CHAPTER 4

STEREO IMAGE COMPRESSION USING DCT II

IMAGE TRANSFORMATION

4.1 Discrete Cosine Transforms (DCT)

The discrete cosine transform (DCT) represents an image as a sum of sinusoids of varying magnitudes and frequencies. The DCT II function computes the two-dimensional discrete cosine transform (DCT) of an image. The DCT has the property that, for a typical image, most of the visually significant information about the image is concentrated in just a few coefficients of the DCT[28]. For this reason, the DCT is often used in image compression applications. For example, the DCT is at the heart of the international standard lossy image compression Algorithm known as JPEG.

The Discrete Cosine Transform (DCT) is a technique for converting a signal into elementary frequency components. It is widely used in image compression. The rapid growth of digital imaging applications, including desktop publishing, multimedia, teleconferencing, and high-definition television (HDTV) has increased the need for effective and standardized image compression techniques. Among the emerging standards are JPEG, for compression of still images; MPEG, for compression of motion video; and CCITT, H.261, for compression of video...
telephony and teleconferencing. All three of these standards employ a basic technique known as the discrete cosine transform (DCT).

4.2 Quantization of DCT coefficients

DCT-based image compression relies on two techniques to reduce the data required to represent the image. The first is quantization of the image's DCT coefficients; the second is entropy coding of the quantized coefficients. Quantization is the process of reducing the number of possible values of a quantity, thereby reducing the number of bits needed to represent it. Entropy coding is a technique for representing the quantized data as compactly as possible.

The result of a DCT is a transformation of the original source into the frequency domain. The top left entry stores the "amplitude" the "base" frequency and frequency increases both along the horizontal and vertical axes. The outcome of the DCT is usually a collection of amplitudes at the more usual lower frequencies (the top left quadrant) and less entries at the higher frequencies.

The DCT is closely related to the DFT. Both of them taking a set of points from the spatial domain and transform them into an equivalent representation in the frequency domain [39]. The DCT can concentrate the energy of the transformed signal in low frequency, whereas the DFT cannot. According to Parseval’s theorem, the energy is the same in the
spatial domain and in the frequency domain. Because the human eyes are less sensitive to the low frequency component, we can focus on the low frequency component and reduce the contribution of the high frequency component after taking DCT.2. For image compression, the DCT can reduce the blocking effect than the DFT. After transformation, the element in the upper most left corresponding to zero frequency in both directions is the “DC coefficient” and the rest are called “AC coefficients.”

It is usual to just zero out these higher frequencies as they typically constitute very minor parts of the source. However, this does result in loss of information.

Fig 4.1 Frequency Domain of Image
To complete the compression it is usual to use a lossless compression over the DCT source. This is where the compression comes in as all those runs of zeros get packed down to almost nothing.

One possible advantage of using the DCT to find similar regions is that you can do a first pass match on low frequency values (top-left corner). This reduces the number of values you need to match against. If you find matches of low frequency values, you can increase into comparing the higher frequencies. In the JPEG image compression Algorithm, the input image is divided into 8-by-8 or 16-by-16 blocks, and the two-dimensional DCT is computed for each block. The DCT coefficients are then quantized, coded, and transmitted [64]. The JPEG receiver decodes the quantized DCT coefficients, computes the inverse two-dimensional DCT of each block, and then puts the blocks back together into a single image.

For typical images, many of the DCT coefficients have values close to zero; these coefficients can be discarded without seriously affecting the quality of the reconstructed image. Although there is some loss of quality in the reconstructed image, it is clearly recognizable, even though almost 85% of the DCT coefficients were discarded.
4.3 Stereo Image Compression using DCT II

In Stereo Image pair compression using DCT II transform Algorithm, the objective is to reduce irrelevance and redundancy of the image data and to store and transmit in an efficient form. Stereoscopic imaging systems are increasingly used for multimedia, telepresence or video conference. This approach permits as to enhance 3D perception of the image by preserving a slightly different viewpoint of the scene of each eye. Since the usage of stereoscopy system imply a significant increase in the amount of visual information for remote transmission. So we need compact representation for the information.

Stereo image pair compression is minimizing size of an image file without degrading the quality of the image to a unacceptable level. The deduction in file size allows more images to be stored in a given amount of disk memory space and also reduce the time needed for images to be sent over the internet and downloaded from internet.

The transformation coding is a component of image processing and is very effective in the compression of images[68]. Transformation is mainly used to remove the duplicate information between neighboring pixels the images. It depends on the assumption that, a pixel in an image is correlated to its neighbor and this correlation is used to predict the value of the neighboring pixel. The 2D discrete cosine Transform (DCT II)
was used to perform the transformation because of its strong energy compaction property.

4.4 DCT II Compression Algorithm

The steps involved in DCT II stereo image compression Algorithm are given below.

Step 1: 2D Forward Discrete Cosine Transform

The left image in the stereo pair is called as reference image. Reference image is transformed into 2D forward discrete cosine Transform. (FDCT II). The resultant image matrix is quantized using the quantization matrix.

Ensure that, the pixels are centered on zero. Since the value of pixel is between the range of 0 to 255, than each pixel value is subtracted from 128 to make the pixel value centered around zero. The image is divided into 8 x 8 blocks. The 2D DCT for each block is calculated by the formulae 4.1,

\[ C(u,v) = \alpha(u) \alpha(v) \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cos \left( \frac{\pi(2x+1)u}{2N} \right) \cos \left( \frac{\pi(2y+1)v}{2N} \right) \]

............4.1

For u and v, the values are 0,1,2,...7

alpha (u) and alpha (v) -- horizontal and vertical spatial frequencies.

They are defined in 4.2
\[
\alpha(u) = \begin{cases} 
\sqrt{\frac{1}{N}} & \text{for } u = 0, \\
\sqrt{\frac{2}{N}} & \text{for } u \neq 0.
\end{cases}
\]

**Step 2: Quantization**

The goal of quantization is to reduce most of the less important high frequency DCT coefficients to zero, the more zeros we generate the better the image will compress. The human eyes are insensible to variations in brightness of high frequency components over a large area. So that the high frequency pixel values in the image matrix are rounded off to zero without the user identifying any difference in the appearance of the image. This process perform the compression by dividing the DCT output of the pixel values of each block by a quantization coefficient then rounding down the result into the near integer. The 8 x 8 Quantization matrix is given below:

\[
\begin{bmatrix}
16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\
12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\
14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\
14 & 17 & 22 & 29 & 51 & 87 & 88 & 62 \\
18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\
24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\
49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\
72 & 92 & 95 & 98 & 112 & 100 & 103 & 99
\end{bmatrix}
\]

**Fig 4.2** Quantization matrix
After the DCT transformation is performed the image pixels are divided into 8x8 blocks and each and every pixel is normalized by Equation 4.3.

\[ C_{ij}' = \text{round} \left( \frac{C_{ij}}{Q_{ij}} \right) \] 

......4.3

**Pixel normalization by Quantization**

The quantization is performed by dividing the pixel value of Image quantization matrix so the large pixels values are become smaller and they need small size of storage it is a lossy step of compression the selection of quantization matrix or quantized value may affect the entropy and the ratio of compression.

By taking small value for quantization than the quality is better and yields less MSEC means square error. But the compression ratio is less. The quality of compressed image is also affect by the size of block. The higher block size yields higher compression ratio and the quality is poor.

**Step 3: Huffman coding**

The Huffman coding is an entropy encoding Algorithm used for lossless data compression. Here the variable length code table is derived based on the probability of occurrence for each possible value of the source symbol.

Huffman coding performs lossless compression by

- Assisting the pixel probabilities in a decreasing orders and considers them as leaf nodes of a tree.
• Combine the two nodes with smallest probability to form a new node the probability of new node is the sum of the two merged nodes.
• Assign 1 and 0 to each pair of merged made.
• Access sequentially from the root node to the leaf node where the pixel is located.

The Algorithms produce the Huffman code book for any given set of probabilities. Coding and decoding in performed simply by accessing the values from a table in code book.

**Step 4: Block matching Algorithm**

Take the right image to do compression since the left and right images are very similar to each other the difference between the two imaged are estimated by Block matching Algorithm The result of bloc matching Algorithm is disparity vector which are compressed into a bit stream Huffman coding The basic concept of block matching Algorithm is as follows:

The blocks will have M rows and N columns and the block size to be searched is assumed to be ± r pixels. The parameter search size is the range over which the reference image will search for the most excellent matching block in the second image. This means that if the search size is 8 pixels, the best matching block will be searched over ±8 pixels relative to the position of the current block. The x-component and the y-
component of the image vectors were stored in mvx and mvy respectively. Therefore, mvx (i, j) stores the x-component of the disparity vector corresponding to (i,j) -th and mvy (i,j) stores the y-component of the disparity vector corresponding to the particular block. The block matching Algorithm is very easiest to implement.

The Algorithm to estimate the disparity is given below

**Step 1:** Determine the size of the reference frame.

**Step 2:** Calculate the size of disparity vectors mvx and mvy, based on the Size of reference Image, M and N

For each block, do the following steps:

- Extract the block from reference Image
- Search the second image, over a region of ±R pixels relative to the position of the current block.
- Calculate the sum of absolute difference between the pixels of the reference image and the second image.
- The minimum SAD is the best matching block.
- The difference between the reference image and the second image is the disparity vector for the two images.
- This value is stored in the appropriate locations in mvx and mvy.

**Step 3.** Decoding the Image. To decode the image, all the steps in encoding were performed in reverse order.
Fig 4.3 Flow of Reconstructing the Image by DCT II

In Inverse Quantization the image matrix is divided into 8x8 blocks and each block is multiplied by the Quantization matrix in Fig 4.4. This step is referred to as inverse quantization even though quantization cannot be inverted as it is lossy, the image is divided into NxN blocks and the inverse 2-D DCT of each pixel is calculated as:

The image is divided into NxN blocks and the inverse 2-D DCT (Discrete Cosine Transform) of each pixel is calculated as

\[ f(x, y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} a(u, v) c(u, v) \cos \left( \frac{\pi(2u+1)x}{2N} \right) \cos \left( \frac{\pi(2v+1)y}{2N} \right) \]

\[ \ldots \ldots 4.4 \]

The number 128 was subtracted from each pixel while calculating the forward DCT-II, therefore 128 is now added to each pixel value.

To reconstruct the second image, we need image matrix of the reconstructed left image and the disparity vectors calculated. The Algorithm for reconstructing the right image is:
**Step 1:** Calculate M and N, based on the size of the reference image and mvx.

**Step 2:** Initialize disparity compensation vector and second image as the same size as the reference image.

**Step 3:** For each block in the reference image, do

i. Extract the current block from reference image

ii. Extract the disparity vectors corresponding to the current location.

iii. Add the disparity vectors to the current block

The above Algorithm is implemented in JAVA Developer Kit 7.0 and the results are represented by the Table 4.1 and the figure 4.7.

**The experimental results of various images compressed by DCT II**

**Stereo images are given below,**

**Table 4.1 Performance of DCT II Stereo image compression**

<table>
<thead>
<tr>
<th>Image Name</th>
<th>Bit Rate</th>
<th>PSNR</th>
<th>Size of original Image</th>
<th>Size of Compressed Image</th>
<th>Compression Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corridor</td>
<td>1.82272</td>
<td>18.0</td>
<td>27KB</td>
<td>8.90kb</td>
<td>3.033</td>
</tr>
<tr>
<td>Hotel lane</td>
<td>1.253376</td>
<td>16.5</td>
<td>26 kb</td>
<td>6.12kb</td>
<td>4.248</td>
</tr>
<tr>
<td>Forest</td>
<td>1.376256</td>
<td>27.0</td>
<td>33 kb</td>
<td>6.72kb</td>
<td>4.910</td>
</tr>
<tr>
<td>Pillar</td>
<td>1.757184</td>
<td>16.5</td>
<td>38 kb</td>
<td>8.58kb</td>
<td>4.428</td>
</tr>
</tbody>
</table>
Fig 4.4 Performance of DCT II Stereo Image Compression (graph)

The table 4.1 and the Fig 4.4 illustrate the characteristics of the stereo pair image compression Algorithm using DCT II image transformation. It shows the PSNR and Bitrate value of the images and the size of original image as well as compressed image with their compression ratio.