Huffman Coding For Digital Photography

Raydhitya Yoseph 13509092
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13509092@std.stei.itb.ac.id
raydhitya.yoseph@gmail.com

Abstract**—***This paper contains explanation of Huffman coding algorithm in encoding, decoding, and its application in JPEG compression. All the explanation prove Huffman coding is useful to achieve smaller file size and storing more files.*

Index Terms**—**coding, digital, Huffman, JPEG, photography.

I. Introduction

There are many wonderful sceneries on the earth. There also many ways of enjoying them, seeing by own eyes is one of the ways. People then started to reproduce the scenery to enjoy the scenery in a medium such as paintings and photographs.

Photography is the art, science, and practice of creating durable images by recoding light or other electromagnectic radiation, either electronicallyby means of and image sendsor or chemically by means of a light-sensitive material such as photographic film.

Photography history started by Aristotle and Euclid who described pinhole camera in the 5th and 4th century BC. During early days of photography, photographs, or generally images, were produced using physics and chemistry. Many processes and inventions were done before in 1884 George Eastman developed technology of film as used by film cameras today.

Traditional photography using films burdened photographers without easy access to processing facilities and competition which pressures photographer to deliver photograph with greater speed for news. As technology improves and information age come into place, digital photography, new form of photography, was being inveted.

In digital photography images is captured using electronic image sensor as a set of electronic data rather than as chemical changes on film. Digital photography made photography easier than before. Everyone uses digital photography to capture their daily life. Nowadays even digital photography can be found inside a mobile phone.

Easier method to take photographs means more clicks being made each second and more photographs being taken each second. This leads to bigger storage medium to store the photographs.

Another concern is a standard. Digital image file is a form of data, or more primitively bits, grouped to a file. That means it needs a standard. So, people can say a file is a digital image file if the mentioned file structure is the same as the standard.

Responding two concerns above various standards and compression methods arise. The most popular and widely used standard is coming from Joint Photographic Expert Group named the same as the group or commonly known as JPEG. This paper will give brief explanation about the standard, Huffman coding, and those two application in digital images

II. Joint Photographic Expert Group

The name "JPEG" stands for Joint Photographic Expert Group, the name of the committe that created the JPEG standard. JPEG is a commonly used method of lossy compression for digital photography.

The JPEG compr ession algorithm is at its best on photographs and painting of realistic scenes with smooth variations of tone and color. As the typical use of JPEG is a lossy compresion method, which somewhat reduces the image fidelity, it should not be used in scenarios where the exact reproduction of the data is required.

The idea of lossy compression is discarding some of the data. In digital images it is based on the idea that human eyes cannot see very detailed color and respon more to the images brightness. For example a photograph of the sky which contains many blue pixels. Each of the pixel actually different in color by only a little. Human eyes cannot see the little difference so by compression the pixels with little difference being made into identical pixels. This compression will not change the perception of humans seeing the images because the will perceive it as the same image, which is a photograph of the sky.

There are several step to compress an image using JPEG standard compression.

1. The representation of the colors in the image is converted from RGB to Y'CbCr, consisting of one luma component (Y'), representing brightness, and two chroma components (Cb and Cr) representing color. This step cometimes skipped.
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 8x8 pixels, and for each block each of the Y, Cb, and Cr data undergoes a discrete cosine transform (DCT). A DCT is similiar to a Fourier transform in the sense that it produces a kind of spatial frequency spectrum.
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 component are stored with a lowe accuracy than the low-frequency components.
5. The resulting data for all 8x8 blocks is further compressed with a lossless algorithm, a variant of Huffman encoding.

III. Huffman Coding

Huffman coding is a very popular coding to represent data with minimum memory needed to store the data. Huffman coding was created to reduce data redundancy in a file. Huffman coding was developed by David A. Huffman while he was a Ph.D., student at MIT, and published in the 1952 paper “A Method for the Construction of Minimum-Redundancy Codes.

A. Character Encoding

One file compromised of many bits to represent its content. That bits is grouped into sets of bits. This grouped bits represent something that understandable in human language. For example the character 'a' is represented by 01000001. Changing the representation of bits into character is called character encoding scheme. Representation of the human readable character is highly dependent at the coding used. For example ASCII and UNICODE have different representation of the character ‘a’ in bits.

B. Statistical Data Redundancy

A character can be inside the file multiple times. For example in the string “sunny”, the data of character 'n' is inside the string twice and therefore stored twice. Storing a character more than once is what we called statistical data redundancy. It is redundant and wasting space to store the same data more than once, even though, the character needs to be more than once to complete the information. This what the Huffman coding accomplishes which is storing an information in a efficient way using minimum memory.

C. Huffman Encoding

The idea of Huffman coding is using the frequency of data inside a file. Data which has the highest frequency being assigned by the shortest bit and data which has the lowest frequency being assigned by the longest bit. This means using variable-length data as opposed to fixed-length data in the raw information. Fixed-length all data have same length whether the data is frequent or not. This will cause an information needs larger space.

Bit assigning is determined by making a tree structure data first. Each node from top to bottom will contain data from the most frequent to the least frequent data. The consensus is left child assigned by '0' and right child by '1'. The tree will continue to span until it reaches a leaf that contains one data. So, a node in Huffman tree will always have two children.

Bit assigning also depends on the amount of unique data. If there are two data, we only need '0' and '1'. If there are three data, we need '0', “10”, and “11”. The more the data are, the longer the bit needed. Compared to 8 bit extended ACII binary code which often used, Huffman coding save many wasted space.

Given a string there are 4 step to do the coding:

1. Count the frequency of each character in the string;
2. Form the Huffman tree based on the character frequency from bottom up using greedy algorithm;
3. Assign each character in the tree according to the convention bit;
4. Convert the given string old bits representation into new bits representation using the Huffman code for each charcter.

An example of above algorithm using following information

Table 1 Huffman Table 1

Symbol	A	B	C	D	E
Frequency	15/39	7/39	6/39	6/39	5/39

Two nodes which have the lowest frequency are D and E. So, those two nodes combined together and D assigned '0' and also E assigned with '1'. The first step will make the table

Table 2 Huffman Table 2

Symbol	A	B	C	DE
Frequency	15/39	7/39	6/39	11/39

Two nodes with the lowest frequency are B and C. Therefore the previous action being done again and will make the table

Table 3 Huffman Table 3

Symbol	A	BC	DE
Frequency	15/39	13/39	11/39

Now, two nodes with the lowest frequency are BC node and DE node. Therefore we make a node by combining those two and will make the table

Table 4 Huffman Table 4

Symbol	A	BCDE
Frequency	15/39	24/39

The last action will combine node A and node BCDE into ABCDE node with the frequency 39/39. All the process will be better if represented in a tree.

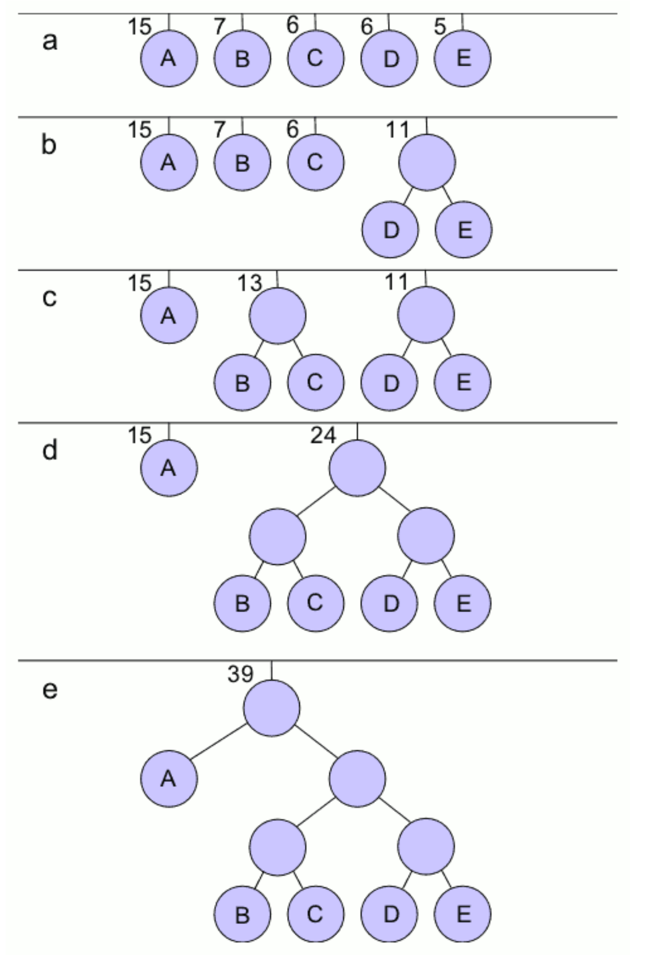


Figure 1 Huffman Tree

The Huffman encoding will given symbol with following code

Table 5 Huffman Encoding Result

Symbol	ASCII Code	Huffman Code
A	0100 0001	0
B	0100 0010	100
C	0100 0011	101
D	0100 0100	110
E	0100 0101	111

If we used fixed-length data, we need 8*39 = 312 bit. If we used variable-length data as the Huffman code, we only need 1*15 + 3*7 + 3*6 + 3*6 + 3*5 = 87 bit. Huffman coding saved 312-87 = 225 bit, which is a huge 72.11% of space. Averagely Huffman coding will save 20% to 30% of space.

D. Huffman Decoding

Above process is called coding which is to convert information into another form of representation, in Huffman code case into a tree structure data. This process is done for the information to get communicated. To recognize the information again, we need the reverse process which is called decoding. Decoding converts the information back into its original representation so it could be recognized by the receiver.

Decoding an information coded with Huffman coding can be achieved by two means. First is to traverse the tree, write the symbol found during the traversal, traverse again from the tree's root. The first method is a modified version of deep first search algorithm. Second is to use a lookup table which contains all symbols and its bit code. To compare the two methods we'll use previous example and try to decode “01100111” into “ADAE”.

1. Traversing Huffman Tree

The idea of the first method is to traverse the Huffman tree bit by bit and write the symbol found. To decode “01100111” we traverse the string and find '0'. Like DFS we begins from root node. In here the implementation is better to implements recursive DFS. We only doing recursive into the child node with coresponding bit with the string. So we are doing recursive into the root's child assigned by '0'. There we find a symbol which is A therefore we write 'A' and back again to the tree's root.

Second traversal we find the bit '1' so we do recursive to the right child from the root node. At the right child we don't find a symbol therefore we traverse into the next bit which is '1' and again move to the right child. The second right child also doesn't contain a symbol so we move to the next bit which is '0'. Moving to the left child will get us a symbol which is 'D'. We write 'D' and again come back to tree's root.

Repeating the process will get us 'A' and 'E' as the next symbol. After the last symbol, there is no next bit so the bit stream “01100111” finished decoded into “ADAE”.

This search is better implemented recursively because we will always find the goal in the branch we are traversing and do not need the other branch.

Notice that we don't record anything from the bit stream we simply traversing it which is different with the second method.

2. Lookup Table

The idea of the second method is to provide a lookup table which contains all the symbols and the bit assigned to that symbol. The process is to traverse only the bit stream not including the Huffman tree.

First we make a lookup table that contains all symbol with their respective bit assignment. The table basically is Table 5. Next we traverse the bit stream and traversing the bit stream will net us '0' as the first bit. We record the bit and then search the table whether there is or not a symbol assigned by the bit '0'. We found 'A' is assigned by '0' in the table so we write the symbol 'A' and can search for the next symbol.

The difference between the first method we only traversing to the next bit and don't come back to the tree's root. The next bit we net is '1', we record it and search the table. There is no symbol assigned by '1' so we traverse the next bit. Again we find '1', we record it which made our current bit stream become “11”. Do the second search with “11” will also net us with nothing. Traversing the next bit our current bit stream become “110”. Searching that will get us the 'D' symbol. We write the symbol and can start to search for the next symbol. Repeating the process will make us decode “01100111” bit stream into “ADAE”. Notice that the second difference between the first method is we record the bit we currently search the symbol for.

Modifying the second method, we could also save the bit assigned to a symbol length. We then can use that information to modify the traversal. Traversing our bit stream again will get us the first '0' and the first 'A'. The next bit will be '1'. Notice that all bit assigned to a symbol beside the symbol 'A' have the length 3. So, when we get a '1' after we found a symbol, we traverse the stream three times while recording each bit. After that, we search the table to find the symbol. Back to the bit stream we will get “110” and can then search the table to find the symbol 'D'. Repeating the process, once again we decode the “01100111” bit stream into “ADAE”.

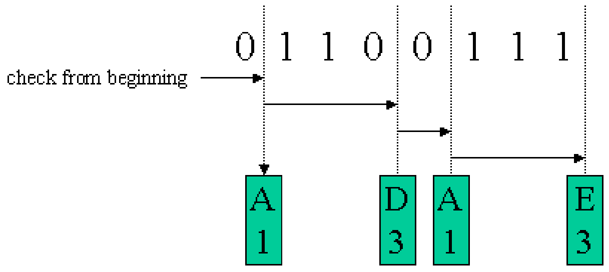


Figure 2 Huffman Decoding

Both code will have their use in different ways. Traversing the bit stream and Huffman tree is more complex than traversing only the bit stream. Searching a table with too many data is also troublesome. In the end what we want to decode that matters. Whether the information is long or not and whether the information is being build by many data or not.

IV. Huffman Coding Inside JPEG

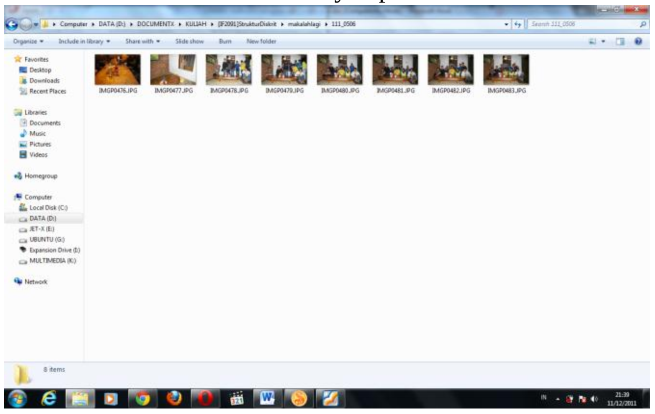
The Huffman encoding is used at the end of JPEG compression. The images went through a lossy compression at first until the quantization. After that, the quantization bits went through a lossless compression method of Huffman encoding.

The Huffman coding result depends on the photographs itself. Photographs with similiar color will have smaller size than photographs with many colors. For example a person photographs will have a smaller size than a scenery photographs.

V. When Size Does Matter

People nowadays has many photographs because of the easiness of taking picture. When size does matter further compression of the photographs can be done using DEFLATE algorithm to make ZIP files. DEFLATE algorithm is a lossless compression using Huffman code.

Here is the files I used for my experiments



Above pictures total size is 24 446 550 bytes. The pictures were similiar. Using DEFLATE algorithm to compress the files, I got 24 392 979 bytes. That is 53 571 bytes less than the original files.

VI. Conclusion

The conclusion of the explanation

1. Huffman coding is used in JPEG compression and useful for digital photography;
2. Huffman coding is useful when there is many data that has big probability in the file;
3. Huffman coding is useful for further compression.

References

[1] http://en.wikipedia.org/wiki/Photography retrieved at 11 December 2011 10:18
[2] http://en.wikipedia.org/wiki/Information_Age retrieved at 11 December 2011 10:58
[3] http://en.wikipedia.org/wiki/JPEG retrieved at 11 December 2011 11:55
[4] http://en.wikipedia.org/wiki/Lossy_compression retrieved at 11 December 2011 12:35
[5] Raydhitya Yoseph, “Greedy And DFS In Huffman Coding”, unpublished.
[6] Raydhitya Yoseph, “Huffman Coding For Your Hearing Pleasure”, unpublished.

DECLARATION

I hereby declare the paper is my own writing, not an adaptation, nor translation from another person paper, and not a form of plagiarism.

Bandung, 11th December 2011

[illegible]

Raydhitya Yoseph 13509092