An Implementation of Text Hiding in Medical Images based on Graph Coloring for Android Devices

Muhammad Wildan Iskandar¹, Adiwijaya²
¹,² School of Computing, Telkom University

¹mwildanis@student.telkomuniversity.ac.id, ²adiwijaya@telkomuniversity.ac.id

Abstract. Medical images are representations of highly vulnerable digital images and useful objects that help diagnose a patient. However, it is not easy to maintain the authenticity of a medical image because there are many ways that it can be manipulated. Therefore, several methods have been developed to protect the authenticity of medical images. One way is to add a watermark onto the medical images. However, digital image watermarking techniques are usually widely implemented using software running on a computer. In this paper, the authors built a system to hide the text on medical images using an Android device. The system consists of an embedding and extracting process. In the embedding process, the method used is vector quantization and graph coloring based on a Genetic Algorithm. The test result shows that the scheme has good imperceptibility and a fast enough time consumption to be implemented on an Android device.

1. Introduction

The digital image has many uses in life, one of which is in the field of medicine. Medical images are representations of highly vulnerable digital images and useful objects that help diagnose a patient’s condition. Medical devices can be used to obtain these medical images, one of which is the Ultrasound image or via Ultrasonography (USG) [1]. The authenticity of the medical image is very important because it will have an impact on the diagnosis of a patient’s disease. However, it is not easy to retain the authenticity of the medical image because it can be easily manipulated in many ways, one of which is with the help of software [1]. Therefore, medical images must have copyright information embedded on it.

To maintain the authenticity of medical images, the researchers developed a method for entering information onto images known as image watermarking. Digital image watermarking is a technique of inserting certain information onto a digital image. This inserted information is called a watermark. This watermark can take the form of text, image, audio, or even video [2]. At present, digital image watermarking techniques have been widely implemented using software running on a computer. With the growing development of mobile technology nowadays, especially Android OS, it is possible that digital watermarking be done using an Android device. This is because Android devices can easily be used at any time or place.

In this paper, the authors built a system to hide the text on medical images using an Android device. The system consists of an embedding and extracting process. In the embedding process, the method used is vector quantization and graph coloring based on a Genetic Algorithm (GA). The testing was done to determine the value of Peak Signal to Noise Ratio (PSNR) between the watermarked medical images and original medical images. In addition to testing imperceptibility, the watermarked medical images were also tested for robustness via determination of the
percentage of Bit Error Rate (BER) \textsuperscript{[3]}. The author also tested the accuracy of the extracted text before the watermarked medical image was manipulated.

2. Related works

Many studies have developed methods for digital image watermarking. Adiwijaya \textit{et al.} \textsuperscript{[4]} implemented steganography to embed medical records onto medical images. The research itself provided a method that leveraged on a data hiding scheme using Vector Quantization and graph coloring based on evolutionary algorithms, i.e. the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). The result of the research showed that GA provided better performance than PSO in terms of computational time, but both algorithms yielded approximately the same performance in terms of hiding capacity.

Adiwijaya \textit{et al.} \textsuperscript{[1]} discussed a scheme for a reversible watermark using a modified LSB and Huffman compression to detect and recover a manipulated medical image. The testing results showed that the system was capable of detecting an attack with up to 100% accuracy and could also perform recovery with recovery rate of up to 98% accuracy for some attacks.

Ari Muzakir \textsuperscript{[5]} implemented watermarking using a Least Significant Bit (LSB) technique on an Android device. The research utilized an existing RGB channel in the image and then converted each channel to 8 bits. After that, every character of the existing message was converted into 8 bits based on ASCII code. The results of the simulations carried out using an Android smartphone showed that the LSB watermarking technique was not able to be seen by the naked human eye, meaning that there was no significant difference in the image of the original files with the images that had been watermarked.

Mohammad Farukh Hashmi \textit{et al.} \textsuperscript{[6]} proposed real-time copyright protection and implementation of image and video processing on Android and embedded platforms. In the system, pre-specified copyright information was embedded directly on pictures as they were taken. The system also implemented real-time video processing such as Canny Edge Detection, Dilatation, and Successive frame subtraction on Android and embedded platforms. The results showed that the quality of the watermarked image had a PSNR value exceeding the minimum standard of PSNR.

Raffaele Pizzolante \textit{et al.} \textsuperscript{[7]} developed a tool for portable devices based on Android OS, that permits the addition of a visible or invisible watermark to images. The implemented algorithm for the visible watermark converted the string of the watermark into a bitmap image, and then merged the bitmap obtained with the input image in a random position. Meanwhile, the algorithm used for the invisible watermark was a modified version of labeling techniques. Testing was done to determine the robustness of the watermarked image under JPEG compression. The testing result showed that there was a percentage error obtained from the extraction process when the compression factor of the JPEG algorithm was low (i.e. 50% and 30%); however, it was still possible to easily distinguish the string of the watermark.

3. System design

Before proceeding with the discussion on system design, the following section describes the characteristics of the data used in this paper. The data used was 8-bit grayscale ultrasound images with 1024x1024 pixel resolution in JPEG format, making up to as many as 15 images. The data was obtained from Dr. Wahidin Sudiro Husodo hospital. Figure 1 shows an example of the data used in this paper. The text data used is a string.
Overall, the system was designed as part of two schemes; the first is the embedding scheme, which aims to insert messages into medical images. In this scheme, there are two inputs, which are the medical records of the patient and the medical images with the resulting output of medical images that contain secret data. To increase hiding capacity, secret text data was compressed using a Huffman Algorithm \cite{4}. The flowchart of the embedding scheme is described in figure 2.

\textbf{3.1 Embedding process}

The embedding process includes processes such as ROI and RONI separation, codebook generation, graph construction, graph coloring, graph refining, and quantization. Figure 3 illustrates the embedding process.

The first step in the embedding process is ROI and RONI separation. To avoid misdiagnosis, medical images were divided into two parts: Region of Interest (ROI) and Region of Non-Interest (RONI). ROI is a sensitive part in a medical image, therefore the text cannot be inserted into the ROI section because it can change the shape of the original medical image \cite{8}. To divide the ROI and RONI in a medical image, cropping was performed. Figures 4-6 below show the division of RONI and ROI in a medical image.

After ROI and RONI separation, a process called codebook generation was performed. The codebook is a set of codewords \cite{9}. To build a codebook, the image was divided into several blocks, where each block is called a training vector. The training vectors then undergo a clustering process \cite{8}. In this paper, the algorithm used to build the codebook is the Linde Buzo Gray (LBG) algorithm \cite{10}.

The next stage is graph construction. A graph was constructed by adding a vertex $v$ for each codeword $cw$ in the codebook. To determine whether or not there is an edge between the vertices $v[i]$ and $v[j]$, which represent $cw[i]$ and $cw[j]$, respectively, the scheme checked the Euclidean Distance between $cw[i]$ and $cw[j]$. If the Euclidean Distance did not exceed the constant coefficient of the adjacency threshold, the edge between $v[i]$ and $v[j]$ would be added \cite{4}. 
Figure 3. Embedding process flowchart.

Figure 4. A medical image.

Figure 5. RONI of the medical image of figure 4.

Figure 6. ROI of the medical image of figure 4.
After the graph is created, the next process is graph coloring. The graph was colored with a number of colors called \textit{color\_number}. In this paper, each color was represented by an integer, such that the colors = \{0,1,2,...,\textit{color\_number}-1\} \cite{8}. The value of the \textit{color\_number} must satisfy $2^x$ where \(x\) is the natural number. Each color corresponds with a \(x\)-bits secret message. For example, if \textit{color\_number} = 8, then color = \{0,1,2,3,4,5,6,7\} = \{000,001,010,011,100,101,110,111\}. Since graph coloring is a NP-complete problem and the Genetic Algorithm provides better performance than Particle Swarm Optimization \cite{4}, this paper utilized the Genetic Algorithm \cite{11} to color the graph.

The next process is refining the graph. This was done to reduce any failure during the quantization process \cite{4}. The step for refining the graph was done for every vertex \(v\) present in the graph, utilizing a Breadth First Search to determine whether it was within a certain range from \(v\) or \(bfs\_thresh\) from \(v\) in which all colors could be found \cite{4}. If there is a color that cannot be found, \(v\) is removed from the graph.

The last step in the embedding process is final quantization. In the first step, RONI was divided into 4x4 blocks; the quantization process started from the first block. Next, in the iteration step, the most similar codeword \(cw\) to the current block was determined using Euclidean Distance. Then, the vertex \(v\) from the graph that corresponds to codeword \(cw\) was determined. If \(v\) was found, from secret bit data, \(b\) bit was taken as \(x\), where \(2 = \textit{color\_number}\). If \(v\) was not found, the codeword \(cw\) that corresponds with \(v\) would be the output. In the next step, Breadth First Search was used for vertex \(v'\) which is connected with \(v\) and a satisfactory color \((v') = x\) and the scheme search codeword \(cw'\) corresponding with vertex \(v'\), where state \(cw'\) is the output. If the current block is the last block, the quantization process is ended. If the current block is not the last block, the next block is moved as the current block and the iteration step is initiated.

### 3.2 Extracting process

The second scheme is extraction. In this scheme, the required inputs are codebooks and the colored graph, which was created during the embedding process. In the first step, RONI was divided into 4x4 blocks. The extracting process starts from the first block. Next, in the iteration step, the most similar codeword \(cw\) to the current block was found using Euclidean Distance. Then, vertex \(v\) from the graph that corresponds to codeword \(cw\) was determined. If \(v\) is found, color(\(v\)) will be the secret bit data \cite{12}. If the current block is the last block, the extracting process is ended. If the current block is not the last block, the next block is moved as the current block and the iteration step is initiated.

### 4. Result and discussion

In this paper, the author conducted 3 tests to determine the quality of imperceptibility, accuracy of the extracted text, and robustness. The system was made in the form of an application that runs on an Android platform. The device used for testing ran on an Android 7.0 (Nougat) operating system, with 3GB RAM capacity, and Octa-core 1.6 GHz Cortex-A53 CPU. The parameters used in this test are a codebook with size 128, \textit{color\_number} with 8 colors, \textit{adj\_thresh} with range value from 20–60, and \textit{bfs\_thresh} with a value of 150. The metric used to test robustness was BER. First, the watermarked RONI was extracted without attack, to determine the accuracy of the extracted text. Then, the watermarked RONI was attacked using sharpening, Gaussian noise, median filter, and salt and pepper noise attacks, to determine its robustness. All types of attacks were implemented using OpenCV for Android libraries \cite{13}.
Table 1. Results of PSNR watermarked image and running time.

| Image Name  | PSNR   | Codebook Generation | Graph Coloring | Final Quantization |
|-------------|--------|---------------------|----------------|--------------------|
| sample_1.jpg | 47,92 dB | 20.30               | 8.34           | 0.80              |
| sample_2.jpg | 47,29 dB | 23.05               | 5.56           | 0.79              |
| sample_3.jpg | 51.1 dB  | 22.49               | 70.82          | 0.79              |
| sample_4.jpg | 57.85 dB | 21.46               | 1.33           | 0.80              |
| sample_5.jpg | 55.94 dB | 27.31               | 75.53          | 0.86              |
| sample_6.jpg | 62.66 dB | 25.84               | 1.40           | 0.93              |
| sample_7.jpg | 58.45 dB | 20.75               | 12.46          | 0.85              |
| sample_8.jpg | 63.65 dB | 24.50               | 137.98         | 0.93              |
| sample_9.jpg | 64.7 dB  | 21.06               | 15.07          | 0.89              |
| sample_10.jpg | 63.02 dB | 17.97               | 99.21          | 0.81              |
| sample_11.jpg | 54.4 dB  | 20.65               | 25.26          | 0.89              |
| sample_12.jpg | 47.49 dB | 19.08               | 50.78          | 0.82              |
| sample_13.jpg | 55.85 dB | 23.30               | 56.32          | 0.82              |
| sample_14.jpg | 53.58 dB | 25.36               | 7.53           | 0.84              |
| sample_15.jpg | 46.5 dB  | 20.30               | 0.62           | 0.81              |

Average 55.36 dB 20.83 35.52 0.79

Based on table 1, the average PSNR value of all medical images was found to exceed the standard PSNR of 30 dB [3]. The value of PSNR exceeded the minimum standard because of the embedding scheme was based on vector quantization. Vector quantization is an image compression method that manipulates the fact that, in a large block of similar images [12], the watermarked medical image will not be very different from the original medical image. In addition to imperceptibility testing, the author also included the running time of each complicated process. From the codebook-generating process, the average running time was 20.83 seconds; this time will increase with an increase in codebook size. For graph coloring, average running time was 35.52 seconds. However, when testing the sample_8 image, the time required for coloring the graph was 137.98 seconds, while the time for graph coloring sample_15 was only 0.62 seconds. This is because the GA algorithm sometimes showed very good performance, so the initialization of color on the graph and the color created from the initialization would be directly valid. Then, the last step was determining the running time during the process of quantization, where average running time was found to be 0.79 seconds. The time required was short because during the quantization process, all secret data bits had been inserted into the picture. However, if the current image block is not the last image block, then the quantization process will be stopped. The running time will then increase with increased length of secret bit data.

Table 2. The accuracy of the extracted text and extracting time.

| Image Name  | Accuracy | Extracting time (s) |
|-------------|----------|---------------------|
| sample_1.jpg | 100 %    | 3.62                |
| sample_2.jpg | 100 %    | 3.29                |
| sample_3.jpg | 100 %    | 3.43                |
| sample_4.jpg | 100 %    | 3.68                |
| sample_5.jpg | 100 %    | 3.55                |
| sample_6.jpg | 100 %    | 3.67                |
| sample_7.jpg | 100 %    | 4.57                |
Based on table 2, the average accuracy was found to be 100%. This indicates that the text extracted from all the watermarked medical images is the same as the original text. This can be obtained when the watermarked RONI has not been manipulated. Since the codebook and graph used during the extraction process were the same as those used during the embedding process, the extracted text was able to achieve 100% accuracy. The average extracting time was 3.57 seconds. This average time is fast because, during extraction, when the length of secret bit data that had been extracted is the same as the length of original secret bit data, the extraction process will be stopped. The extracting time will increase with increased length of secret bit data.

Table 3. Average test result of BER and PSNR against attack.

| Type of Attack       | Attack Value | PSNR      | Bit Error Rate |
|----------------------|--------------|-----------|----------------|
|                      |              |           |                |
| Sharpening           | 0.5          | 42.21 dB  | 16.94 %        |
|                      | 1            | 41.77 dB  | 35.3%          |
|                      | 2            | 40.67 dB  | 55.72%         |
|                      | 3            | 39.35 dB  | 64.46%         |
|                      | 0.0005       | 24.54 dB  | 97.74%         |
|                      | 0.005        | 17.06 dB  | 100%           |
|                      | 0.001        | 22.15 dB  | 98.47%         |
|                      | 0.01         | 15.34 dB  | 10 %           |
|                      | 3 x 3        | 53.1 dB   | 45.17%         |
|                      | 7 x 7        | 39.46 dB  | 67.52%         |
|                      | 9 x 9        | 34.74 dB  | 70.46%         |
|                      | 11 x 11      | 30.96 dB  | 71.68%         |
| Gaussian Noise       | 0.0005       | 24.54 dB  | 97.74%         |
|                      | 0.005        | 17.06 dB  | 100%           |
|                      | 0.001        | 22.15 dB  | 98.47%         |
|                      | 0.01         | 15.34 dB  | 10 %           |
|                      | 3 x 3        | 53.1 dB   | 45.17%         |
|                      | 7 x 7        | 39.46 dB  | 67.52%         |
|                      | 9 x 9        | 34.74 dB  | 70.46%         |
|                      | 11 x 11      | 30.96 dB  | 71.68%         |
|                      | 0.0001       | 37.06 dB  | 0.73%          |
|                      | 0.005        | 30.08 dB  | 4.09%          |
|                      | 0.01         | 27.13 dB  | 6.42%          |
|                      | 0.05         | 20.22 dB  | 31.93%         |
| Median Filter        |              |           |                |
|                      |              |           |                |
| Salt & Pepper Noise  | 0.001        | 37.06 dB  | 0.73%          |
|                      | 0.005        | 30.08 dB  | 4.09%          |
|                      | 0.01         | 27.13 dB  | 6.42%          |
|                      | 0.05         | 20.22 dB  | 31.93%         |
| Average              |              |           | 54.16%         |

Based on table 3, all the executed attacks resulted in average BER values of above 50%. This is because all types of attacks were done to manipulate the image, and this would change the pixel values that exist in the image. Therefore, during the extraction process, an error will emerge in the second step when the search process codeword is most similar to the block of image to be extracted.

5. Conclusions

According to the results and discussion, it can be concluded that the embedding scheme has good imperceptibility because all the watermarked medical images yielded an average PSNR value above the minimum standard. This embedding scheme also showed good accuracy with the
extracted text given that the watermarked medical images had not been manipulated. However, if the watermarked medical images have been manipulated, this embedding scheme would produce an average BER value of above 50%. This indicates that the embedding scheme has weak robustness. In terms of security, the text hidden in the watermarked medical images using this embedding scheme could not be retrieved without first having knowledge of the colored graph. Based on the contributions of this study, the author has shown that the embedding scheme is suitable to implement on Android devices because it has good imperceptibility and a fast enough time consumption. In future work, the author will improve the data insertion method into images that have good imperceptibility and robustness as well as that can be implemented in mobile devices.

References

[1] Adiwijaya, P. N. Faoziyah, F. P. Permana, T. A. B. Wirayuda, U. N. Wisesty, "Tamper Detection and Recovery of Medical Image Watermarking using Modified LSB and Huffman Compression," Second International Conference on Informatics Applications (ICIA), pp. 129-132, 2013.

[2] S.-C. Liew, "Reversible Medical Image Watermarking For Tamper Detection And Recovery With Run Length Encoding Compression," World Academy of Science, Engineering and Technology, pp. 799-803, 2010.

[3] J. S. Vaishali and G. R. Sachin, "Literature Review of Wavelet Based Digital Image," International Journal of Computer Applications, vol. 31, 2011.

[4] W. Astuti and Adiwijaya, "Graph Coloring Based on Evolutionary Algorithms to Support Data Hiding Scheme on Medical Images," Procedia Computer Science, pp. 173 - 177, 2015.

[5] M. Ari and H. Mailan, "Watermarking Techniques Using Least Significant Bit Algorithm for Digital Image Security Standard SolutionBased Android," Scientific Journal of Informatics, vol. 4, 2017.

[6] H. F. Mohammad, S. J. Ronak, K. G. Avinash, "Real Time Copyright Protection and Implementation of Image and Video Processing on Android and Embedded Platforms," Procedia Computer Science, p. 1626 – 1634, 2015.

[7] P. Raffaele and C. Bruno, "Copyright Protection for Images on Mobile Devices," in Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2012.

[8] W. Astuti, Adiwijaya, U. N. Wisesty, "Data Hiding Scheme on Medical Image using Graph Coloring," Journal of Physics: Conference Series, vol. 622, 2015.

[9] T.-C. Lu, "A Survey of VQ Codebook Generation," Journal of Information Hiding, pp. 190-203, 2010.

[10] W.-H. Steeb, in Mathematical Tools in Signal Processing with C++ & Java Simulations, South Africa, World Scientific, 2005, pp. 26-35.

[11] A. Reza and M. Malek, "An Efficient Hierarchical Parallel Genetic Algorithm for Graph Coloring Problem," Proceedings of The 13th annual conference on Genetic and evolutionary computation, pp. 521-528, 2011.

[12] S. Yue, Z.-H. Wang and C.-C. Chang, "An Image Data Hiding Scheme Based on Vector Quantization and Graph Coloring," Recent Advances in Information Hiding and Applications, pp. 1-17, 2013.
[13] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[14] Adiwijaya, Matematika Diskrit dan Aplikasinya, Bandung: Alfabeta, 2016.
[15] P. K. Arup and S. Anup, "An Efficient Codebook Initialization Approach For LBG Algorithm," *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, vol. 1, 2011.