***DISREET WAVELET TRANSFORM***

It is a domain whose sole applications include the concepts of image processing and digital image compression. The foundation of the discreet wavelet transformation is primarily derived from the preliminary concepts of calculus and linear algebra. Its provides a pathway to the Fourier series and discreet convolution , and allows near immediate access to the current applications . the various concepts related to the discreet wavelet transformation are the signal denoising and compression , edge detection , image compression and use computer in both the derivation and implementation of the DWT.

DISCREET HAAR WAVELET TRANSFORM :

It’s the most elementary form of all transformations and its serves as an excellent didactic tool for the development of more sophisticated discreet wavelet transformations and their utilizations to digital signals or images.

Henceforth , we provoke the transformation as follows : Suppose we wish to transmit a list (vector) of N numbers, N even, to a friend via the Internet. To reduce transfer time, we decide to send only a length N/2 approximation of the data. One way to form the approximation is to send pairwise averages of the numbers. For example, the vector [6, 12, 15, 15, 14, 12, 120, 116] T can be approximated by the vector [9, 15, 13, 118] T . Of course, it is impossible to determine the original N numbers from this approximation, but if, in addition, we transmit the N/2 averaged directed distances, then our friend could completely recover the original data. For our example, the averaged directed distances are [3, 0, −1, −2] .

We define the one–dimensional discrete Haar wavelet transformation as the linear transformation

x → [a/d] ,

where x ∈ R N , N even, and the elements of a, d ∈ R^n/2 are

ak = x2k−1 + x2k 2 ,

dk = −x2k−1 + x2k 2 ,

where k=1 ,2 ,… ,N/2.

Provided these averages and differences , the original data can be easily recovered.

In matrix form , the change-over can be also be expressed as

Now the main advantage of transforming **x** into **a** and **d** is that the vector **a** gives us an approximation of the original vector **x** ,whereas the vector **d** allows us to catalogue the larger /smaller differences between the pairwise averages and the original data . Large differences the might be of interest if detecting edges in digital images was concerned , for example .On the other hand , the inclination might be more towards the values that are absolutely minimal , assuming the original data were homogenous , has the effect of producing large no. of zeroes in the contused vector d .In this case , the altered transforms are more susceptible to data coders in image compression applications. And

In the contused vector **d**. In such cases , the modified transforms are perceptible to data coders in image compression applications . And of course there’s no hope of recovering the original data x from the modified transform , but depending on the applications , the oblation of retardation of the resoluteness for storage size might be favourable .

***TWO DIMENSIONAL DISREET HAAR WAVELET TRANFORMATION***

(brave history see!)

// Doubt Doubt Doubt!!

If we view a grayscale digital image as an M × N matrix A whose entries are integers between 0 and 255 inclusive, where 0 represents black and 255 is white, and integers in between indicate varying degrees of intensities of gray, then application of the one-dimensional Haar wavelet transform W˜M acts on the columns of A. Each column is transformed to a column whose top half gives an approximation, or blur, of the original column and whose bottom half holds the differences or details in the data. We can process the rows of A as well by multiplying W˜M A by W˜ T N . We define the two-dimensional discrete Haar wavelet transformation of the matrix A to be W˜M AW˜ T N . Of course, both M and N must be even integers. An image (stored in A), W˜M A, and W˜M AW˜ T N are plotted in Figure

// doubtful portion complete

***Huffman coding***

Its very essential to compress a data in a format that’s easy to use to a compact format . Likewise , a decompression program returns the information . Our primary goal here is speed .The speed of transmission depends on the number of bits to send , the time required for the encoder to generate the coded message , and the subsequent time required for the decoder to recover the original ensemble. In a data storage operations , although the degree of compression is the preliminary requisite , its nonetheless necessary that the algorithm be competent for the scheme to be practical.

It is an entropy based algorithm that relies on dissection of the frequency of symbols of an array. Huffman coding can be exhibited most vividly by compressing a large raster image as follows :

Suppose we initially take a raster image with size : 5x5 with 8-bit color ,i.e. 256 different colors. So the uncompressed image will take 5x5x8 = 200 bits of storage.

The method goes like like this ---

1) The number of occurrences of each colour in the image is counted and sorting is done in decreasing order of occurrence/frequency.

2)The colours are combined together in the form of a tree such that the colours farthest from the tree are the ones with the lowest frequency.

3)The colours are then joined in pairs with a node forming a connection in between them .A node can associate to another node or to a colour.

The tree that’s obtained after all these operations is termed Huffman tree which can then be used for both encoding and decoding.

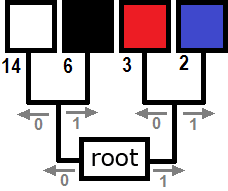
ENCODING :

* Codes are generated by moving from the root of the tree to each colour.
* If we turn right at a node , we write 1 ,and if we turn left , we write 0.
* This process yields Huffman code table in which each symbol is assigned a bit code such that the most frequently occurring symbol has the shortest code , while the least common symbol is given the longest code .

Its to be noted that the Huffman tree may not be only one of its type – there may be many other versions possible for the same image data!

Because each colour has a unique bit code that is not a prefix of any other , the colours can be replaced by their bit codes in the image record.

Below is a typical example of a Huffman tree:



Its code table then as per the rule aforementioned would be :-



So according to the algorithm , what would happen is that the colour(each one of which is represented by bit code in the image file) which appears most intermittently , which in this case is White , is will be represented with a solo bit rather than 8 bits .

In a similar fashion black will take 2 bits , Red and Blue will take 3. After these replacements are made , the 200-bit image will be compressed to 14x1 + 6x2 + 3x3 + 2x3 = 41 bits , which is five times lesser tan the original size of the image in here.

EMBEDDING PROCESS

Consider a mxn matrix 'I' representing host image or watermark image, to apply singular value decomposition (SVP) a coloured image have to be separated into three band monochrome images, where each band corresponds to a different colour, typically red, blue and green or RGB each takes values from (0-255)

A colored image is vector valued function and mathematically expressed as:

I(x , y) → [r(x, y) , b(x, y) , g(x, y)]

Then singular value decomposition applied to three matrices bands and obtained matrices U, P and V for these three matrixes separately and recombined to get: *I = UDV T*

Modifying singular value D using watermark image W of size mxn

D’ = D + αW

Where:

D: Singular values of original image

W: Watermark image

α: Positive real adjusted for watermark strength

Applying SVD on D' of equation (15) to obtain its corresponding singular values D'' in (16)

D’ = UD’’ V’T

Combining, we obtain a colored watermarked image in equation

Iw = UD’’ VT

The singular values of an image have very good stability, that is, when a same perturbation is added to an image, its Singular values do not change significantly. Each singular value specifies the brightness of an image layer while the corresponding pair of singular vectors specifies the geometry or rotation of the image[[1]](Watermarking-Colored-Digital-Image-Using-Singular-Value-Decomposition-for-Data-Protection.pdf).

B. **Method**

**1)** Algorithm stops

The flow chart of our method is given in Fig. 1. Specific steps are as follows:

**a)** Transforming: For the original noisy image, apply DCST transform yielding its coefficients in DCST domain.

**b)** Denoising: Using a threshold rule to each DCST coefficient.

**c)** Compressing: Huffman coding method is employed to encode the DCST coefficients to form a compressed format for easy storage and transmission.

**d)** Decompressing: Employing Huffman decoding method to recover the coefficients in DCST domain.

**e)** Inverse transforming: Applying inverse DCST transform to the

coefficients recovering the time-domain image.

**2)** Choice of the threshold : Thresholding processing in wavelet transform domain is a classical denoising method. Considering the relationship between S-transform and Wavelet transform, we introduce this processing into our algorithm[1].