Application of DCT in image processing

Nitesh Agarwal
Department of Computer Science
Jodhpur Institute of Engineering & Technology
Jodhpur, India
niteshagarwal.234@rediffmail.com

Dr. A.M. Khan
Department Of Applied Sciences
Jodhpur Institute of Engineering & Technology
Jodhpur, India
arif.khan@jietjodhpur.com

Abstract—Discrete Cosine Transform (DCT) is an important technique or method to convert a signal into elementary frequency component. It is widely used in image compression techniques like in JPEG compression. It converts each pixel value of an image into its corresponding frequency value. The present paper deals with the study of transformation of an 8 bit (b/w) image into its frequency domain through DCT technique.

1. Introduction

DCT convert an image into its equivalent frequency domain by partitioning image pixel matrix into blocks of size N*N, N depends upon the type of image. For example if we used a black & white image of 8 bit then all shading of black & white color can be expressed into 8 bit hence we use N=8, similarly for color image of 24 bit we can use N=24 but using block size N=24 time complexity may increase hence we operate DCT on individual color component for a color image. Color image consist of 8 bit red + 8 bit green + 8 bit blue hence we apply DCT on each color component (Red, Green, Blue) using block size N=8.

1.1 One-Dimensional DCT

If we have one-D sequence of signal value of length N then its equivalent DCT can be expressed as

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

for $u = 0, 1, 2, ..., N - 1$.

& inverse transformation is defined as

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

Where $f(x)$ is signal value at point x & $\alpha(u)$ is transform coefficient for value $u$.

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{N}} & \text{for } u = 0 \\ \frac{2}{\sqrt{N}} & \text{for } u \neq 0 \end{cases}$$

It is clear from (1) for $u=0$,

$$C(u = 0) = \frac{1}{N} \sum_{x=0}^{N-1} f(x)$$

i.e. 1st transformation coefficient is the average value of sample sequence, this coefficient known as DC coefficient & all other coefficient known as AC coefficient.

1.2 Two – Dimensional DCT

An image is 2-D pixel matrix where each position (i,j) represents a color value for that particular point or position. Hence to transform an image into its equivalent DCT matrix we use 2-D DCT.

2-D DCT can be defined as

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

for $u, v = 0, 1, 2, ..., N - 1$.

& inverse transformation is defined as

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

Where $C(u,v)$ represents frequency value for $u,v$ & $f(x,y)$ represents pixel color value at position $(x, y)$.

$$\alpha(u) = \begin{cases} \frac{1}{\sqrt{N}} & \text{for } u = 0 \\ \frac{2}{\sqrt{N}} & \text{for } u \neq 0 \end{cases}$$

$$\alpha(v) = \begin{cases} \frac{1}{\sqrt{N}} & \text{for } v = 0 \\ \frac{2}{\sqrt{N}} & \text{for } v \neq 0 \end{cases}$$
2. Main Results

2.1 Implementation of DCT

This paper describes how a b/w image is converted into an equivalent frequency domain using DCT.

Steps involved in this implementation
1. Create pixel matrix of the image & divide it into blocks of size 8*8
2. Apply FDCT (Forward Discrete Cosine Transform) on each 8*8 block of pixel matrix to get equivalent 8*8 DCT blocks.
3. To get original image we apply IDCT (Inverse Discrete Cosine Transform) on each 8*8 block DCT & get its equivalent 8*8 IDCT block.
4. Using 8*8 IDCT blocks we create original pixel matrix to get original image.

2.1.1 Algorithm 1

Get_8*8_blocks (image)
{
    n=8, k=0;
    width=width of image;
    height=height of image;
    for ( i=0; i < width/n; i++)
    {
        for ( j=0; j < height/n; j++)
        {
            xpos = i * n;
            ypos = j * n;
            for ( a=0; a < n; a++)
            {
                for ( b=0; b < n; b++)
                {
                    color = color at position(xpos+a, ypos+b);
                    block[k][a][b]=color-128;
                }
            }
            k=k+1;
        }
    }
} // end of Get_8*8_blocks

2.1.2 Algorithm 2

FDCT (block[][])[]
{
    width=width of image, N=8;
    height=height of image;
    q=(width/8)*(height/8)
    for ( i=0; i < q; i++)
    {
        for ( u=0; u < N; u++)
        {
            for ( v=0; v < N; v++)
            {
                if (u==0) {
                    alpha(u)=\sqrt{\frac{1}{N}}
                } else {
                    alpha(u)=\sqrt{\frac{2}{N}}
                }
            }
        }
    }
    sum=0;
    for(x=0;x<N;x++)
    {
        for(y=0;y<N;y++)
        {
            sum+=block[i][x][y] * cos[\pi \frac{(2x+1)u}{2N}] * cos[\pi \frac{(2y+1)v}{2N}];
        }
    }
    dct[i][j][k]=alpha(u)*alpha(v)*sum;
} // end of FDCT
2.1.3 Algorithm 3

IDCT(dct[][][])
{
    width=width of image, N=8;
    height=height of image;
    q=(width/8)*(height/8)
    for ( i=0; i < q; i++)
    {
        for ( x=0; x< N; x++)
        {
            for ( y=0; y< N; y++)
            {
                sum=0;
                for ( u=0; u<N; u++)
                {
                    for ( v=0; v<N; v++)
                    {
                        if (u==0)  {
                            \( \alpha(u) = \frac{1}{\sqrt{N}} \)
                        }
                        else{
                            \( \alpha(u) = \frac{2}{\sqrt{N}} \)
                        }
                        if (v==0){
                            \( \alpha(v) = \frac{1}{\sqrt{N}} \)
                        }
                        else {
                            \( \alpha(v) = \frac{2}{\sqrt{N}} \)
                        }
                        \(
                        \text{sum} = \text{sum} + \alpha(u) \times \alpha(v) \times \text{dct}[i][u][v] \times \cos \left( \frac{\pi (2x+1)u}{2N} \right) \times \cos \left( \frac{\pi (2y+1)v}{2N} \right);
                        \)
                    }
                }
                \text{idct}[i][j][k]=sum;
            }
        }
    }
} // end of IDCT

2.1.4 Algorithm 4

Get_Image(pixmat[][][])
{
    k=0;
    width=width of image;
    height=height of image;
    for ( i=0; i < width; i++)
    {
        for ( j=0; j < height; j++)
        {
            x_pos = i * n;
            y_pos = j * n;
            for (a=0; a < n; a++)
            {
                for (b=0; b < n; b++)
                {
                    color=(int)pixmat[k][a][b];
                    set color at position (x_pos+a, y_pos+b);
                }
            }
            k++;
        }
    }
} // end of Get_Image

2.2 Outputs

1. Convert pixel matrix into blocks of size 8*8

Input Image of size 16*16

Output blocks of size 8*8

2. Transform Input image into equivalent DCT image

Input Image of size 16*16

Output DCT Image of size 16*16
3. Get original image from DCT image

3. Modification in original DCT

3.1 Using sin operator rather than cos
There is a difference of π/2 between sin & cos operator hence using sin rather than cos operator in DCT may loss some pixel data.

3.2 Change in block size
All shading of black & white image can be expressed in 8 bit of blocks hence we use block size 8*8 to perform DCT on it. But in color image each color value of a pixel can be expressed into 24 bit of block which contain 8 bit red + 8 bit green + 8 bit blue. To transform a color image into its equivalent DCT format we extract each 8 bit color component from 24 bit of block & then perform 8*8 DCT on each color component rather than using 24*24 DCT for 24 bit block. The main reason is that if use 24*24 DCT rather than 8*8 DCT the time complexity of DCT is increases in a very large amount.

For example

For an image of size 48*48

1. If 8*8 DCT used
   Total no of blocks q=(48/8)*(48/8)=36
   For FDCT
   for ( i=0; i < q; i++) // loop runs 36 times
     { for ( u=0; u< 8; u++) // loop runs 36*8 times
       { for ( v=0; v < 8; v++) // loop runs 36*8*8 times
         { }
       } } // end of for loop y
   } // end of for loop x
 } // end of for loop v
} // end of for loop u
} // end of for loop i

Total no. of iteration = 36*8*8*8*8= 147456

2. If 24*24 DCT used
   Total no of blocks q=(48/24)*(48/24)=4
   For FDCT
   for ( i=0; i < q; i++) // loop runs 4 times
     { for ( u=0; u < 24; u++) // loop runs 4*24 times
       { for ( v=0; v < 24; v++) // loop runs 4*24*24 times
         { }
       } } // end of for loop v
   } // end of for loop u
 } // end of for loop x
} // end of for loop y
} // end of for loop i

IDCT
Input DCT Image of size 16*16
Output Image of size 16*16

FDCT
Input Image of size 16*16
Output DCT Image of size 16*16

IDCT
Input Image of size 16*16
Output Image of size 16*16

www.ijert.org
Total no. of iteration = 4 * 24 * 24 * 24 * 24 = 1327104

Hence 24 * 24 DCT required 1327104 - 147456 = 1179648 extra iteration to perform DCT which increases time complexity in large amount hence DCT used with block size 8*8.

4. Conclusion

The result presented in this document shows that
1. It is very easy to implement DCT rather than other transformation on image.
2. If DCT used with sin operator rather than cos some pixel data may lose. But if we use DCT with sin operator as

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

and its inverse as

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

then there is no loss of pixel data because it is equivalent to DCT with cos operator.
3. If DCT used with block size 24*24 rather than block size 8*8 then time complexity of DCT is increases in very large amount.

References