IMAGE WATERMARKING PROJECT

Introduction

Watermarking is a method employed to insert data or symbols into digital media like images, videos, or documents. These symbols have multiple uses, including safeguarding copyrights, verifying the authenticity of content, and confirming ownership. Watermarks are usually designed to be inconspicuous or partially transparent, ensuring that the original content is not compromised while enabling the identification and tracking of the media's origin.

There are different types of watermarking techniques that can be classified based on their characteristics and applications. Two widely employed methods are Discrete Cosine Transform (DCT) watermarking and Spread Spectrum (SS) watermarking which I used in the project. In DCT watermarking, the watermark information is embedded by manipulating the frequency domain coefficients derived from the DCT transformation. Conversely, SS watermarking spreads the watermark data over a wide frequency range.

I will mention about how I implemented them and what the results are. Firstly, DCT watermarking will be discussed.

Image Watermarking Using Spread Spectrum Transform

When I research on the internet, I have seen methods which work on spatial domain, frequency domain or transform domain. I decided to use firstly spread spectrum watermarking method.

Spread spectrum is a method of transmitting signals over a wider bandwidth than what is required for the transmission. SSP involves various signal processing techniques applied to spread spectrum systems for tasks such as signal modulation, synchronization, interference mitigation, and signal recovery.

The code is in mainssp.m. Code takes firstly original image and watermarking image. It separates each image into 3 images which have rgb values. Therefore, neither original image nor watermark image must not be originally gray scaled. Then, watermarking image sized with same size as original image. Thus, one can put watermark image to different sized image. Both rgb images of original image and watermarking image are normalized. Then the

main part starts with determining random seed. For each original image, sequence images are created with using random numbers.

```
watermarkedImageR = hostImageR + alpha * pnSequenceR .* watermarkImageR;
watermarkedImageG = hostImageG + alpha * pnSequenceG .* watermarkImageG;
watermarkedImageB = hostImageB + alpha * pnSequenceB .* watermarkImageB;
```

At this part the final images are created. At the end, these three images are composed to one image. I have two images which are 'eye.jpg' to use as watermark and 'nature.jpg' to use as original image.

Image 1 Original Image

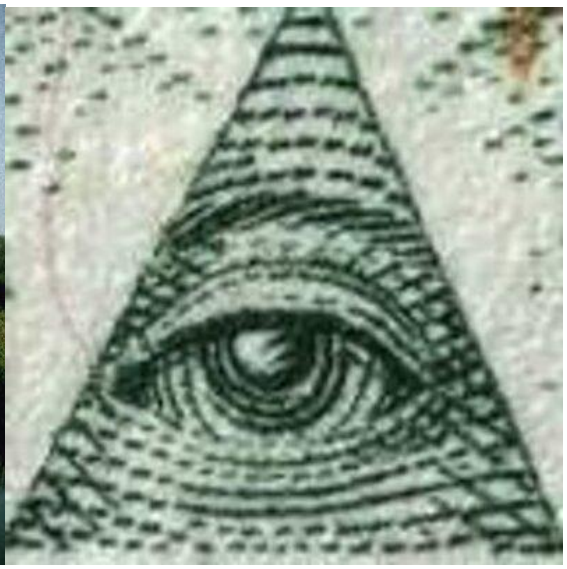


Image 2 Watermark

The watermarked images below for different alpha values.

Image 3 a=0.5

Image 4 a=0.4

Image 5 a=0.3

Image 6 a=0.2

Image 7 a=0.15

Image 8 a=0.1

Actually, we have not known yet these whether these alpha values are ideal but 0.1 has nice resurrection for watermarked image. When a is higher, the image is seen as like one whose eye's streamed while looking at image.

The second part of the code we try to extract what we have embedded. The extracted watermark signals are normalized individually for each color channel. The minimum value is subtracted from each channel, and then the result is divided by the range difference between the maximum and minimum values) of that channel. This step ensures that the extracted watermark values are scaled to the range [0, 1]. To extract the image, we must get original image and have to make adjustment on the alpha and random seed. However, I use same alpha and sequence image values so that I reach the ideal alpha value.

Image 9 Original Image

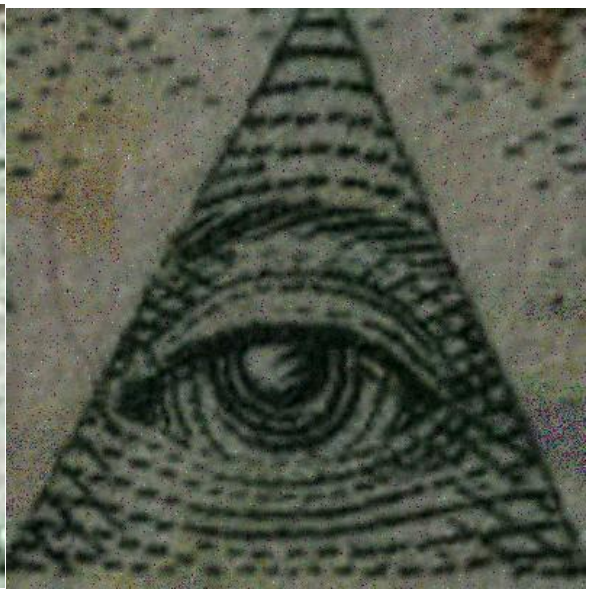


Image 10 a=0.2

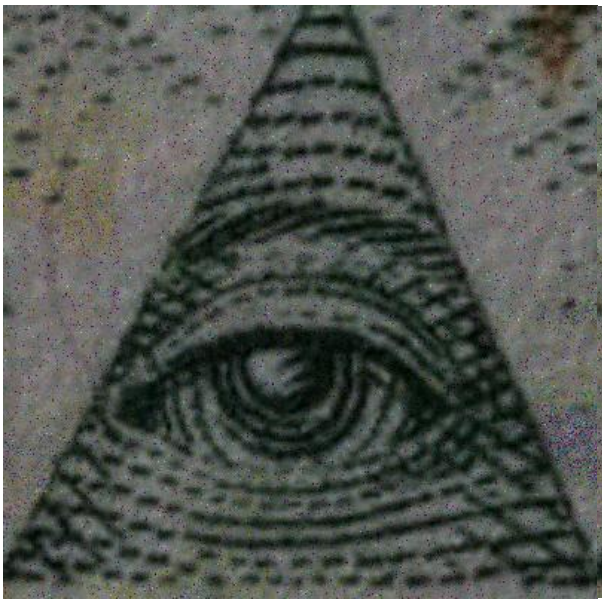


Image 11 a=0.15

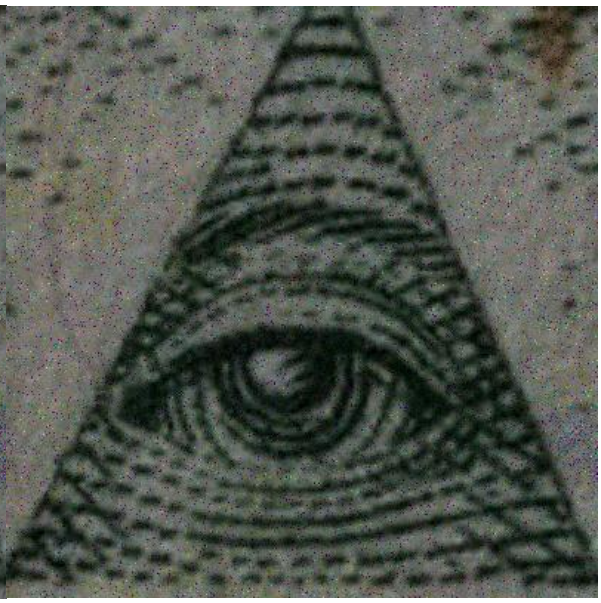


Image 12 a=0.1

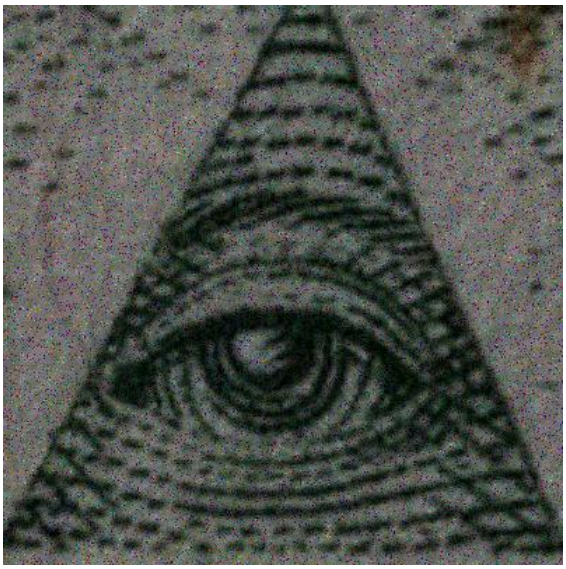


Image 13 a=0.05

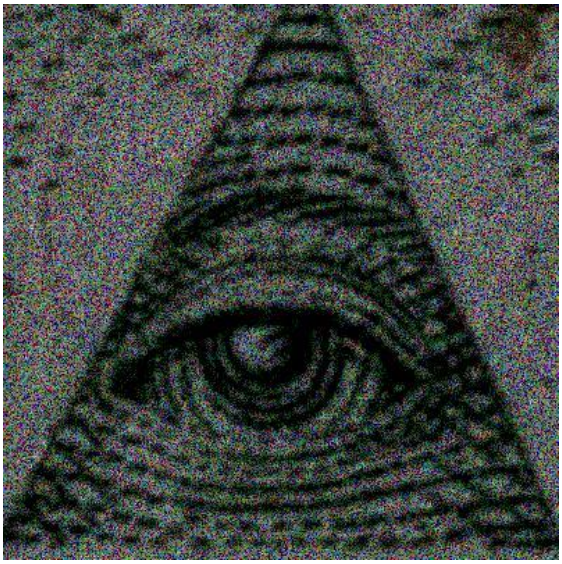


Image 14 a=0.01

As I see from the images, at the right bottom part of the image, while a is lower, the purple dots are less seen until 0.1. Not only in this part of the image, but also in other parts of the images one can see different colors in the image. However when we try do decrease a value more, the image gets worse. A=0.05 value does not have problem but 0.01 makes appeared colorful so many dots. For these two images a=0.05 and 0.1 are ideal values for watermarking these images.

Image 15 Watermarked Image a=0.2

Image 16 Extracted Watermark a=0.2

Image 17 Watermarked Image a=0.1

Image 18 Extracted Watermark a=0.1

Image 19 Watermark a=0.05

Image 20 Extracted Watermark a=0.05

Image 21 Watermark a=0.1

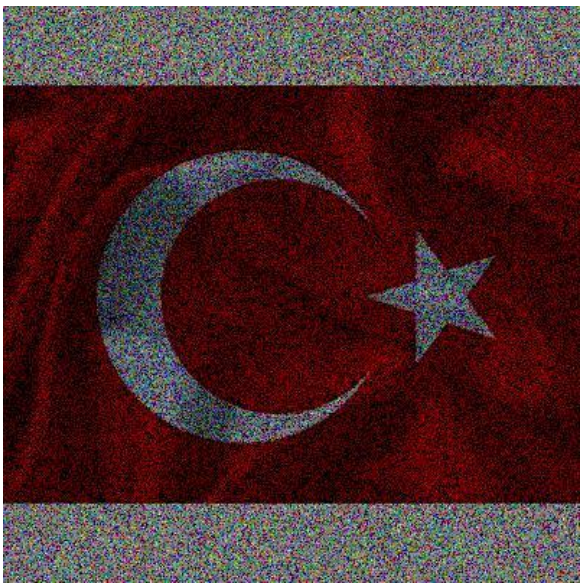


Image 22 Extracted Watermark a=0.1

The ideal alpha value is for these images 0.05 as I inspect the images.

Image Watermarking Using Discrete Cosine Transform

DCT stands for Discrete Cosine Transform. It is a widely used transform technique in signal and image processing. The DCT converts a signal or an image from the spatial domain to the frequency domain, decomposing it into a set of cosine functions with different frequencies.

The code is in maindct.m. Code takes firstly original image and watermarking image. It separates each image into 3 images which have rgb values. Therefore, neither original image nor watermark image must not be originally gray scaled. Then, watermarking image sized with same size as original image. Both rgb images of original image and watermarking image are normalized. Then the main part starts with dct2 function. Each segmented image of original image is transformed by this function. Then, adding watermark images to transformed images is next implementation. At the last, we inversely transform these images and compose to one image.

After watermarking part, I want to extract image. To do that, we need the original image and alpha value except watermarked image. I basically did the inverse of what I did in the watermarking part.

Image 23 Watermark a=0.3

Image 24 Extracted Watermark a=0.3

Image 25 Watermark a=0.2

Image 26 Extracted Watermark a=0.2

Image 27 Watermark a=0.1

Image 28 Extracted Watermark a=0.1

Image 29 Watermark a=0.05

Image 30 Extracted Watermark a=0.05

Image 31 Watermark a=0.01

Image 32 Extracted Watermark a=0.01

Image 33 Watermark a=0.005

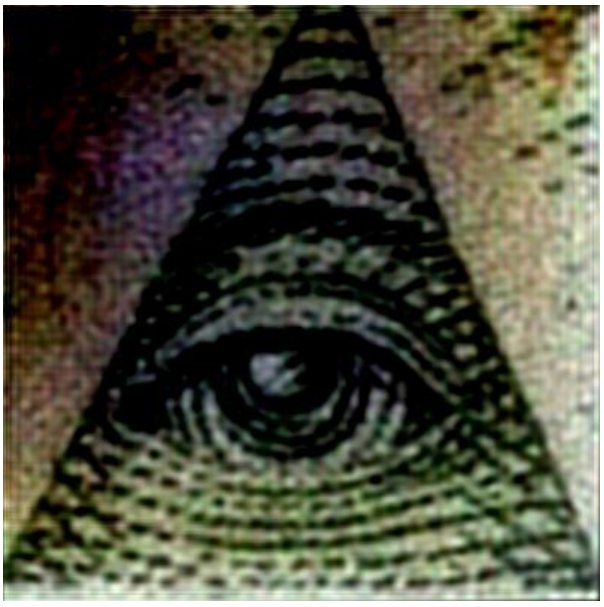


Image 34 Extracted Watermark a=0.005

Image 35 Watermark a=0.001

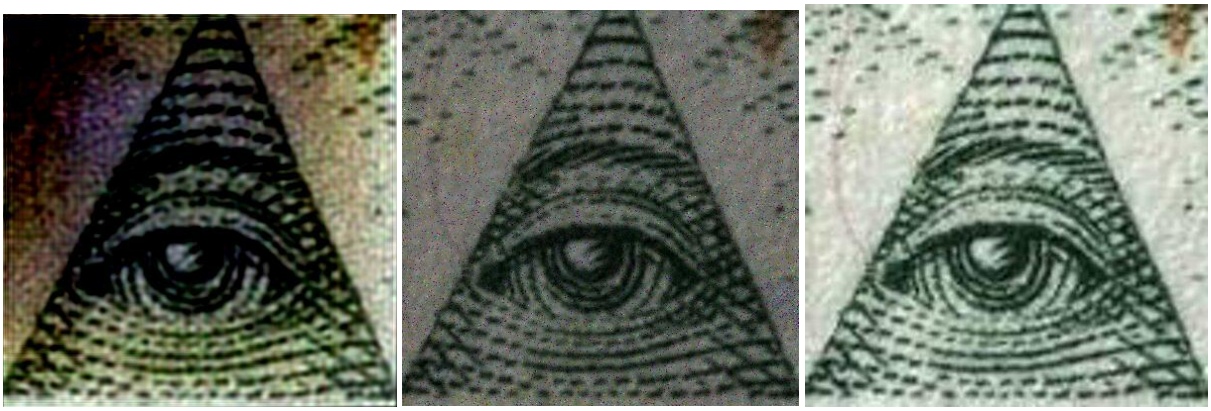
Image 36 Extracted Watermark a=0.001

As I see from the watermarked images, at the left top part of the images is more corrupted while a is higher. Above 0.01 the corruption can be seen clearly. On the other hand examination of other part starts to differ below 0.01. Above 0.01 while the alpha is lowered, the image is lightened more. 0.01 and 0.005 values seems ideal at this part. When we try 0.001, the extracted watermark is much more corrupted than the other images and this clarifies ideal values for the images.

Conclusion

I used two techniques for the watermarking: DCT and SSP. To compare on eye.jpg and nature.jpg I see that, for the ideal values that I determined, dct method is better for original image. As one can see, in watermarked image in ssp, there are many irrelevant dots.

Dct Watermarked *Ssp Watermarked* *Original Image*



Dct Extracted *Ssp Extracted* *Watermark Image*

On the other hand, extracted images are seen better in ssp. Actually I know DCT is successful image watermarking methods as I researched on the net. Therefore, I deduced that the code should be improved to better extraction quality. DCT has good imperceptibility in my work. The watermark is embedded well.

The extracted image quality also depends on sizes. Even if code permits to use different sized images to get better resolution one should use same sized images.

When I use different image on dct, got similar results. The left top part of the extracted image is black, and the other part is OK. However, when the alpha value decreases, the blackness of image decreases but the image has worse resolution as eye.jpg. I used also

boun.png as watermark after trying eye.jpg. boun.png is transparent so when it turns into jpg its background becomes black. I get the best result at this image when a=0.01. This is because the black part less perceptible compared to eye.jpg.

EXTERNAL BENEFICIAL LINKS AND SOURCES

https://www.mdpi.com/2078-2489/11/2/110

https://www.mathworks.com/matlabcentral/fileexchange/53188-simple-spread-spectrum-watermarking-algorthm-in-spatial-domain

https://www.mathworks.com/help/images/ref/dct2.html