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

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

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

& inverse transformation is defined as

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

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

$$\alpha(u)=\begin{cases}\sqrt{\dfrac{1}{N}} & for\ \ u=0\\ \sqrt{\dfrac{2}{N}} & for\ u\neq 0\end{cases} \qquad (3)$$

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

$$C(u=0)=\frac{1}{N}\sum_{x=0}^{N-1} f(x) \qquad (4)$$

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

$$C(u,v)=\alpha(u)\,\alpha(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} f(x,y)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \qquad (5)$$

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

& inverse transformation is defined as

$$f(x,y)=\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}\alpha(u)\,\alpha(v)c(u,v)\cos\left[\frac{\pi(2x+1)u}{2N}\right]\cos\left[\frac{\pi(2y+1)v}{2N}\right] \qquad (6)$$

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

$$\alpha(u)=\begin{cases}\sqrt{\dfrac{1}{N}} & for\ \ u=0\\ \sqrt{\dfrac{2}{N}} & for\ u\neq 0\end{cases} \qquad (7)$$

$$\alpha(v)=\begin{cases}\sqrt{\dfrac{1}{N}} & for\ \ v=0\\ \sqrt{\dfrac{2}{N}} & for\ v\neq 0\end{cases} \qquad (8)$$

2. Main Results

2.1 Implementation of DCT

This paper describe how a b/w image is convert into equivalent frequency domain using DCT.

Steps involved in this implementation

1. Create pixel matrix of the image & divided 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;
           }  //end of for loop b
             k=k+1;
        }     // end of for loop a
     }         // end of for loop j
}              // end of for loop i
}// 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{

$$\alpha(u) = \sqrt{\frac{2}{N}}$$

}

if (v==0){

$$\alpha(v) = \sqrt{\frac{1}{N}}$$

}

else {

$$\alpha(v) = \sqrt{\frac{2}{N}}$$

}

sum=0;

for(x=0;x<N;x++)

{ for(y=0;y<N;y++)

{ sum= sum + block[i][x][y] * $\cos\left[\frac{\pi(2x+1)u}{2N}\right] * \cos\left[\frac{\pi(2y+1)v}{2N}\right]$;

} // end of for loop y

} // end of for loop x

dct[i][j][k]=($\alpha(u) * \alpha(v)$ *sum);

} // end of for loop v

} // end of for loop u

} // end of for loop i

}//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;b<N;v++)
{ if (u==0) {

$$\alpha(u) = \sqrt{\frac{1}{N}}$$

}
else{

$$\alpha(u) = \sqrt{\frac{2}{N}}$$

}
if (v==0){

$$\alpha(v) = \sqrt{\frac{1}{N}}$$

}
else {

$$\alpha(v) = \sqrt{\frac{2}{N}}$$

}

sum= sum $+\alpha(u)*\alpha(v)$ *dct[i][u][v] * $\cos\left[\frac{\pi(2x+1)u}{2N}\right] * \cos\left[\frac{\pi(2y+1)v}{2N}\right]$;

} // end of for loop v
} // end of for loop u
idct[i][j][k]=sum;
} // end of for loop y
} // end of for loop x
} // end of for loop i

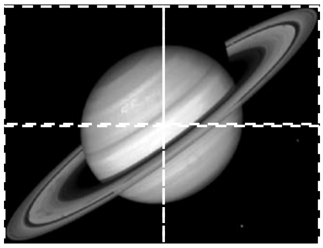
} // end of IDCT

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++) {
xpos = i * n;
ypos = j * n;
for (a=0; a < n; a++)
{ for (b=0; b < n; b++)
{
color=(int)pixmat[k][a][b];
set color at position (xpos+a, ypos+b);
}// end of loop b
} // end of loop a
k++;
} // end of loop j
} // end of loop i
}// end of Get_Image

2.2 Outputs

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



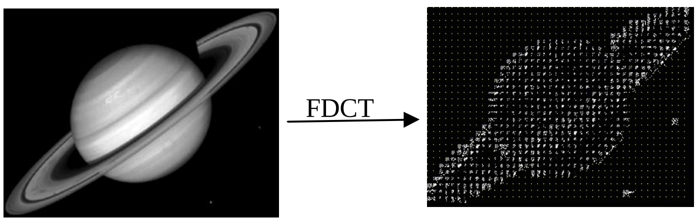
=

8*8 block 1	8*8 block 2
8*8 block 3	8*8 block 4

Input Image of size 16*16

Output blocks of size 8*8

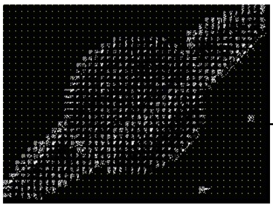
2. Transform Input image into equivalent DCT image



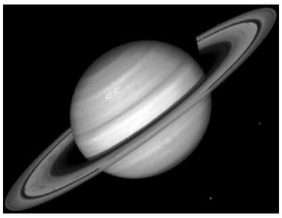
Input Image of size 16*16

Output DCT Image of size 16*16

3. Get original image from DCT image



IDCT



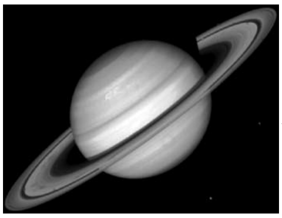
Input DCT Image of size 16*16

Output Image of size 16*16

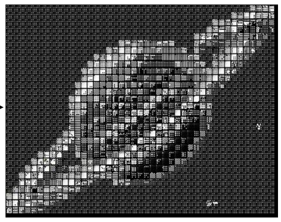
3. Modification in original DCT

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

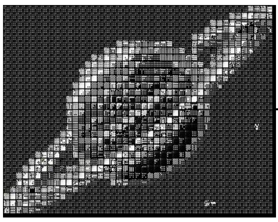


FDCT Using sin operator

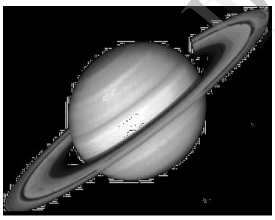


Input Image of size 16*16

Output DCT Image of size 16*16



IDCT Using sin operator



Input Image of size 16*16

Output Image of size 16*16

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
     {

     }
    for( x=0;x<8;x++)// loop runs 36*8*8*8 times
     {  for( y=0;y<8;y++) // loop runs 36*8*8*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
```

$\boxed{\text{Total no. of iteration = 36*8*8*8*8= 147456}}$

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
     {

     }
 for( x=0;x<24;x++)// loop runs 4*24*24*24 times
  {
for( y=0;y<24;y++) // loop runs 4*24*24*24*24 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 =4*24*24*24*24= 1327104

Hence 24 * 24 DCT required 1327104-147456=1179648 extra iteration to preform 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

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

& it's inverse as

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

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

[1] N.Ahmed, T.Natatarajan, and K.R. Rao, "Discrete Cosine Transform", IEEE Transactions on Computers, vol. C-32, pp. 90-93, Jan. 1974.
[2] Maneesha Gupta and Dr.Amit Kumar Garg, "Analysis Of Image Compression Algorithm Using DCT" International Journal of Engineering Research and Applications (IJERA), vol.2, pp. 515-521, Jan-Feb 2012
[3] Andrew B. Watson, "Image Compression Using Discrete Cosine Transform", NASA Ames Research Centre, 4(1), pp. 81-88,1994.
[4] Anjali Kapoor and Dr. Renu Dhir, "Image Compression Using Fast 2-D DCT Technique", International Journal on Computer Science and Engineering (IJCSE), vol. 3 pp. 2415-2419, 6 June 2011.
[5] Harley R. Myler and Arthur R. Weeks "The Pocket Handbook of Image Processing Algorithms in C", ISBN 0-13-642240-3 Prentice Hall P T R Englewood Cliffs , New Jercy 07632.
[6] Iain E.G. Richardson "H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia", ISBN 0470848375, 9780470848371, Wiley,2003.