IMAGE WATERMARKING USING DCT

Submitted by

Chandreshwar Vishwakarma

21CS06002

M.Tech. (CSE)

Under the supervision of

Dr. Manoranjan Satpathy

Science and Forensic Lab II

Experiment III

School of Electrical Science

INDIAN INSTITUTE OF TECHNOLOGY, BHUBANESHWAR

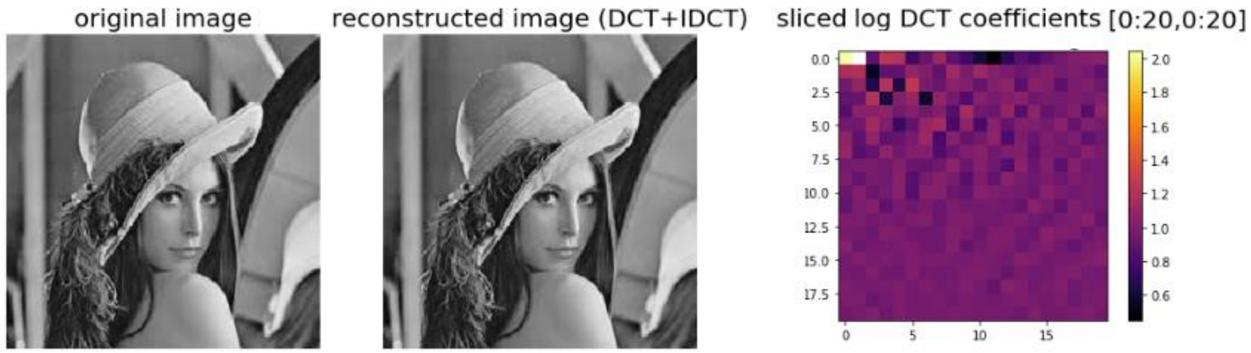
2021

Introduction

Digital watermarking is a technology for embedding various types of information in digital content. In general, information for protecting copyrights and proving the validity of data is embedded as a watermark. A digital watermark is a digital signal or pattern inserted into digital content. The digital content could be a still image, an audio clip, a video clip, a text document, or some form of digital data that the creator or owner would like to protect. The main purpose of the watermark is to identify who the owner of the digital data is, but it can also identify the intended recipient.

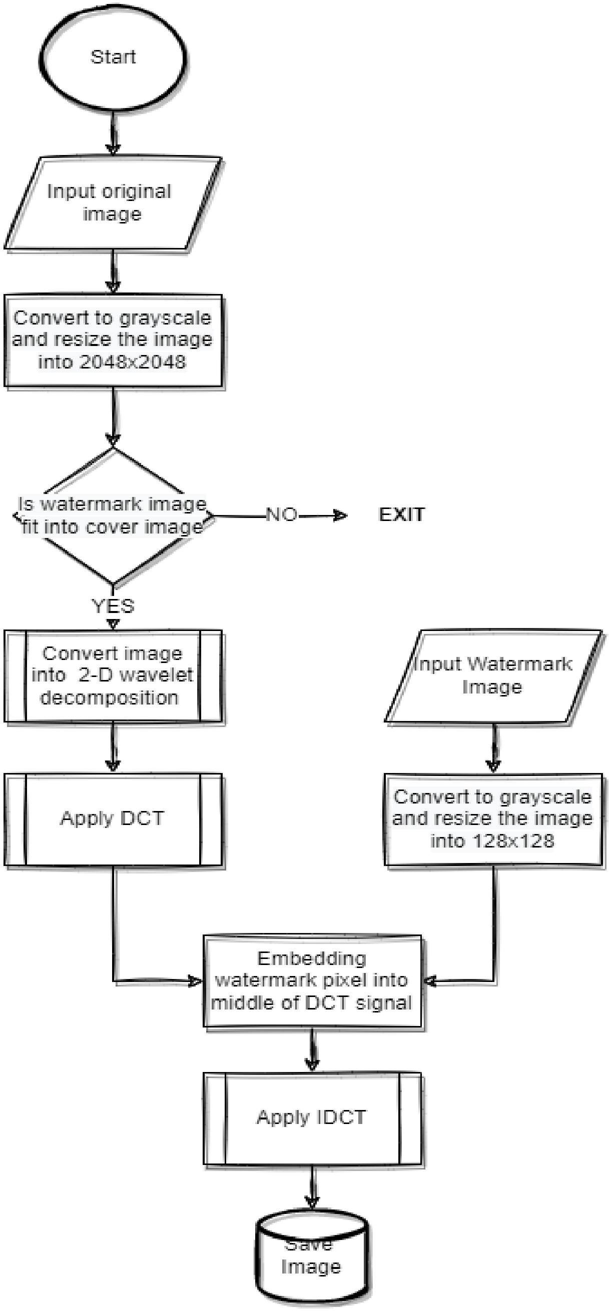
The DCT allows an image to be broken up into different frequency bands, making it much easier to embed watermarking information into the middle frequency bands of an image.

DCT stands for Discrete Cosine Transform. It is a type of fast computing Fourier transform which maps real signals to corresponding values in frequency domain. DCT just works on the real part of the complex signal because most of the real-world signals are real signals with no complex components.

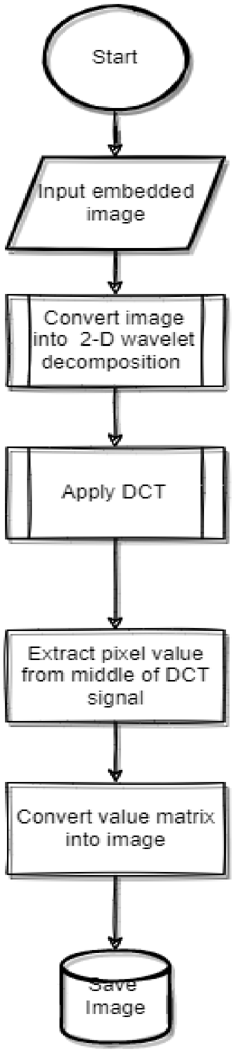


original image reconstructed image (DCT+IDCT) sliced log DCT coefficients [0:20,0:20]

Flow Chart


Start
Input original image
Convert to grayscale and resize the image into 2048x2048
Is watermark image fit into cover image
NO
EXIT
YES
Convert image into 2-D wavelet decomposition
Apply DCT
Input Watermark Image
Convert to grayscale and resize the image into 128x128
Embedding watermark pixel into middle of DCT signal
Apply IDCT
Save Image

Embedding watermark image into cover image


Start
Input embedded image
Convert image into 2-D wavelet decomposition
Apply DCT
Extract pixel value from middle of DCT signal
Convert value matrix into image
Save Image

Extract watermark image from embedded image

Algorithm

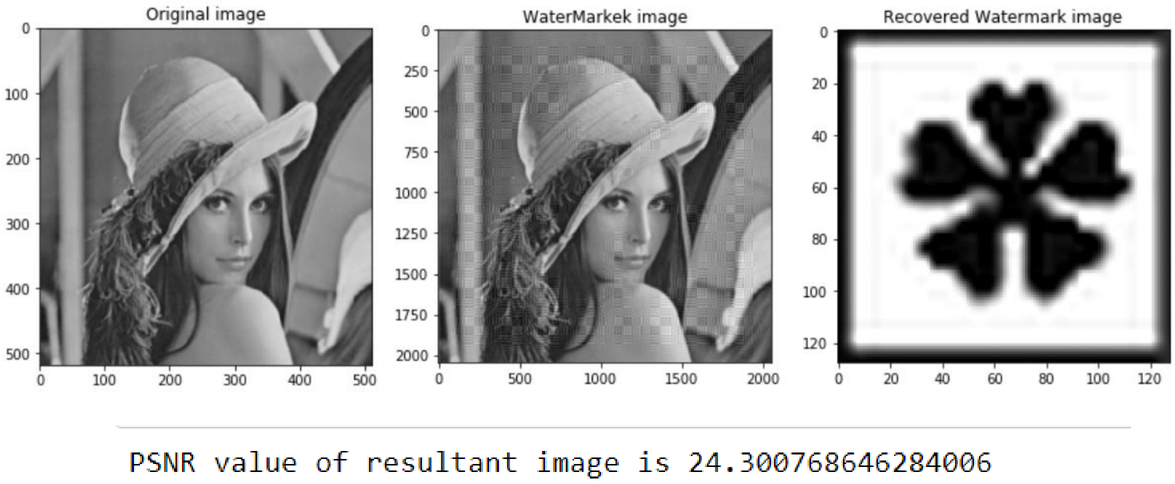
➢ Embedding

1. Input the image, I. Let the size be NxN.
2. Decompose I into blocks of appropriate size (nxn).
3. Compute the sum of intensity values for the block k.
4. Compute the block average (av).
5. Find the sum of difference (sd) between the pixel and the block's average intensity value.
6. Select blocks with (sd > th*av) for watermark embedding, else got o step 3 for computations on next block k+1.
7. Embed the blocks found eligible in step 6 as follows.
    a. Compute DCT of the block.
    b. if wm(t)==0, fw(i,j)=sf*f(i,j); where f(i,j) is a middle frequency DCT coefficient from the block and fw is the modified element.
       if wm(t)==1, leave the coefficient unaffected.
       Here wm is the watermark bit stream to be added and t =1, 2 …L, where L is the length of the watermark.
    c. Compute inverse DCT of the block
8. Repeat from steps 3 for embedding the remaining watermarks for next block k=k+1.
9. Output the watermarked image, Iw

- **Extracting**

1. Input the image I and watermarked image, Iw.
2. Decompose I and Iw into blocks of size (nxn).
3. Compute the sum of intensity values for the block, k from I.
4. Compute the block average (av).
5. Find the sum of difference (sd) between the pixels and the block's average intensity value.
6. Select all blocks with (sd > th*av), else goto step 3 for computations on next block k=k+1.
7. Compare the blocks found eligible as follows.
    a. Compute DCT of the identical blocks from I and Iw.
    b. if fw(i,j) > f(i,j), wm(t)==0 else wm(t)==1; where f and fw are mid-frequency DCT coefficients from a similar position from the blocks in I and Iw. Here, wm is the watermark bit stream.
8. Repeat the procedure from step 3 for next k=k+1 block, until all blocks are compared.
9. Output the watermark image, w.

Result



```
PSNR value of resultant image is 24.300768646284006
```

Appendix

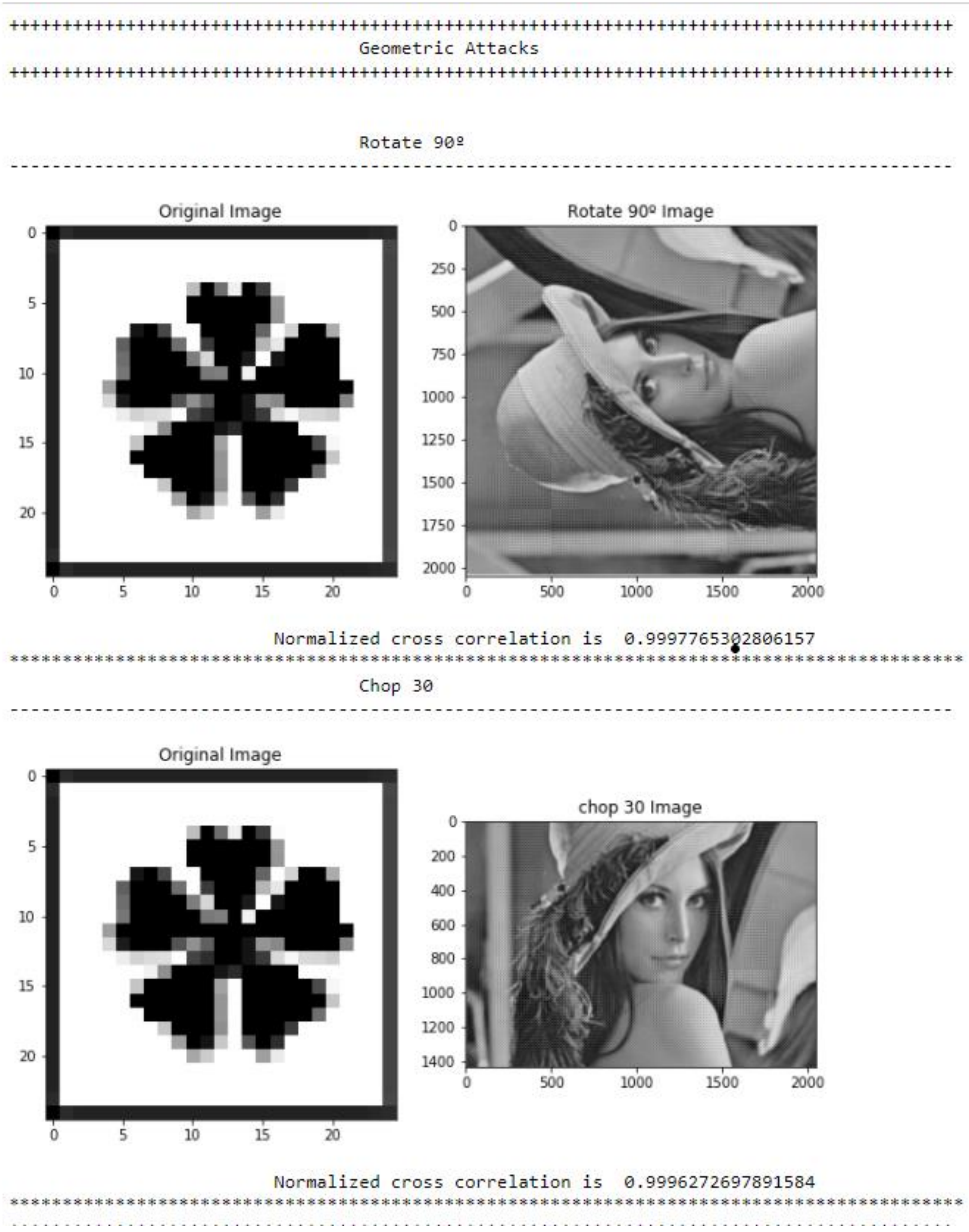
Github Link

https://github.com/voilentKiller0/Image-watermarking-using-DCT

Attacks

1. Geometric Attack :

All manipulations that affect the geometry of the image such as flipping, rotation, cropping, etc. should be detectable. A cropping attack from the right-hand side and the bottom of the image is an example of this attack.



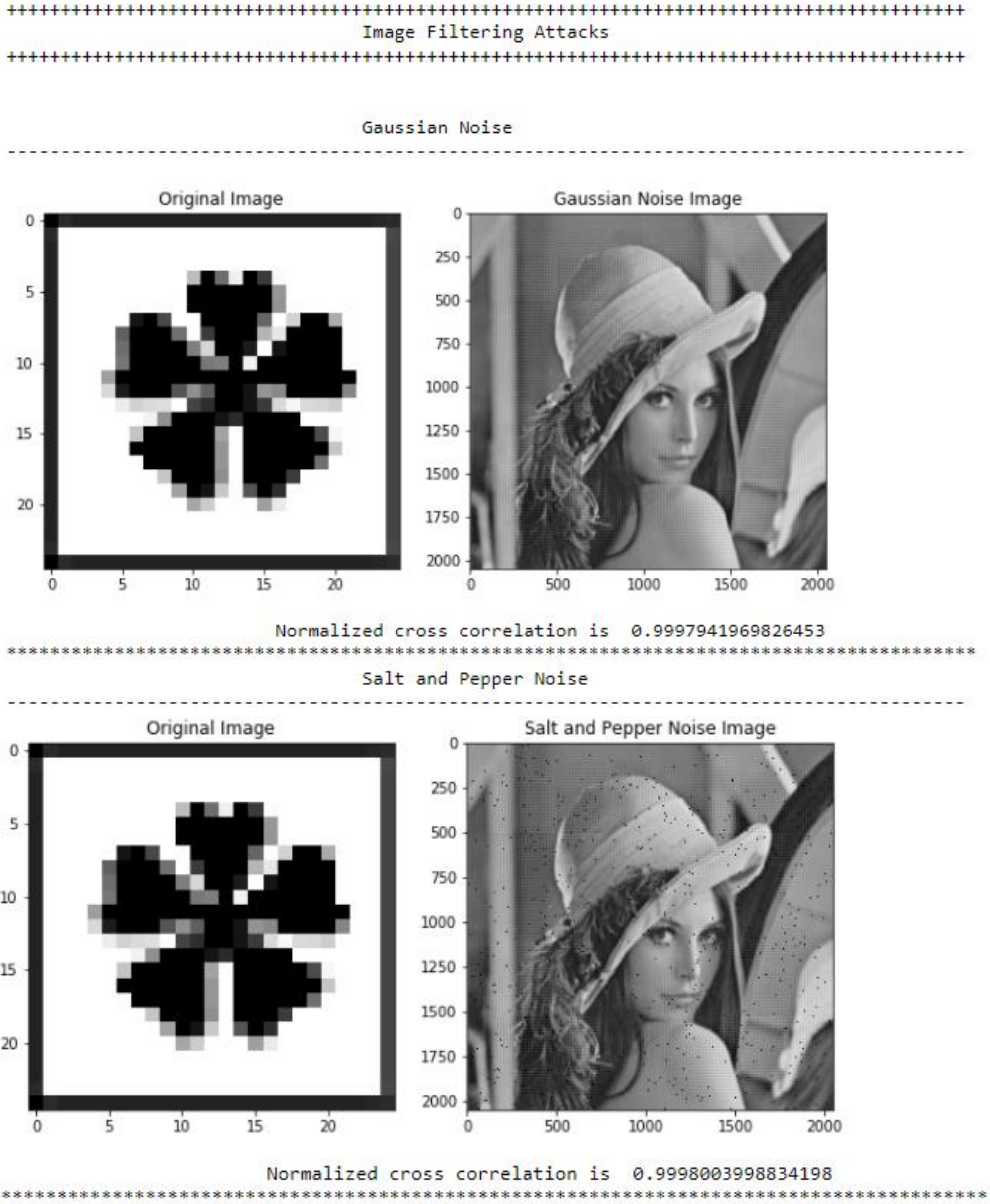
```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                Geometric Attacks
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


                                Rotate 90º
------------------------------------------------------------------------------------
```

```
                        Normalized cross correlation is  0.9997765302806157
*************************************************************************************
                                Chop 30
------------------------------------------------------------------------------------
```

```
                        Normalized cross correlation is  0.9996272697891584
*************************************************************************************
....................................................................................
```

2. **Low Pass Filtering Attack :**

   A low pass filtering is done over the watermarked image and it results in a difference map composed of noise.

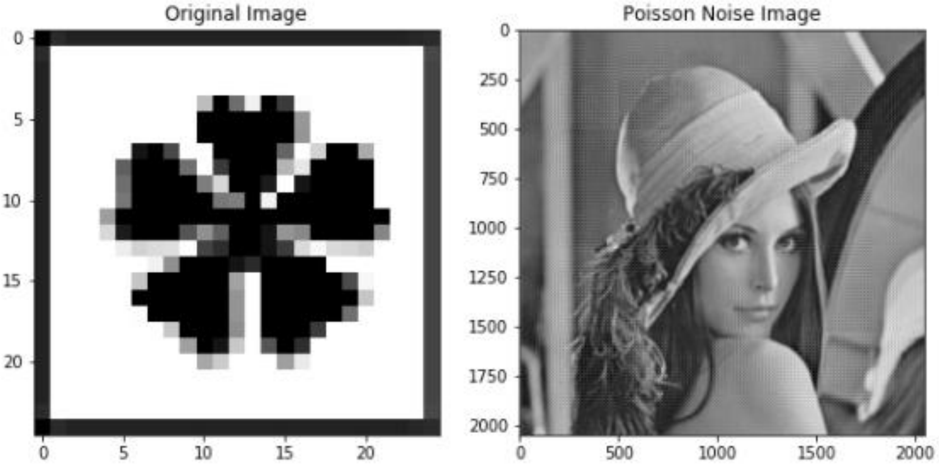


```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                              Image Filtering Attacks
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++


                              Gaussian Noise
-----------------------------------------------------------------------------------
```

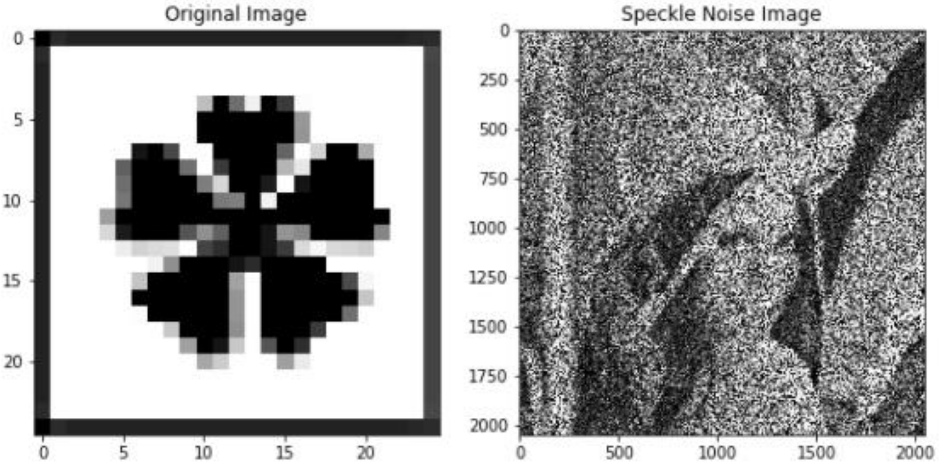
```
                        Normalized cross correlation is  0.9997941969826453
***********************************************************************************
                              Salt and Pepper Noise
-----------------------------------------------------------------------------------
```

```
                        Normalized cross correlation is  0.9998003998834198
*************************************************************************************
```

```
                                 Poisson Noise
------------------------------------------------------------------------------------------
```



```
                         Normalized cross correlation is  0.9997942733024805
*******************************************************************************************
                                 Speckle Noise
------------------------------------------------------------------------------------------
```



```
                         Normalized cross correlation is  0.999929251735013
*******************************************************************************************
```