Mobile Application of Video Watermarking using Discrete Cosine Transform on Android Platform

Moch Ginanjar Busiri
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
gbusiri@gmail.com

Rinaldi Munir
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, Indonesia
Rinaldi.munir@itb.ac.id

***Abstract*— Digital watermarking is a way to keep the security of digital data such as videos and images. Steganography is a method to hide message or watermark into data in order to keep it secret and avoid attention from others. In this study, an application of video watermarking using steganography or using Discrete Cosine Transform (DCT) method has been developed. DCT is robust method, watermark still be safe even if it's attacked or edited. DCT is often used in compression of JPEG image with transforming coefficient and then doing quantization to reduce the size. Watermark is embedded into quantized coefficient. Coefficients are divided into three frequency: low frequency, middle frequency, and high frequency. Watermark will be embedded using Koch Zhao algorithm where the embedding is performed on middle frequency. After the watermark is successfully embedded, it will be compared to original video's quality using PSNR parameter. The experiment showed that watermarked videos have PSNR $\geq$ 30 dB. The weakness of DCT is against Gaussian Noise and the strength of DCT is against compression. Application built on Android platform because in the present era smartphone technology with camera has been widely used by people and with video watermarking our video will not be owned by others.**

Keywords—watermark; DCT; Koch Zhao; robust; embedding; extracting; Android;

I. INTRODUCTION

The development and use of digital data in the form of video is growing rapidly with computer. Furthermore, now it is the era that everything becomes very easy to exchange or distribute digital data, especially video whether using internet-based or not. But with the ease of access, making copyright violations or misunderstanding due to the edited video may occur. Therefore, the protection mechanism is required so that the original digital data is not misused by others.

Digital watermarking is one of popular solutions to maintain the security or authenticity of the data.

There are two methods of digital watermarking, fragile watermarking or spatial domain, and robust watermarking or frequency domain [1]. We can embed watermark with spatial domain or frequency domain, but spatial domain is not robust when the digital data is attacked by some noise. So, the proposed method in this paper to keep watermark from any attack and noise is robust watermarking, that is frequency domain.

Discrete Cosine Transform (DCT) is one of method that belongs to robust watermarking. Generally, DCT is often used in JPEG compression to reduce size of image [2]. DCT will produce coefficients to be used as a place of watermark insertion. There are three kinds of frequencies in coefficients generated by DCT, low frequency, middle frequency, and high frequency. The place that used to embed watermark is middle frequency because in this frequency watermark will be safe and its process will not change the object or digital data too much [3]. Koch Zhao algorithm is one of technique that works on DCT method, it will embed watermark in middle frequency in DCT coefficient [4].

Several studies have been conducted on video watermarking using robust or frequency domain method [5][6][7]. The difference is the method that used in those studies. There is a research on video watermarking using Discrete Wavelet Transform (DWT) combined with DCT [5]. Another research used combination of Singular Value Decomposition (SVD), DWT, and Angular transform method [6]. But essentially all of their researches are applications built on Desktop platform. Meanwhile there is almost no research about video watermarking with robust watermarking on Android platform.

II. FUNDAMENTAL CONCEPT

A. Color Representation

There are many color representations in digital data image and video such as RGB, YUV, or YIQ [8]. Those types can be used as place of watermark, but in this paper we will use RGB color representation because Android Studio has library to access that color representation. YUV and YIQ also can be used but need to be converted by some equations and it also needs more process.

B. Discrete Cosine Transform (DCT)

Discrete Cosine Transform was first introduced in 1974 that can be used in the area of digital processing [11]. DCT is very suitable method and often used for compression file JPEG [2]. DCT is a lossy compression scheme where block 8x8 are transformed from spatial domain into cosine signal domain. Block 8x8 after transformed is usually named as coefficient of DCT. Coefficient of DCT is divided into 3 frequency areas. Fig. 1 shows the area of coefficient DCT.

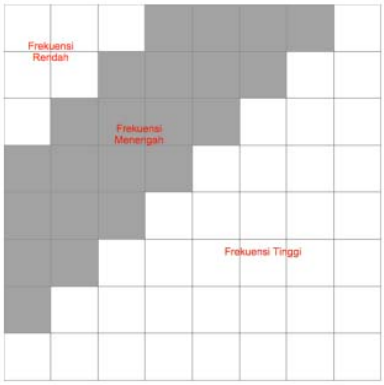

Figure 1. Area Frequency of Coefficient DCT

Low frequencies are most important value. It is located in the top-left corner. If watermark is embedded in this area, the watermark will be robust and safe but quality of video will decrease significantly. High frequencies are least important value and located in bottom-right corner. If watermark is embedded in this area, the quality of video will not change too much but the watermark will be vulnerable if data is attacked [7]. In this paper watermark will be embedded into middle frequency, called Koch Zhao algorithm. In that case it will ensure hiding information in the signal domain that is less essential for the human visual system and the information will not be distorted by JPEG compression with low ratios of compression [12].

C. Android Platform

Android is an operating system developed by Google companies that has open source license so development can be done freely. Android operating system is developed on Linux Kernel that supports various types of electronic devices such as smartphones, tablets, smartwatch, and others [9]. Tools for Android application development is Android Studio. Application development on the Android platform initially only uses Java Native, but now it can be developed using C/C++ language by utilizing NDK (Native Development Kit), and recently there has been a development of React Native. Android has a variety of versions from the first version of Froyo until now the latest is Nougat. The version is very important to know when developing Android application because the latest version sometimes have features and libraries that do not support the previous version.

III. Proposed Method

There are many researches that have been conducted on video watermarking, but the difference is the method used and how they embed watermark into video. Fig. 2 shows overview of system to be used in this paper.

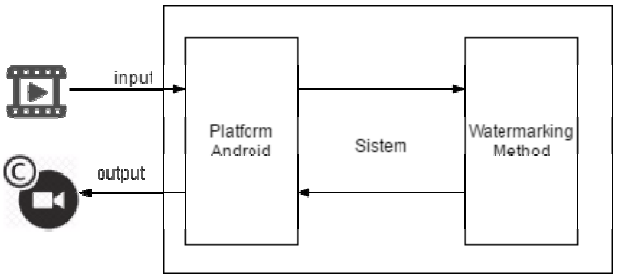

Figure 2. System Description

At first, video file to be embedded by watermark is opened with Android mobile. Then the video will be processed using watermarking method. When the process is complete, the system will return watermarked video. Process watermarking will be explained below.

A. Embedding Algorithm

1) Video Splitting

Video is a collection of images. To embed watermark into video, the first step is to split video into images and access the images one by one. Video is split into frames and audio using library Ffmpeg for Android. Ffmpeg can detect fps from video so when it merges images back it will look like original video.

Video is split into *.png* format images because *.png* is lossless format. It means *.png* format will produce more DCT coefficient than compressed image formats. We can also choose *.jpeg* format when splitting video and it will take fewer memory storage because *.jpeg* is compressed image format [2]. But in *.jpeg* format, it will be hard to find DCT coefficient and it will take a long process time in embedding.

Ffmpeg command to split the video into images can be seen below:

```
“ffmpeg -i VIDEO_LOCATION -r 24 -q:v 1 /storage/emulated/0/images/pic%03d.png”
```

Ffmpeg command to extract audio from video can be seen below:

```
“ffmpeg -i VIDEO_LOCATION -vn -ar 44100 -ac 2 -ab 192 -f mp3 /storage/emulated/0/audio/audio.mp3"
```

2) DCT Transformation

After video is successfully split, the next step is to process images one by one to embed watermark. In the process, get 8x8 pixel block from image because DCT works on matrix with size 8x8 [2]. The block contains value of RGB but in practice we can choose one component of it, for example value of Red. In Android Studio, we can use `Bitmap Android Library` and get the value of RGB with function `getPixel()`. Fig. 3 show the example of getting 8x8 block from image [10].

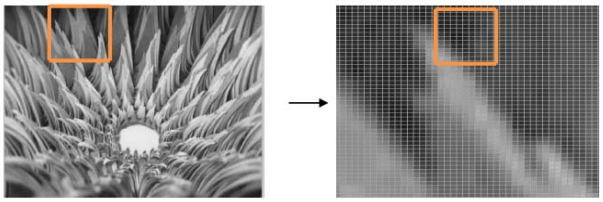

Figure 3. Get Block 8x8 from Image

Assume that block is named original and contains value like matrix below:

$$\text{Original} = \begin{pmatrix} 104 & 101 & 99 & 97 & 95 & 95 & 96 & 99 \\ 101 & 96 & 95 & 95 & 96 & 98 & 99 & 99 \\ 99 & 92 & 91 & 92 & 97 & 102 & 103 & 99 \\ 98 & 90 & 89 & 91 & 98 & 104 & 104 & 98 \\ 98 & 90 & 89 & 92 & 97 & 103 & 104 & 98 \\ 99 & 91 & 90 & 92 & 97 & 102 & 103 & 98 \\ 99 & 92 & 91 & 93 & 97 & 102 & 102 & 98 \\ 100 & 93 & 92 & 94 & 98 & 102 & 102 & 98 \end{pmatrix}$$

At first, subtract original matrix with 128 to convert that into cosinus signal. The result can be seen below as matrix M:

$$\text{M} = \begin{pmatrix} -24 & -27 & -29 & -31 & -33 & -33 & -32 & -29 \\ -27 & -32 & -33 & -33 & -32 & -30 & -29 & -29 \\ -29 & -36 & -37 & -36 & -31 & -36 & -25 & -29 \\ -30 & -38 & -39 & -37 & -30 & -24 & -24 & -30 \\ -30 & -38 & -39 & -36 & -31 & -25 & -24 & -30 \\ -29 & -37 & -38 & -36 & -31 & -26 & -25 & -30 \\ -29 & -36 & -37 & -35 & -31 & 26 & -26 & -30 \\ -28 & -35 & -36 & -34 & -30 & -26 & -26 & -30 \end{pmatrix}$$

After that we can transform matrix M into DCT coefficients with some processes. Transform matrix M using equation below:

$$F(u,v) = \frac{4C(u)C(v)}{n^2} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} f(j,k) \cos\left[\frac{(2j+1)u\pi}{2n}\right] \cos\left[\frac{(2k+1)v\pi}{2n}\right] \quad (1)$$

With:

$$C(w) = \begin{cases} 1/\sqrt{2} & w = 0 \\ 1 & w = 1, 2, \ldots, n-1 \end{cases}$$

And the result is matrix D:

$$\text{D} = \begin{pmatrix} -248.00 & -16.1592 & 11.3996 & 19.2935 & 0.50 & 6.1245 & 2.0431 & 1.5029 \\ 2.2777 & 8.2394 & 1.8936 & -6.5075 & 0.8284 & -1.8108 & 0.0814 & -0.2827 \\ 3.9989 & 11.1516 & 0.6036 & -7.3684 & 1.5772 & -1.8480 & 0.1036 & -0.8600 \\ 0.2381 & 4.3461 & 0.8372 & -2.9047 & 0.6091 & -0.2061 & -0.2008 & -0.4762 \\ 1.0000 & 1.3390 & -0.3266 & -0.1420 & -0.0000 & -0.1829 & -0.1353 & -0.1688 \\ 0.0710 & 0.1907 & 0.1665 & 0.3974 & -0.4070 & -0.5022 & 0.2587 & 0.0355 \\ 0.5084 & 0.2214 & 0.6036 & 0.1661 & -0.1121 & -0.0122 & -0.1036 & 0.3436 \\ 0.0179 & -0.1792 & 0.2387 & -0.3785 & -0.1648 & 0.2781 & -0.1218 & -0.3325 \end{pmatrix}$$

3) Quantization

In JPEG compression, quantization means to reduce image file size depending value of constant used [2]. The higher constant level used, the stronger compression level on file. In watermarking, we will embed watermark on quantized blocks. Quantization process is DCT coefficient matrix divided by constant value and then round it. Equation to calculate quantized block can be seen below:

$$C_{ij} = \text{round}\left(\frac{D_{ij}}{Q_{ij}}\right) \quad (2)$$

Frequently used constant value is constant level 50. Value constant level 50 can be seen below:

$$Q_{50} = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix}$$

The result of quantization process of the matrix D by value constant Q_{50} can be seen below as matrix C:

$$\text{C} = \begin{pmatrix} -15 & -1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

4) Embedding Process

After quantization process and got matrix C, embedding watermark will be performed on middle frequency based on Fig. 1. Generally, Koch Zhao is method of inserting watermark into middle frequency. And technically for its

insertion there are several ways to embed watermark into digital data. In this paper watermark will be embedded using LSB technique.

Before that, watermark is generated using library Random Java on Android to get 100 bitstream 0 and 1. Watermark will be embedded into value that produce not 0 on matrix C, if value is 0 so embedding process on that position will be skipped and continue to next position. For example:

- First watermark bitstream is 0.
- Find value of bit on each pixel using `Integer Java Library` on Android or `Integer.toBinaryString(value)`.
- Based on Fig. 1 middle frequency on matrix C is in position (3, 0) with center (0, 0) in top-left corner. Value in matrix position (3, 0) is 1, and its bit representation is also (1). Another example, if the value is 2, its bit representation is (10), 3 is (11), etc.
- Embed watermark with LSB. Watermark 0 should be embedded with changing bit on value 1. So, value of 1 will change into 2 because in bit representation it will return (10) and watermark 0 successfully embedded.
- Last step is to update new value in matrix C as C' with `Integer Library Interger.parseInt( string_bit, 2)`. Value of 2 means convert from Binary format.

5) Inverse DCT

After watermark successfully embedded, we should be turn it back into value of RGB with Inverse DCT. Before that, matrix C' should be dequantized with constant value with equation below:

$$D'_{i,k} = round\left(Q_{i,k} \cdot C'_{i,k}\right) \quad (3)$$

And then transform matrix D' with equation IDCT below:

$$f(i,k) = \sum_{u=0}^{n-1}\sum_{v=0}^{n-1} C(u)\,C(v)\,F(u,v)\cos\left[\frac{(2i+1)u\pi}{2n}\right]\cos\left[\frac{(2k+1)v\pi}{2n}\right] \quad (4)$$

After apply that equation we will get new value of RGB and then set back to image using `Bitmap Android Library` with function `setPixel()`. Iterate process until all watermark successfully embedded.

6) Combine into Video

The final step in embedding process is to combine all images and audio into video again with the same fps, duration, and resolution. Same as splitting process, this process is done using library Ffmpeg for Android. Command to combine images and audio into video can be seen below:

```
"ffmpeg -f image2 -framerate 24 -i /storage/emulated/0/images/pic%03d.png -i /storage/emulated/0/audio/audio.mp3 -c:v libx264 -profile:v high444 -refs 16 -crf 0 -qp 0 -f mp4 /storage/emulated/0/result/output.mp4"
```

B. Extracting Algorithm

Extracting process is a process to get watermark that already embedded in video. The extraction process steps are not much different with embedding process. First, get 8x8 block from image using `Color Library` on Android and transform into DCT coefficients with eq. 1 then quantize with the same constant that used when embedding, Q_{50}. After quantization, matrix C' will be produced and the value of matrix should be same when embedding process succeed. Get watermark using LSB method on the same position when embedding process based on Fig.1. Iterate all blocks in each image until we get all watermark.

IV. Experimental Results

A. Functional Testing

Three videos with different duration and different resolution have been tested. This test aims to ensure application runs without problems and then compare watermarked video to original video using PSNR parameter. Generally, PSNR equation can be seen below:

$$PSNR = 10\log_{10}\left(\frac{R^2}{MSE}\right) \quad (5)$$

But in library Ffmpeg already provides command to compare and get value of PSNR, so to compare two videos we will use the command below:

```
"ffmpeg -i watermarked_video.mp4 -i original_video.mp4 -filter_complex "psnr" output.mp4"
```

Table 1. Test Case 1

Video 1	
	
Watermark ori	Watermark extracted
01011111100000000001 01111011000000010111 01101110000010111111 00010000001100001111 01110001001011111111	01011111100000000001 01111011000000010111 01101110000010111111 00010000001100001111 01110001001011111111
Duration	00:04
Size	7.72 MB
Resolution	720x720
Time embedding	126.35 s
Time extracting	40.24 s
PSNR	35.561321 dB
Accuracy	100%

Table 2. Test Case 2

Video 2	
	
Watermark ori	Watermark extracted
10011111010100000 01 11011110011111000 10 01000111011101100000 00101011000111011001 10000001000111110001	10011111010100000 01 11011110011111000 10 01000111011101100000 00111111011111110110 11111111011111000000
Duration	00:30
Size	59.6 MB
Resolution	854x480
Time embedding	137.256 s
Time extracting	36.221 s
PSNR	34.590816 dB
Accuracy	80%

Table 3. Test Case 3

Video 3	
	
Watermark ori	Watermark extracted
10101111110101101010 10000101011111011011 00100011011101010111 10001001000111011101 10000110101010000000	10111111110101111010 10001111111111011011 00100011011101010111 10001001000111011101 10000110101010000000
Duration	00:58
Size	25.2 MB
Resolution	240x240
Time embedding	151.152 s
Time extracting	10.156 s
PSNR	31.755172 dB
Accuracy	95%

B. Robustness Testing

Several attacks have been selected to test robustness of watermark. Those attacks are adding object on video, gaussian noise, changing brightness and contrast, cropping, and compression. Table. 4 shows the detail of testing and its accuracy for one test video.

Table 4. Type of Attacks

Type of attack	**Attack rate**	**Result**	**Accuracy**
Adding object	-		100%
Gaussian Noise	20%		51%
Change Brightness	0.1		100%
Change Contrast	1.3		97%
Cropping	-		100%
Compression	crf 5	3.42 MB	97%
Compression	crf 10	2.10 MB	77%
Compression	crf 23	1.22 MB	63%
Compression	crf 30	494 KB	-

Based on functionality testing, we got that watermark has been embedded without any problems and the result getting PSNR $\geq$ 30 dB for all test case videos. It means the videos are still good and the difference is not visible.

Next, based on robustness testing, DCT method is robust for several attacks like adding object, changing brightness, contrast, and cropping because the accuracy is almost 100%. The result of compression attack depending on level constant rate factor (crf), accuracy is still good when crf is less or equal to 10. The weakness of DCT is not robust against gaussian noise because the result shows that at level 20% the accuracy of watermark is 51%.

V. Conclusion And Future Work

The insertion and extraction of watermarks on a video using the DCT method on the Android platform is selecting an 8x8 block RGB on images of the video using Android's library Color. The quality of the watermarked video is considered as good if PSNR reaches $\geq$ 30 dB. The weakness of DCT is against Gaussian Noise and the strength of DCT is against compression.

There are still room for improvements and this research can be developed further. In future works, performance of DCT algorithm could be improved by using Strassen algorithm for DCT transform process. Also, there are bugs that cannot be handled by Android and Ffmpeg Library where value of result matrix produces RGB more than 255 or less than 0. Bitmap Library on Android just handle value 0-255 so if result value is not on its range, information will be lost because the value will be forced to nearest value whether 0 or 255. Solution already found is that we store value on Alpha channel, but unfortunately the Ffmpeg Library doesn't support Alpha channel when combining images into video.

References

[1] Juanda. (2002). *Aplikasi Watermarking Untuk Data Video Digital*. Bandung.

[2] Wallace, G. K. (1991). The JPEG Still Picture Compression Standard. Maynard: *IEEE Transactions on Consumer Electronics*.

[3] Rubel, A. S., & Fedorov, O. (2015). Detection of Hidden Data Embedded by the Koch and Zhao Method. Kharkiv, Ukraine.

[4] Betri, T. J., Suryani, E., & Aziz, A. (2014). Video Watermarking Untuk Perlindungan Hak Cipta Dengan Algoritma Koch Zhao. *JURNAL ITSMART Vol. 3*.

[5] Agarwal, A., Bhadana, R., & Chavan, S. (2011). A Robust Video Watermarking Using DWT and DCT. *International Journal of Computer Science and Information Technologies*.

[6] Mendegar, R. S. (2014). A Robust Video Watermarking Scheme Using 'Arnold'. 'DWT', and 'SVD' Transformation. *International journal of Engineering Research Online*.

[7] Kothari, A. M., & Dwivedi, V. V. (2011). Performance Analysis of Digital Video Watermarking using Discrete Cosine Transform. Rajasthan, India: *Original Scientific Paper*.

[8] Dwiandiyanta, B. Y. (2015). Perbandingan Watermarking Citra Dengan Alihragam Wavelet dan Discrete Cosine Transform. *Simposium Nasional RAPI*.

[9] Bhati, S., Sharma, S., & Singh, K. (2013). Review on Google Android Mobile Platform. *IOSR Journal of Computer Engineering*.

[10] Nugroho, F. A. (2011). Implementasi Teknik Kompresi Video dengan Menggunakan Algoritma Discrete Cosine Transform pada Perangkat Bergerak. Sumatera.

[11] Ahmed, N.; Natarajan, T.; Rao, K. R. (January 1974), "Discrete Cosine Transform", *IEEE Transactions on Computers*, C–23 (1): 90–93

[12] E. Koch, J. Zhao, "Towards Robust and Hidden Image Copyright Labeling", IEEE Workshop on Nonlinear Signal and Image Processing, Greece, pp.123-132, June 20-22, 1995