Cover Regions and Parity Bits

Any nonempty subset of {c1,……..,cI(c)} is called a cover-region. By dividing the cover into several disjoint regions, it is possible to store one bit of information in a whole cover-region rather than in a single element. A parity bit of a region I can be calculated by:

B (I) ==_LSB (cj) mod 2 j_I

We will call any nonempty subset of $\{c1, \ldots, cl(c)\}$ a cover-region. By dividing the cover in several disjoint regions, it is possible to store one bit of information in a whole cover-region rather than in a single element. A *parity bit* of a region *I* can be calculated by

$$p(I) = \sum_{j \in I} LSB(c_j) \bmod 2 \qquad (3.3)$$

In the embedding step, *l(m)* disjoint cover-regions *Ii* ($1 \leq i \leq l(m)$) are selected, each encodes one secret bit *mi* in the parity bit *p(Ii)*. If the parity bit of one cover-region *Ii* does not match with the secret bit *mi* to encode, one LSB of the values in *Ii* is flipped. This will result in *p(Ii) = mi*. In the decoding process, the parity bits of all selected regions are calculated and lined up to reconstruct the message. Again, the cover-regions can be constructed pseudo-randomly using the stego-key as a seed.

Transform Domain Techniques

We have seen that LSB modification techniques are easy ways to embed information, but they are highly vulnerable to even small cover modifications. An attacker can simply apply signal processing techniques in order to destroy the secret information entirely. In many cases even the small changes resulting out of lossy compression systems yield to total information loss.

It has been noted early in the development of steganographic systems that embedding information in the frequency domain of a signal can be much more robust than embedding rules operating in the time domain. Most robust steganographic systems known today actually operate in some sort of transform domain.

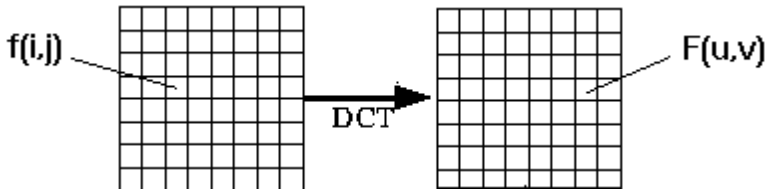
Transform domain methods hide messages in significant areas of the cover image which makes them more robust to attacks, such as compression, cropping, and some image processing, than the LSB approach. However, while they are more robust to various kinds of signal processing, they remain imperceptible to the human sensory system.

Many transform domain variations exist. One method is to use the discrete cosine transformation (DCT) as a vehicle to embed information in images; another would be the use of wavelet transforms. Transformations can be applied over the entire image, to blocks

throughout the image, or other variations. However, a trade-off exists between the amount of information added to the image and the robustness obtained. Many transform domain methods are independent to image format and may survive conversion between lossless and lossy formats.

Discrete Cosine Transform:

DCT commonly used for multimedia image/video compression. The discrete cosine transform (DCT) helps separate the image into parts with respect to the image's visual qualities. high, low & middle frequency components. The DCT transforms a signal or image from the spatial domain to the frequency domain.



A DCT is a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even proportion and computationally quite simpler than FFT. Much of the signal energy in image lies at low frequencies which appear in the upper left corner of the DCT. The lower right values represent higher frequencies which are small enough to be neglected with little visible distortion

2D DCT:

For 2D N by M image 2D DCT is defined:

$$F(u,v) = \left(\frac{2}{N}\right)^{\frac{1}{2}} \left(\frac{2}{M}\right)^{\frac{1}{2}} \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} \Lambda(u) \cdot \Lambda(v). \cos\left[\frac{\pi \cdot u}{2 \cdot N}(2i+1)\right] \cos\left[\frac{\pi \cdot v}{2 \cdot M}(2j+1)\right] \cdot f(i,j)$$

And the corresponding „inverse“ 2D DCT transform is simple F-1(u,v), i.e:

$$
\begin{aligned}
f(i,j) &= F^{-1}(u,v) \\
&= \left(\frac{2}{N}\right)^{\frac{1}{2}} \sum_{u=0}^{N-1} \sum_{v=0}^{M-1} \Lambda(u) \cdot \Lambda(v). \\
&\quad cos\left[\frac{\pi.u}{2.N}(2i+1)\right] \cdot cos\left[\frac{\pi \cdot v}{2 \cdot M}(2j+1)\right] \cdot F(u,v)
\end{aligned}
$$

where

$$
\Lambda(\xi) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for} \xi = 0 \\ 1 & \text{otherwise} \end{cases}
$$

Steganography in the DCT Domain

One popular method of encoding secret information in the frequency domain is modulating the relative size of two (or more) DCT coefficients within one image block.

Algorithm *DCT-Steg encoding process*
for *i = 1,...,l(M)* ***do***
choose one cover-block bi
Bi = D{bi}
if *mi = 0* ***then***
if *Bi(u1, v1) > Bi(u2, v2* ***then***
swap Bi(u1, v1) and Bi(u2, v2)
end if
else
if *Bi(u1, v1) < Bi(u2, v2)* ***then***
swap Bi(u1, v1) and Bi(u2, v2)
end if
end if
adjust both values so that |Bi(u1, v1) - Bi(u2, v2)| > x
b 'i = D-1{Bi}
end for
create stego-image out of all b 'i

During the encoding process, the sender splits the cover-image in 8×8 pixel blocks; each block encodes exactly one secret message bit. The embedding process starts with selecting a pseudorandom block *bi* which will be used to code the *i*th message bit. Let *Bi = D{bi}* be the DCT-transformed image block.

Before the communication starts, both sender and receiver have to agree on the location of two DCT coefficients, which will be used in the

embedding process; let us denote these two indices by ($u1$, $v1$) and ($u2$, $v2$). The two coefficients should correspond to cosine functions with middle frequencies; this ensures that the information is stored in significant parts of the signal (hence the embedded information will not be completely damaged by JPEG compression). Furthermore, we can assume that the embedding process will not degenerate the cover heavily, because it is widely believed that DCT coefficients of middle frequencies have similar magnitudes. Since the constructed system should be robust against JPEG compression, we choose the DCT coefficients in such a way that the quantization values associated with them in the JPEG compression algorithm are equal. The coefficients (4,1) and (3,2) or (1,2) and (3,0) are good candidates. One block encodes a "1," if $Bi(u1, v1) > Bi(u2, v2)$, otherwise a "0." In the encoding step, the two coefficients are swapped if their relative size does not match with the bit to be encoded. Since the JPEG compression can (in the quantization step)

Algorithm DCT-Steg decoding process
for $i = 1,...,l(M)$ **do**
get cover-block bi associated with bit i
$Bi = D\{bi\}$
if $Bi(u1, v1) \leq Bi(u2, v2)$ **then**
$mi = 0$
else
$mi = 1$
end if
end for

affect the relative sizes of the coefficients, the algorithm ensures that $|Bi(u1, v1) - Bi(u2, v2)| > x$ for some $x > 0$, by adding random values to both coefficients. The higher x is, the more robust the algorithm will be against JPEG compression, however, at the expense of image quality. The sender then performs an inverse DCT to map the coefficients back into the space domain. To decode the picture, all available blocks are DCT-transformed. By comparing the two coefficients of every block, the information can be restored.

If the constant x and the location of the used DCT coefficients are chosen properly, the embedding process will not degenerate the cover visibly. We can expect this method to be robust against JPEG compression, since in the quantization process both coefficients are divided by the same quantization values. Their relative size will therefore only be affected in the rounding step.

Performing DCT for Steganography: To embed the data in image consider a cover image of MxN& the data is to be concealed. First divide the image in 8x8 blocks. & perform 2D DCT on each block. This will generate a matrix of DCT Coefficients. The lower right coefficients represent higher frequencies which are negligible. So in next step quantization is applied on each 8x8 block to compress the block. Then LSB of DCT coefficients are replaced with data bits (that are to be hidden). Perform Inverse DCT on each resulting block. Then combine all blocks to form a stego image.