BMP file format has grown and changed as Microsoft Windows has grown and changed. There are five or six different versions of BMP files. An excellent source of information for BMP and all other image file formats is [1]. Further information for BMP is in [2] and [3].

BMP files have (1) a file header (2) a bit map header (3) a color table (4) the image data.

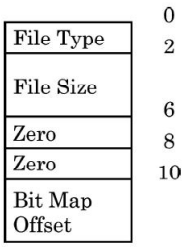


Fig 2: The BMP file header

The file header, as shown in fig2, occupies the first 14 bytes of all BMP files. The first two bytes are the file type which always equals 'BM.' The next four bytes give the size of the BMP file. The next two bytes are reserved and are always zero. The last four bytes of the header give the offset to the image data. This value points to where in the file the image data begins.

The next 40 bytes are the bit map header, shown in fig3. These are unique to the 3.x version of BMP files. The bit map header begins with the size of the header (always 40). Next comes the width and height of the image data (the numbers of columns and rows). If the height is a negative number the image is stored bottom-up. That is the normal format for BMP files. The number of color planes is usually 1.The next two fields deal with image data compression. The compression field is 0 for no compression and 1 for run length encoding compression. The size of bitmap field gives the size of the image data when the data is compressed. It is zero when not compressed, and the software calculates the size of the data. The next two field deal with the resolution of the image data and the final two deal with the colors or gray shades in the image. The horizontal and vertical resolutions are expressed in pixels per meter. The colors field states how many colors or gray shades are in the image. The important colors field states how many of the colors are important to the image [2].

After the headers comes the color table. The BMP color table has four bytes in each color table entry. The bytes are for the blue, green, and red color values. The fourth byte is padding and is always zero.

The final part of the BMP file is the image data. The data is stored row by row with padding on the end of each row. The padding ensures the image rows are multiples of four. The four, just like in the color table, makes it easier to read blocks and keep track of addresses [2].

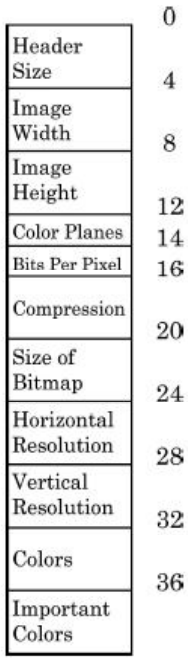


Fig3: The Bitmap Header

3. The JPEG file Format

JPEG is an image compression standard used for storing images in a compressed format. It stands for Joint Photographic Experts Group. The remarkable quality of JPEG is that it achieves high compression ratios with little loss in quality. JPEG format is quite popular and is used in a number of devices such as digital cameras and is also the format of choice when exchanging large sized images in a bandwidth constrained environment such as the Internet [9].

The JPEG algorithm is best suited for photographs and paintings of realistic scenes with smooth variations of tone and color. JPEG is not suited for images with many edges and sharp variations as this can lead to many artifacts in the resultant image. In these situations it is best to use lossless formats such as PNG, TIFF or GIF. It is for this reason that JPEG is not used in medical and scientific applications where the needs to reproduce the exact data as captured and the slightest of errors may snowball into bigger ones [9].

How JPEG works

JPEG divides up the image into 8 by 8 pixel blocks, and then calculates the discrete cosine transform (DCT) of each block The detail about Discrete Cosine Transforms is at [12]. A quantizer rounds off the DCT coefficients according to the quantization matrix. This step produces the "lossy" nature of JPEG, but allows for large compression ratios.

JPEG's compression technique uses a variable length code on these coefficients, and then writes the compressed data stream to an output file (*.jpg). For decompression, JPEG recovers the quantized DCT coefficients from the compressed data stream, takes the inverse transforms and displays the image [4]. Fig4 and fig5 shows this process.

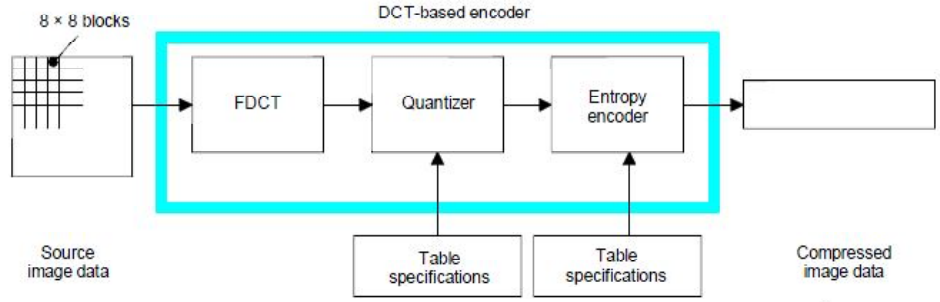


Fig4: JPEG Encoder [5]

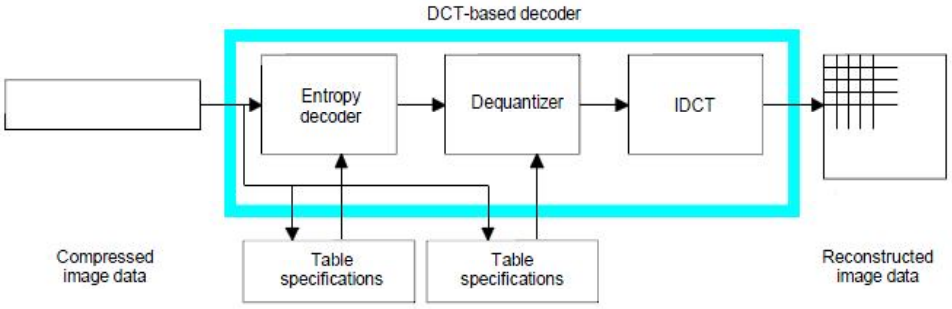


Fig5: JPEG Decoder [5].

The JPEG standard includes two basic compression methods, each with various modes of operation. A DCT-based method is specified for "lossy" compression, and a predictive method for "lossless" compression. JPEG features a simple lossy technique known **as** the Baseline method, a subset of the other DCT-based modes of operation. The overview of JPEG Standard and the Baseline method is at [6].

The JPEG File Structure

A JPEG image consists of a sequence of *segments*, each beginning with a *marker*. Markers are used to break a JPEG stream into its component structures. They are 2 bytes in length, with the first byte always having the value FF16. The second byte contains a code that specifies the marker type. Any number of bytes with the value FF16 may be used as a fill character before the start of any marker. A byte in a JPEG stream with a value of FF16 that is followed by another FF16 byte is always ignored [8].

JPEG markers can be grouped into two general types. Stand-alone markers consist of no data other than their 2 bytes and are shown in table 1; markers that do not stand alone are immediately followed by a 2-byte-long value that gives the number of bytes of data the marker contains.

The length count includes the 2 length bytes but not the marker length itself. JPEG markers with data are shown in table 2.The JPEG standard is fairly flexible when it comes to the ordering of markers within a file. Its strictest rule is that a file must begin with an SOI marker and end with an EOI marker. In most other cases markers can appear in any order.More information about JPEG can be found at [11].

Value	Symbol Used in JPEG Standard	Description
FF01*	TEM	Temporary for arithmetic coding
FFD0-FFD7	RST_0-RST_7	Restart marker
FFD8	SOI	Start of image
FFD9	EOI	End of image

Table1: Stand-alone JPEG Markers

Value	Symbol Used in JPEG Standard	Description
FFC0	SOF_0	Start of frame, baseline
FFC1	SOF_1	Start of frame, extended sequential
FFC2	SOF_2	Start of frame, progressive
FFC3*	SOF_3	Start of frame, lossless
FFC4	DHT	Define Huffman table
FFC5*	SOF_5	Start of frame, differential sequential
FFC6*	SOF_6	Start of frame, differential progressive
FFC7*	SOF_7	Start of frame, differential lossless
FFC8*	JPG	Reserved
FFC9*	SOF_9	Start of frame, extended sequential, arithmetic coding
FFCA*	SOF_{10}	Start of frame, progressive, arithmetic coding
FFCB*	SOF_{11}	Start of frame, lossless, arithmetic coding
FFCC*	DAC	Define arithmetic coding conditions
FFCD*	SOF_{13}	Start of frame, differential sequential, arithmetic coding
FFCE*	SOF_{14}	Start of frame, differential progressive, arithmetic coding
FFCF*	SOF_{15}	Start of frame, differential lossless, arithmetic coding
FFDA	SOS	Start of scan
FFDB	DQT	Define quantization tables
FFDC*	DNL	Define number of lines
FFDD	DRI	Define restart interval
FFDE*	DHP	Define hierarchical progression
FFDF*	EXP	Expand reference components
FFE0-FFEF	APP_0-APP_{15}	Application-specific data
FFFE	COM	Comment
FFF0-FFFD*	JPG_0-JPG_{13}	Reserved
FF02-FFBF*	RES	Reserved

Table2: JPEG markers with data [8].

4. The Conversion Process

Step1: Read the BMP File.

1. Open the BMP File in binary mode.
2. Read the first two bytes and check whether the file is really a bmp file.
3. Read the following header information such as file size, reserved1, reserved2, bitmap offset ,image width, image height, color planes, bits per pixel, compression, size of bitmap, horizontal resolution, vertical resolution, colors, and important colors.

4. Read the color table.
5. Move to start of image data.
6. Allocate space for temporary image for processing.
7. BMP files are backwards so move to end of image.
8. Loop through the image data; starting with the first row and for column one to image width; read blue byte, green byte, red byte and advance three bytes. Repeat for rows up to image length.
9. Close file.
10. Return pointer to temporary image.

Step2: Perform Compression.

1. The representation of the colors in the image is converted from RGB to YC_BC_R consisting of one luma component (Y'), representing brightness, and two chroma components, (C_B and C_R), representing color.
2. The resolution of the chroma data is reduced, usually by a factor of 2. This reflects the fact that the eye is less sensitive to fine color details than to fine brightness details.
3. The image is split into blocks of 8×8 pixels, and for each block, each of the Y, C_B, and C_R data undergoes a discrete cosine transform (DCT).
4. The amplitudes of the frequency components are quantized . Human vision is much more sensitive to small variations in color or brightness over large areas than to the strength of high-frequency brightness variations. Therefore, the magnitudes of the high-frequency components are stored with a lower accuracy than the low-frequency components. The quality setting of the encoder (for example 50 or 95 on a scale of 0–100 in the Independent JPEG Group's library) affects to what extent the resolution of each frequency component is reduced. If an excessively low quality setting is used, the high-frequency components are discarded altogether. The detail of quantization is at [7].
5. The resulting data for all 8×8 blocks is further compressed with a lossless algorithm, a variant of Huffman Encoding [9].

Step3: Write the JPEG File.

The JPEG standard is fairly flexible when it comes to the ordering of markers within an output stream. Three major marker ordering restrictions must be followed:

- The file must start with an SOI marker, follow with a JFIF APP0 marker, and end with an EOI marker.
- There can be only one SOF marker and it must occur before any SOS markers
- All Huffman and quantization tables used by a scan must be defined by DHT and DQT markers that come before the SOS marker [8].

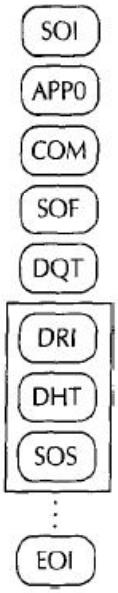


Fig 6: Block ordering of a JPEG file [8].

Figure 6 shows a block ordering for a JPEG file that will work in most situations. The sample JPEG encoder follows this scheme. Encoders that need to store application-specific information need to insert APPn blocks containing this data into the output file.

SOI

The SOI (Start of Image) marker marks the start of a JPEG image. It must be at the very beginning of the file and there can only be one per file. The SOI marker stands alone [8].

APP*n*

The APP0-APP15 markers hold application-specific data. They are used by image-processing applications to hold additional information beyond what is specified by the JPEG standard.

COM

The COM (Comment) marker is used to hold comment strings such as copyright information. Its interpretation is application specific, but a JPEG encoder should assume that the decoder is going to ignore this information. It may appear anywhere within a JPEG file [8].

SOF*n*

The SOF*n* (Start of Frame) marker defines a frame. Although there are many frame types, all have the same format. The SOF marker consists of a fixed header after the marker length followed by a list of structures that define each component used by the frame. The structure of the fixed header is shown in table 3; and the structure of a component definition is shown in table 4. Components are identified by an integer in the range 0 to 255. The JFIF standard is more restrictive and specifies that the components be defined in the order {Y, Cb, Cr} with the identifiers {1, 2, 3} respectively. There can be only one SOF*n* marker per JPEG file and it must precede any SOS markers [8].

Field size	Description
1 byte	Sample precision in bits (can be 8 or 12)
2 bytes	Image height in pixels
2 bytes	Image width in pixels
1 byte	Number of components in the image

Table3: Fixed portion of a SOS marker [8].

Field Size	Description
1 byte	Component identifier. JPEG allows this to be 0 to 255. JFIF restricts it to 1 (Y), 2 (Cb), or 3 (Cr).
1 byte	The 4 high-order bits specify the horizontal sampling for the component. The 4 low-order bits specify the vertical sampling. Either value can be 1, 2, 3, or 4 according to the standard. We do not support values of 3 in our code.
1 byte	The quantization table identifier for the component. Corresponds to the identifier in a DQT marker. Can be 0, 1, 2, or 3.

Table4: Component-specific area of SOS marker [8].

DQT

The DQT (Define Quantization Table) marker defines (or redefines) the quantization tables used in an image. A DQT marker can define multiple quantization tables (up to 4). The quantization table definition follows the markers length field. The DQT marker structure is shown in table5:

Field Size	Description
1 byte	The 4 low-order bits are the table identifier (0, 1, 2, or 3). The 4 high-order bits specify the quanization value size (0 = 1 byte, 1 = 2 bytes).
64 or 128 bytes	64 1- or 2-byte unsigned quantization values

Table5: Quantization Table definition in DQT marker [8].

DRI

The DRI (Define Restart Interval) marker specifies the number of MCUs between restart markers within the compressed data. The value of the 2-byte length field following the marker is always 4. There is only one data field in the marker—a 2-byte value that defines the restart interval. An interval of zero means that restart markers are not used.

DHT

The DHT (Define Huffman Table) marker defines (or redefines) Huffman table. A single DHT marker can define multiple tables; however, baseline mode is limited to two of each type, and progressive and sequential modes are limited to

four. The only restriction on the placement of DHT markers is that if a scan requires a specific table identifier and class, it must have been defined by a DHT marker earlier in a file. The structure of the DHT marker is shown in table 6. Each Huffman table is 17 bytes of fixed data followed by a variable field of up to 256 additional bytes.

The first fixed byte contains the identifier for the table. The next 16 form an array of unsigned 1-byte integers whose elements give the number of Huffman codes for each possible code length (1-16). The sum of the 16 code lengths is the number of values in the Huffman table. The values are 1 byte each and follow, in order of Huffman code, the length counts [8].

Field Size	Description
1 byte	The 4 high-order bits specify the table class. A value of 0 means a DC table, a value of 1 means an AC table. The 4 low-order bits specify the table identifier. This value is 0 or 1 for baseline frames and 0, 1, 2, or 3 for progressive and extended frames.
16 bytes	The count of Huffman codes of length 1 to 16. Each count is stored in 1 byte.
Variable	The 1-byte symbols sorted by Huffman code. The number of symbols is the sum of the 16 code counts.

Table6: DHT Marker Format [8]

SOS

The SOS (Start of Scan) marker marks the beginning of compressed data for a scan in a JPEG stream. Its structure is illustrated in table7. The compressed scan data immediately follows the marker

Field Size	Description
1 byte	Component count
2 component count bytes	Scan component descriptors
1 byte	Spectral selection start (0-63)
1 byte	Spectral selection end (0-63)
1 byte	Successive approximation (two 4-bit fields, each with a value in the range 0-13)

Table7: SOS marker structure [8]

EOI

The EOI (End of Image) marker marks the end of a JPEG image. An EOI marker must be at the end of a JPEG file and there can only be one EOI marker per file and no other markers may follow the EOI marker. The EOI marker stands alone [8].

5. Conclusion

BMP images are quite large and must be converted to a smaller format, such as JPEG, in order to transmit it over a bandwidth-constrained environment such as the Internet. The process of converting an image in BMP format to JPEG mainly involves three steps: Reading the BMP file, performing compression and writing the JPEG file.

6. References

[1] "Encyclopedia of Graphics File Formats," James D. Murray, William Van Ryper, O'Reilly and Associates, 1996.

[2] "The BMP File Format: Part 1," David Charlap, Dr. Dobb's Journal, March 1995.

[3] "The BMP File Format: Part II," David Charlap, Dr. Dobb's Journal, April 1995.

[4] http://cobweb.ecn.purdue.edu/~ace/jpeg-tut/jpegtut1.html

[5] T.81**: Information technology – Digital compression and coding of continuous-tone still images – Requirements and guidelines**

[6] G. K. Wallace, "The JPEG Still Picture Compression Standard", IEEE Transaction On Consumer Electronics, vol.38, Feb 1992.

[7] R. M. Gray, D. L. Neuhoff, "Quantization", IEEE Transactions on Information Theory, Vol. 44, No. 6, 1998.

[8] "Compressed image file formats: JPEG, PNG, GIF, XBM, BMP" by John Miano, July 1999.

[9] www.wikipedia.org/jpeg

[10] http://www.fortunecity.com/skyscraper/windows/364/bmpffrmt.html

[11] William B. Pennebaker and Joan L. Mitchell *JPEG still image data compression standard* (3rd edition) Springer 1993.

[12] N. Ahmed, T. Natrajan, and K. R. Rao, "Discrete Cosine Transform", IEEE Transactions on Computers, vol. 23, July 1989.