**DIGITAL WATERMARKING**

Cryptography:

Clear information encrypted to something using a key and decrypted to get it back using the key.

Steganography:

Embedding a secret message into an innocent cover object. Trying to obscure there is a message present.

Watermarking:

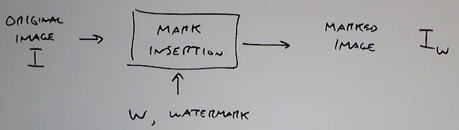
In between cryptography and steganography. Embed information into an image so that:

* Image seems unchanged
* Watermark can be extracted even after processing
* Removing the watermark should destroy the image.

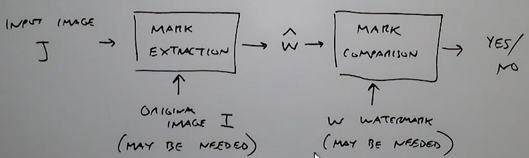
TV channels have watermark.

Movies and images (currencies and bills) have special dots which capture information upon being recorded or stored.

***Insertion/embedding***:



***Detection***:

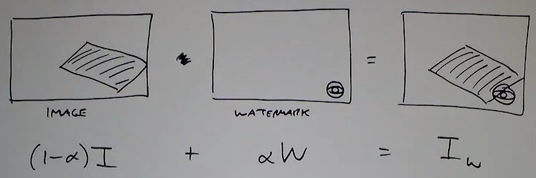


Desirable properties:

* Visually imperceptible: shouldn’t be able to tell anything has changed
* Statistically imperceptible: statistical properties mean, variance, etc) of image shouldn’t change when adding watermark to the image.
* Robust to inadvertent or intentional attacks
  + Cropping, resizing, compression, enhancement, rotation
  + Print image and rescan on high quality paper, collusion (averaging all the frames so as to remove the watermark).
* Must be fragile: watermark breaks as soon as image is modified
* High capacity: must be able to embed lots of information high lots of bits
* Speed of embedding/detection of watermark.

Ways of watermarking:

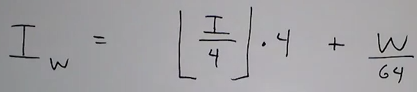
1. ***Spatial watermarking***:



The image can be tampered with in case the watermark is available on its own.

1. ***Mark the Least Significant Bits***

Consider watermark image to be 2-bits and the image itself to be 8-bits



The image now has zeroes in its Least 2 Significant Bits; now add it to the watermark.

Here we are replacing the small bits with the watermark.

* + 6-bit approximation of image
  + 2 LSBs of watermark

So the image will change by approximately three gray levels, which is not very perceptible.

* + Easy to remove
  + Not robust to noise or compression

1. ***Pseudo Random Noise Patterns***:

For encoding:

* + Split the image into blocks
  + Define pseudo-random noise patterns W0, W1 same size as the block
  + To encode bit = k: 

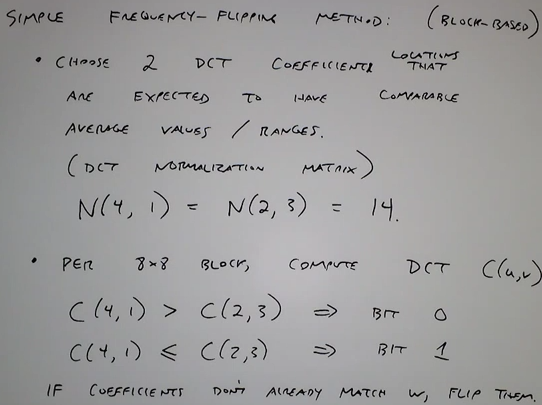
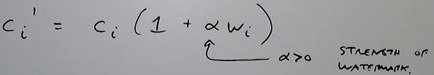
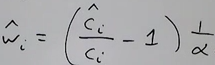
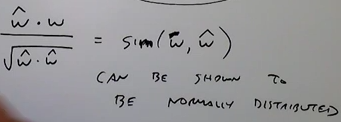
For decoding:

* + Split image into blocks
  + Correlate each block with W0, W1 and extract bit ‘k’ corresponding to higher correlation.

This is still not robust to image cropping, resizing, etc.

So most of these techniques are in the spatial domain. We will see to carry out the same in the frequency domain. It is preferable to use the DCT because most of the frequency coefficients do not carry any information. Most effective watermarking algorithms are in the frequency domain.

So IDEA: hide the information (watermark) in the visually important frequency bands. But why? So that it will not be susceptible to image compression. Storing watermark in less visual frequency bands might seem no harm but will be filtered out during compression.

1. 
2. The robust approach:
   * ***Encoding***:
     + Compute the DCT of the entire image
     + Find the ‘k’ largest magnitude coefficients (c1,… ck), excluding the dc coefficient.
     + Watermark is a k-length random vector (W1,…..Wk). (here the watermark is assumed to be a function of random noise or a random distribution.)
     + Embed the watermark: 
     + Replace ci with ci’ and take the inverse DCT. (So watermark is present only in the most significant frequencies of the image. Hence trying to remove the watermark means messing up the most visually perceptible frequencies and damaging the image)
   * ***Decoding***:
     + Compute the DCT of the image
     + Extract ‘k’ coefficients in the **known** locations 
     + Compute the inverse of the formula used in encoding process: 
     + Compute: 

So if the value is grater than 6, then the normal random variable is more than 6 deviations from the mean. So it tells how close ‘W’ is from ‘W’cap.

* + - Now apply a threshold on the obtained value.

By increasing the strength of the watermark, it would be slightly visible on the image.

