Combination of Huffman Coding Compression Algorithm and Least Significant Bit Method for Image Hiding

D Rachmawati[1*], A Amalia[1], and J Surya[1]

[1]Department of Computer Science, Faculty of Computer Science and Information Technology, University of Sumatera Utara, Jl. Universitas No. 9-A, Medan 20155, Indonesia.

[*]Email: dian.rachmawati@usu.ac.id, amalia@usu.ac.id, johansurya20871@gmail.com

Abstract. Least Significant Bit method can be used to hide image data into another image, but it need 8 bytes of data to store 1 bit of data, therefore, to store an image with red, green and blue channel. It would require 24 times the size of the image itself. To solve the size problem in hiding data, we implement and experiment with Huffman Coding compression algorithm to reduce the size of the image data needed in C#. Our experimental results are as follows. First, the less colour value variation, Huffman Coding algorithm will give a better compressed size. Second, the average file size change in this research is 29.17%.

1. Introduction
Steganography is one of the most powerful techniques to conceal the existence of hidden secret data inside a cover object [1]. Least Significant Bit steganography is one such technique in which least significant bit of pixels of the image is replaced with data bits. This approach has the advantage that it is simplest one to understand, easy to implement and results in stego-images that contain embedded data as hidden [2]. Images created from pixels i.e. If any pixel created by using these three colors red, green and blue are called as RGB. Each color of a pixel is one byte information that shows the density of that color. So if only last layer of information is used, then the last bits of the pixels has to be changed, in other hands we have 3 bits in each pixel so we have 3*height*width bits memory to write our information [5].

Compression refers to reducing the quantity of data used to represent a file, image or video content without excessively reducing the quality of the original data. It also reduces the number of bits required to store and/or transmit digital media [3]. Image compression can be bifurcated as lossy and lossless compression. Lossy compression as the name implies results in loss of some information. The compressed image is similar to the original uncompressed image but not identical to the previous as in the process of compression some information regarding the image has been lost. They are generally suited for photographs. The most common example of lossy compression is JPEG. Lossless compression compresses the image by encoding all the information from the original file, so when the image is decompressed, it will be exactly identical to the original image. Examples of lossless image compression are PNG and GIF. (GIF only allows 8-bit images). When to use a certain image compression format really depends on what is being compressed [6]. The purpose for image compression is to reduce the amount of data required for representing sampled digital images and therefore reduce the cost for storage and transmission [4].

Huffman coding is a loseless data compression technique. Huffman coding is based on the frequency of occurrence of a data item i.e. pixel in images. The technique is to use a lower number of bits to encode the data in to binary codes that occurs more frequently [3].

2. Method

The experiments are conducted on the Windows 7 Notebook which has Intel Core i3 processor with 32-bit architecture and 2048MB RAM. The development environment being used for coding C# scripts is SharpDevelop.

The image hiding process began with the secret image being compressed before insert into cover image. The compression process will create the Huffman Tree at first.
Steps for creating Huffman Tree:

1. Get all the frequency of all color value from the secret image.
2. Select the two lowest frequency color value from it and treat it as the first node of Huffman Tree.
3. Create parent node from those two nodes and count the sum of their frequency.
4. Remove those two nodes and replace with the parent node. It will be used to create the tree.
5. Do the steps above continuously until there is only one node left.
6. Every node that place at left branch will be given 0 value and the right branch will be given 1 value.
7. Read from the root of the tree to all nodes by checking their branch.

The Huffman Code is created from the Huffman Tree and will be use to encode all the secret image data. This compression result will be divided into 3 parts. The first part is the bits that contain information about the image size, the Huffman Code bit length variation which will be used to read the code length and its frequency that will be used to read the Huffman Code table in the second part. The second part is the bits that contain the pixel value and its Huffman Code. This part will be used to decode all the compressed data in the third part. The third part is the bits that contain all the image pixels value that have been encoded with Huffman Code. Those merged part will be insert into cover image by using Least Significant Bit method to produce stego-image. Because the Least Significant Bit method only replace the last bit, then the steps to this method is as follow:

1. If the bit value that need to store is 1 and the color value modulo 2 is 0, then increase the color value by 1
2. If the bit value that need to store is 0 and the color value modulo 2 is 1, then decrease the color value by 1
3. If the bit value that need to store is same with the color value modulo 2, then skip to the next bit that need to store.

The image recovery process began with the stego-image being extracted bit by bit by using Least Significant Bit method. In each extraction process, the system will check the extracted bits for decompression. If one of the decompression process failed, then the whole recovery process will be failed. If the decompression process has produced an image, the recovery process will be success and sending the decompressed image as the result.

3. Results and Discussions

The results of the experiments of each set are presented in Table 1 and Table 2 as follows.

Table 1. The Compression result for 5 sample images.

Image	Color Value Variation (0-255)	Original (Bit)	Compressed (Bit)	Compressed Ratio (%)	Compression Time (ms)
All Black.bmp	1	2400	365	15.21	1
BlackWhite.bmp	2	2400	374	15.58	1
BlackWhiteGray.bmp	3	2400	503	20.96	1
_sicon.bmp	195	24576	17878	72.75	3
woman.bmp	199	98304	28826	29.32	9

In Table 1, it can be seen that the average compressed ratio is 30.76% and the average compression time is 3ms.

Table 2. File size changes for 10 sample images.

Image	File Size	File Size (After inserted)	Change (%)
All Black.bmp	376 bytes	454 bytes	20.74
BlackWhite.bmp	376 bytes	454 bytes	20.74
BlackWhiteGray.bmp	376 bytes	454 bytes	20.74
fivepointstar.bmp	3.05 KB	4.05 KB	32.79
Airplane.bmp	6.80 KB	8.86 KB	30.29
woman.bmp	12.0 KB	16 KB	33.33
_sicon.bmp	3.05 KB	4.05 KB	32.79
brick.bmp	175 KB	234 KB	33.71
Core2Quad.bmp	670 KB	893 KB	33.28
Intel-X38.bmp	597 KB	796 KB	33.33

In Table 2, it can be seen that the average file size change is 29.17%. The file size change showed in Figure 1.

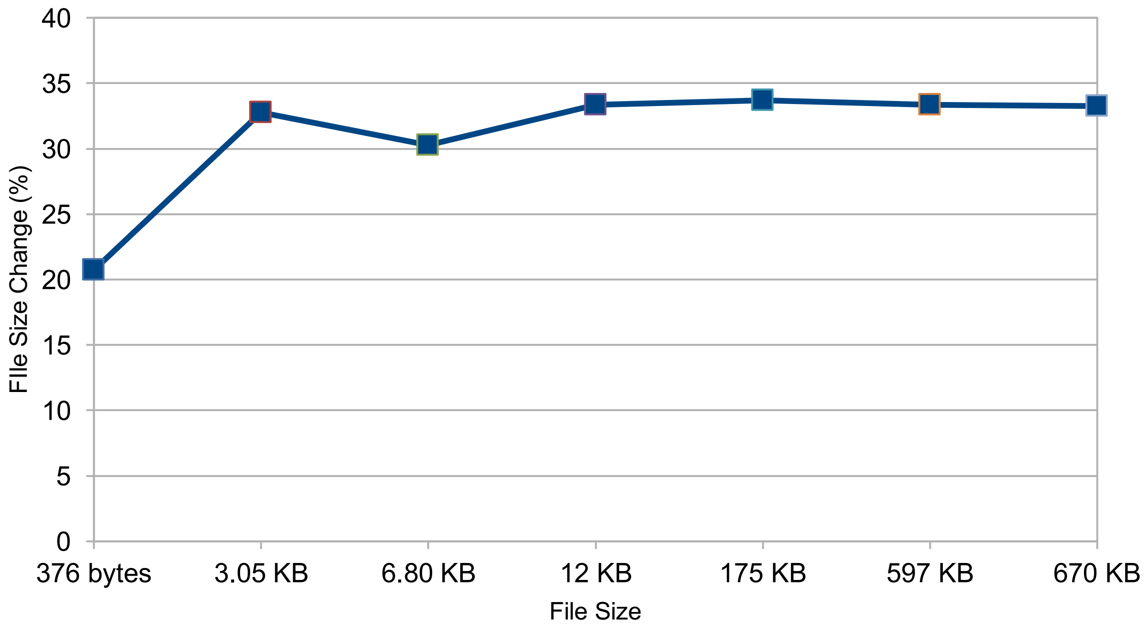

Figure 1. File Size Change

4. Conclusions

The conclusion in this research are as follows. First, the less the color value variation, Huffman Coding algorithm will give a better compressed size. Second, the average file size change in this research is 29.17%.

Acknowledgments

The authors gratefully acknowledge that the present research is supported by Lembaga Penelitian Universitas Sumatera Utara.

References

[1] Champakamala B S, Padmini K and Radhika D K Least Significant Bit algorithm for image steganography *International Journal of Advance Computer Technology (IJACT)* **3** (4) 34-38

[2] Verma V, Poonam and Chawla R 2014 An Enhanced Least Significant Bit Steganography Method Using Midpoint Circle Approach *International Conference on Communication and Signal Processing* 105 -108

[3] Sharma M 2010 Compression Using Huffman Coding *IJCSNS International Journal of Computer Science and Network Security* **10** (5) 133 – 141

[4] Pujar J H and Kadlaskar L M 2010 A New Lossless method of Image Compression and Decompression Using Huffman Coding Techniques *Journal of Theoretical and Applied Information Technology* 18-22

[5] Patel K, Utareja S, Gupta H 2013 Information Hiding Using Least Significant Bit Steganography and Blowfish Algorithm *International Journal of Computer Applications* **63** (13) 24 – 28

[6] Mathur MK, Loonker S, Saxena D 2012 Lossless Huffman Coding Technique For Image Compression And Reconstruction Using Binary Trees *International Journal of Computer Technology and Applications* **3** (1) 76 – 79