

## Lecture - 26

### Discrete Cosine Transform

DFT  
↙  
**Discrete Cosine Transform**  
DCT

Disadvantages of DFT

- it is complex
- it has poor energy compaction

Energy compaction: is the ability to pack the energy of the spatial sequence into as few frequency coefficients as possible


Application: **Image compression**  
 if compaction is high, then transmit few coefficients instead of the whole set of pixels

**Discrete Cosine Transform**

- it is real
- it has better energy compaction

Handwritten notes and diagrams:

- $X(k) \rightarrow a+jb$
- $X(k, l) \rightarrow$
- DFT  $\rightarrow$  [ ]
- 8 bit
- 2, 2
- 8 bit
- DF  $\rightarrow$  [ ]
- [ ] (crossed out)



So, what are the advantages and disadvantages of discrete Fourier transform? I know DFT can be efficiently calculated, which is the advantage. DFT is the frequency domain representation of the signal. But what is the disadvantage? Now, you know that  $X(K)$  or  $X(K, l)$  is a complex signal. If I said it is a complex value, it has in the form of  $a + jb$ , it is in the form of both in x and y direction  $a + jb$ .

So, both are complex. So, if when I say in transform domain signal is complex, a signal is not real. Then it has a poor energy compaction poor energy compaction. What is energy compaction? It is the ability to pack the energy of the spatial sequence into as few frequency coefficients as possible and as few frequency components as possible.

So, the whole energy is packed in a few frequency components, which is called compaction energy compaction. DFT has poor energy compaction; why? Also, I will discuss. Then energy compaction high energy compaction, what is an application of high energy compaction? Why did I require high-energy compaction?

Take an example of image compression. So, what do we require? I require the low frequency or low; let us say I have an image if I have it in the frequency domain. If I find out the whole image, I have taken the DFT. So, I get the image spectra.

So, if I say the whole energy is here, and this portion of the energy is very, very little. Then, what I can say is that instead of transmitting the entire signal, I only transmit the high-energy portion, and I can reconstruct that image without losing so much information. So, that is image compression.

So, image compression is a technique. First, you have to think about which is where the energy is compaction. The which are frequency energy is very high. You can use more bits to quantize that frequency, or you can say the encoded frequency and less number of bits, which are low energy.

So, you know that image compression has a first DCT image energy compaction. And then, based on the energy compaction, you quantized it. And then, you align, assign the code, and transmit it.

An example is that suppose I have encoded a signal in 8-bit. Now, if I know I have a sample whose value cannot be more than 2. So, why should I assign 8 bits for every

sample? I can assign it a dynamic depending on the value of the energy and amount of the energy I can assign the quantization bit.

So, basically, all samples are not the same size though. So, all pixels are not represented by the same number of binary bits. So, I get an image compression. I am not going into details on image compression. So, if you are interested in image processing, you can study image compression.

So, I require an algorithm that has a high energy compaction or better energy compaction. That algorithm is a discrete cosine transform. What is the difference with the DFT? It is real. It is not complex. Cosine transforms, if I multiply with a cos function, it is a real function. And has the better energy compaction, why? I will discuss later on.

So, I have a transform, which is called a discrete cosine transform. It is real, and it has a better energy compaction. That is why this DCT is very much used in image processing. DCT is also used in MFCC coefficient extraction in the case of speech processing. Because it is real and it has a better; because what I want, I want the high energy component to be clubbed together.

(Refer Slide Time: 06:01)

There are various versions of the DCT: DCT-I to DCT-IV  
The most popular is the DCT-II

$$C_x[k] = \begin{cases} \sum_{n=0}^{N-1} 2x[n] \cos\left(\frac{\pi}{2N} k(2n+1)\right) & 0 \leq k < N \\ 0 & \text{otherwise} \end{cases}$$

$$x[n] = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} w[k] C_x[k] \cos\left(\frac{\pi}{2N} k(2n+1)\right) & 0 \leq n < N \\ 0 & \text{otherwise} \end{cases}$$

$$w[k] = \begin{cases} \frac{1}{2} & k=0 \\ 1 & 1 \leq k < N \end{cases}$$

Handwritten notes and diagrams on the slide include:

- Red circles around  $C_x[k]$ ,  $x[n]$ , and  $w[k]$  in the equations.
- Red arrows pointing from the equations to handwritten notes: "1D DCT" (twice), "2D DCT", and "3x3 DCT".
- A small video inset of a man speaking is visible in the bottom right corner.

So, that is why we use discrete cosine transform. So, DCT may be 1D, again, DCT may be 1D, or DCT may be 2D. So, this is an example of 1D DCT. What is 1D DCT?

$$c_x[k] = \sum_{n=0}^{N-1} x[n] \cos\left(\frac{\pi}{2N}k(2n+1)\right) \quad \text{for } 0 \leq k < N$$

this is the expression of DCT this is the DCT coefficient.

What is the inverse DCT IDCT?  $x[n]$  is equal to 1 by N this one, where omega k is equal to half or 1. Why this one? I can derive DC DFT from DCT using DFT I can derive DCT. DCT has a different version, DCT I, DCT II, and DCT IV, but the common use is DCTII. It only differs by this factor. There will be some factors.

(Refer Slide Time: 07:03)

2D DCT

- Corresponding 2D formulation

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

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

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

Handwritten notes:  $\alpha(0) = \frac{1}{\sqrt{N}}$ ,  $\alpha[N-1] = \frac{2}{\sqrt{N}}$ ,  $C(0,0) = \frac{1}{N}$ .

If I say I show you, this is the one another version 2D formula of DCT. Here, you can see that the  $\alpha(u)$  is the value of 1 by root over N, and it is 2 by root over N. If it  $\alpha$  equal to 0, you use this one  $\alpha$  equal to not 0, you use this one.

So, when I say  $C(u,v)$  ok. So, that means  $C(0,0)$ ; when it is, then I use this one. Similarly,  $\alpha(v)$  is also the same 1 by root over of m; if it is, if it is the same, then it is root over of n and 1 by 2 by root over of n.

So, when u is u equal to 0, then and v is equal to 0  $\alpha(0)$ , this will be 1 by root over of N, and this will be 1 by root over of N. So, multiplication means, 1 by N if. So, I can say this is nothing but a 1 by N if both directions are the same dimension.

Now, let us say  $u$  has a dimension or let us say I have a signal  $x$  and  $y$ . If  $x$  has a dimension  $M$  minus 1, and  $y$  has a dimension  $N$  minus 1, it is not necessary. The 2 dimensions will be the same. And 2-dimension DFT will be the same. It may be different also. In that case, it will be 1 by root over of  $M$  into root over of  $N$ .

So, this is the normalization factor. And then, the rest are the same  $\cos \pi * 2x + 1$  divided by  $N$  into  $u$ . So, it is 2 dimensions, that is why, 2 cos function. Suppose it is a single dimension single cos function. So this is called DCT Discrete Cosine Transform. I can write down an algorithm to compute the discrete cosine transform from any signal. I have an image, let us say, 8 cross 8 images. So,  $m$  is equal to here all dimensions are the same. I can use DCT using this, and I can get an image  $u$  and  $v$   $u$ , which also varies from 0 to  $m$   $N$  minus 1, and  $v$  also varies from 0 to  $N$  minus 1.

Then, once I get an image, I know which are the highest power components and which are the lowest power components. So, I can quantize those in a high bit, and I can quantize that low-power component in a low bit. And I can get better image compression.

So, MPEG-4 and MPEG-3 all use DCT discrete cosine transform for image compression because it is better energy compaction. So, why does DCT give you better energy compaction or why was this formula used?

So, what is how; what is the difference between the DFT and DCT? What are we doing in the case of DCT?

(Refer Slide Time: 10:11)

Why DCT has better energy compaction

Let sequence  $x[n]$  which is zero outside of  $\{0, \dots, N-1\}$

Create a sequence  $y[n] = x[n] + x[2N-n-1]$

$$y[n] = \begin{cases} x[n], & 0 \leq n < N \\ x[2N-n-1], & N \leq n < 2N \end{cases}$$

Compute the  $2N$ -point DFT of  $y[n]$

$$Y[k] = \sum_{n=0}^{2N-1} y[n] e^{-j \frac{2\pi}{2N} kn}, \quad 0 \leq k < 2N$$

$$Y[k] = \sum_{n=0}^{N-1} y[n] e^{-j \frac{2\pi}{2N} kn} + \sum_{n=N}^{2N-1} y[n] e^{-j \frac{2\pi}{2N} kn}$$

So, I have a signal  $X[n]$ . And plus,  $X[n]$  varies from 0 to  $N$  minus 1. So, I create a sequence using  $x[n]$ . So,  $x[n]$ , I am creating sequence  $x[n]$  plus  $x[2N-n-1]$ . So, that means I am just doubling  $x[n]$ . I am doubling. So, if I take  $N$  point DFT, then  $x[n]$  is also period  $n$ . So,  $x$  or so, I take 0 to  $N$  minus 1 plus  $N$  to  $2N$  minus 1. So, this is my  $x[n]$ , and this is  $N$ ,  $N$  to  $2N$  minus 1 is my  $x$  of this one.

So, I take 2 complete periods. So,  $y[n]$  is equal to  $x[n]$ , when 0 to  $n$  minus 1. And I can say  $x[2N-n-1]$ . So, that is my signal, ok. Now, compute the. So, what I am saying I am not saying that, after  $N$  minus 1, my signal is 0. I am just repeating the same signal in 2 periods for a 2-period  $x$  of. Suppose I have a full length. Let us say I have an  $x[n]$  equal to 1 2 3.

So, what I think I am taking, I am taking  $y[n]$  is equal to 123, 123. So, if this is length is 3, and this is length is, you know, 3 plus 3 6. So,  $y[n]$  is a length of  $2N$ . Now, I compute the  $2N$ -point DFT of  $y[n]$ . Because the  $y[n]$  of the length of  $2N$ . So, I take the  $2N$ -point DFT of  $y[n]$ . So

$$Y[k] = \sum_{n=0}^{2N-1} y[n] e^{-j \frac{2\pi}{2N} kn} \quad \text{for } 0 \leq k < 2N$$

So,  $k$  varies from. So, I have a  $y$   $k$ , which also ranges from 0 to  $2N$  minus 1, ok. Now, say  $y[k]$  is equal to I put  $y[n]$  plus. So, I can divide this sum  $n$  equal to 0 to  $N$  minus 1. And  $n$

(Refer Slide Time: 13:29)

So, I can say this sum and those signals are the same. So, I can take common this part. So, this part remaining part is

$$Y[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{2N} kn} + \sum_{m=0}^{N-1} x[m] e^{-j \frac{2\pi}{2N} k(2N-1-m)}$$

So, n is nothing, but a m I put the n is nothing, but the index is multiplied with this positive  $j 2\pi$  by k m k m. Here, I said m equal to 0 to N minus 1 n also 0 to N minus 1. So, that is why I, instead of m write n. And then, this multiply minus  $j 2\pi / 2N k$  into  $2N$  this part and minus plus  $j 2\pi$  by  $2N$  into k.

Now, if you see this part  $e^{-j(2\pi/2N)/2kN}$   $2N$   $2N$  cancel. So, it is nothing but a minus it  $\cos 2\pi$  minus  $J \sin 2\pi$ . So, it is nothing but a 1. That is why I put this value equal to 1.

So, this value is 1. So, 1 and this value will be there. So

$$Y[k] = \sum_{n=0}^{N-1} x[n] e^{-j \frac{2\pi}{2N} kn} + e^{-j\pi k} \sum_{m=0}^{N-1} x[m] e^{j \frac{2\pi}{2N} km}$$

(Refer Slide Time: 17:43)

$$\begin{aligned}
 Y[k] &= \sum_{n=0}^{N-1} x[n] \left\{ e^{-j \frac{2\pi}{2N} kn} + e^{j \frac{2\pi}{2N} kn} e^{j \frac{2\pi}{2N} k(2N-1-n)} \right\} \\
 &= \sum_{n=0}^{N-1} x[n] e^{j \frac{2\pi}{2N} kn} \left\{ e^{-j \frac{2\pi}{2N} kn} + e^{j \frac{2\pi}{2N} kn} e^{j \frac{2\pi}{2N} k(2N-1-n)} \right\} \\
 &= \sum_{n=0}^{N-1} 2x[n] e^{j \frac{2\pi}{2N} kn} \cos\left(\frac{\pi}{2N} k(2n+1)\right) \\
 &= e^{j \frac{2\pi}{2N} kN} \sum_{n=0}^{N-1} 2x[n] \cos\left(\frac{\pi}{2N} k(2n+1)\right) \\
 &= e^{j\pi k} \sum_{n=0}^{N-1} 2x[n] \cos\left(\frac{\pi}{2N} k(2n+1)\right) \\
 &= e^{j\pi k} Y[k] \\
 C_x[k] &= \begin{cases} e^{-j \frac{\pi}{2N} k} Y[k] & 0 \leq k < N \\ 0 & \text{otherwise} \end{cases}
 \end{aligned}$$

Now, I said I want to express this part as a cos theta from cos theta is

$$\frac{e^{-j \frac{\theta}{2}} + e^{j \frac{\theta}{2}}}{2} = \cos\left(\frac{\theta}{2}\right), \text{ we know that.}$$



So, in that form, we want to know. Thus, 2 will be multiplied. That is why we have just made that form. So, what do we make? We commonly take the common of  $j\pi$  by  $2Nk$ . Then, here, this term is not there, which is why we multiply. And here, instead of  $2\pi$   $2Nk$ , I take 1 part. So, 1 will be there now. If you see it, it is minus e to the power e to the power minus j. I can say  $2\pi$  by  $2N$  into  $K$  n minus  $\pi$  by  $2Nk$ .

So, or I can say e to the power minus j by  $2N$  if I said the  $2\pi$  by  $2N$  if I take common  $K$  n minus k. And here also if I said that  $2\pi$  by  $2N$  common k n plus k minus k, sorry it is plus k here also it is j  $2\pi$  this also  $2\pi$  by  $2N$  if I say common e to the power minus j then, it will be k n plus k. So, I can say

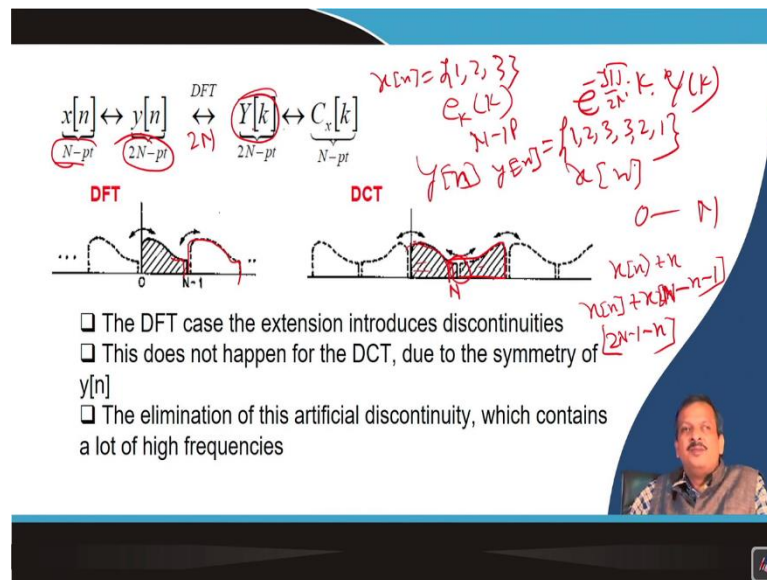
$$\frac{e^{-j\frac{\theta}{2}} + e^{j\frac{\theta}{2}}}{2} = \cos\left(\frac{\theta}{2}\right).$$

So, I can say it is nothing but a  $2 \cos \theta$ .. So,  $\theta$  by 2 will be there. So, it is nothing, but that is why  $\pi$  by  $2N$ ,  $\pi$  by  $2N$ ,  $\pi$  by  $2N$  into k into  $2N$  plus 1 cos and 2 will be multiplied.

So, I can say e to the power  $j\pi$  by  $2N$  into k, and this is my expression. So, this is my DCT coefficient. So, I can say the DCT coefficient is equal to this divided by minus  $j\pi$  by  $2Nk$  into  $y[k]$ , and this varies from 0 to n minus 1. So, I can say  $C_k$  has a value, which is k varies from 0 to  $N$  minus 1 outside that it is 0. So, what is what I gain? I said DCT is nothing, but if I take a sequence  $x[n]$ , I create a sequence by repeating  $x[n]$  2 periods and take the 2 point  $2N$  point DFT.

So, DCT is nothing but an  $e^{-j\pi/2kN}$  into  $y[k]$ . This is my DCT. So, expression, which I said in DCT coefficient, this is my DCT coefficient ok.

(Refer Slide Time: 21:37)



So, this is my DCT coefficient. So, why does it have better energy compression? So, if you see  $x[n]$  is the  $N$  point, I create  $y[n]$ , which is a  $2N$  point. I take the DFT, which is  $2N$  point, and I get  $y[k]$ . But from  $y[k]$ , I get  $C_k$  the DCT coefficient only  $N$  point by just multiplying with  $e^{-j\pi/2kn}$  into  $y[k]$ . If I know  $x[n]$ , I know  $y[n]$ , and I can compute  $y[k]$ .

Now, I get the DCT. So, basically, if I say DCT. So, as you know, let us know. This is my  $n$ . So,  $x[n]$  varies from 0 to  $N$ . So, this is my  $m$ . Let us say this is my  $N$ . So, I am taking  $2N$ . So, in the DFT case, I am taking only  $N$  minus 1. Then, I have taken another  $N$  minus 1, no problem, but there is a discontinuity.

But here I am taking  $2N$ , which is continuous and  $y[n]$  is symmetry. So, instead of  $x[n]$  like this  $x[n]$  plus  $x[n]$ , why have I taken  $x[n]$  plus  $x$  of  $2N - 2n - 1$ ? So,  $2N - 1 - n$ . So, I can say I just invert it. So, I take  $x[n]$ , and then that is why I create  $y[n]$ . So, the example that I have given in the initial, that if the  $x[n]$  is equal to 1, 2, 3 and  $y[n]$  this is not correct.

So, the  $y[n]$  will be instead of 1, 2, 3 it will be 1, 2, 3 then, 3 2 1 this will be the  $y[n]$  ok. That is why we get much more energy compaction. So, that is the use of DCT. So, basically, in image processing. So, if you are a student of image processing, you use this formula to compute DCT. If you want to apply for 1-dimensional DCT, you can use this formula, also.

So, first, you take the signal, do DCT, take the inverse DCT, and you get the spatial domain. So, in image compression, what do they do? They took the image and then took the DCT; in the DCT domain, they quantised that high-frequency high energy component with a low high bit rate and low energy component with a low bit rate. Then, transmitted on the receiver side, they again take the inverse DCT and get the spatial domain image.

So, that is, image compression is clear. So, this is a discrete cosine transform, and they are how their application is, how you can use discrete cosine transform, and what is the difference between DCT and DFT.

Now, you can also calculate DCT using DFT. Suppose I want to calculate DCT. I said no, I want to use DFT to calculate DCT. I can do that I have an  $x[n]$ , I will create  $y[n]$ , take the  $2N$  point DFT and multiply with; I will only take the  $y[k]$  0 to  $N$  minus 1 multiplied by  $e^{-j\pi/2kN}$ .

So, I can use DFT, I can calculate DCT also. So, if I give an example, let us say I have an image. I am giving you an image 3 by 3 image 3 pixel 3-pixel image. I told you to compute the DCT of that image you can do that. So, compute the DCT of the image you can do that.

(Refer Slide Time: 26:51)

**2D DCT**

• Corresponding 2D formulation

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

$u,v=0,1,\dots,N-1$

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

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

Handwritten calculations for a 3x3 image  $f(x,y) = \begin{bmatrix} 1 & 2 \\ 2 & 1 \\ 1 & 1 \end{bmatrix}$ :

- $\alpha(0) = \frac{1}{\sqrt{2}}$
- $\alpha(u) = \frac{1}{\sqrt{3}}$  for  $u \neq 0$
- $C(0,0) = \alpha(0)\alpha(0) \sum_{x=0}^2 \sum_{y=0}^2 f(x,y) = \frac{1}{2} \cdot \frac{1}{2} \cdot (1+2+2+1+1+1) = \frac{1}{4} \cdot 8 = 2$
- Other calculations for  $C(0,1), C(0,2), C(1,0), C(1,1), C(1,2), C(2,0), C(2,1), C(2,2)$  are shown.

So, What will I do? Let us take a slide here. So, I said this is my formula for DCT calculation. Now, I said I have taken an image, which is 3 pixels by 3 pixels, let us say or 2 pixels by 6 1 2 2 1. I said compute the DCT of that image.

So, the first pixel is 1, the second pixel is 2, and the third pixel is 1 and 1. So, this is my 2 by 2 image, a 2 pixel by 2 pixel image, and I have to compute DCT. So, how do I do that? I know  $C[u,v]$ . So,  $\alpha(u)$  is equal to  $1/\sqrt{2}$  if  $u$  is equal to 0 and  $1$  if  $u$  is not equal to 0. Similarly,  $\alpha(v)$  value is either  $1/\sqrt{2}$  or  $1$ . If it is  $u$  equal to 0, that is  $1/\sqrt{2}$ . If it is  $u$ , not 0. Similarly,  $\alpha(v)$  value is also either  $1/\sqrt{2}$  or  $1$ .

So, it is nothing but a 1 or when  $u$  is not equal to 0. So, I can say  $\alpha(u)$  value is either  $1/\sqrt{2}$  or  $1$ . If it is  $u$  equal to 0, that is  $1/\sqrt{2}$ . If it is  $u$ , not 0. Similarly,  $\alpha(v)$  value is also either  $1/\sqrt{2}$  or  $1$ .

So, if I say  $C(0,0)$   $u$  equal to 0  $v$  equal to 0. So,  $\alpha(0)\alpha(0)$  this is also  $1/\sqrt{2} \times 1/\sqrt{2}$   $u$  equal to 0  $v$  equal to 0. So,  $1/\sqrt{2} \times 1/\sqrt{2}$   $u$  value is 0  $v$  value is 0. So,  $u$  value is 0 means,  $\cos(0)$  and  $\cos(0)$ . So, I can say both  $\cos$  are 1.  $\cos(0)$  is 1. So, it is nothing but a  $1/\sqrt{2}$  into  $1/\sqrt{2}$  summation of  $x$  equal to 0 to 1 summation of  $y$  equal to 0 to 1 of  $x \cdot y$ .

Because into 1, so, it is nothing but a

$$C(0,0) = \left(\frac{1}{\sqrt{2}}\right) \left(\frac{1}{\sqrt{2}}\right) [f(0,0) + f(0,1) + f(1,0) + f(1,1)]$$

So, it will be for

$$C(0,0) = \frac{1}{2} [1 + 2 + 1 + 1]$$

So, it is nothing but a 6. So, half into 6 is equal to 3. So, I get matrix value  $C(0,0)$  is equal to 3 then, I get  $C(0,1)$   $u$  equal to 0  $v$  equal to 1 then, I get  $C(1,0)$   $u$  equal to 1  $v$  equal to 0 then, I get  $C(1,1)$ . So, I calculate  $C(0,1)$   $C(1,0)$   $C(1,1)$ . I get the DCT value. This is the matrix. So, what will be  $C$  let us say  $C(0,1)$  if I want to calculate.

(Refer Slide Time: 31:21)

$$\begin{aligned}
 C(0,1) &= \frac{1}{\sqrt{2}} \alpha(0) \sum_{n=0}^1 \sum_{m=0}^1 f(n,m) \cos\left(\frac{\pi(2n+1)}{2} \frac{\pi(2m+1)}{2}\right) \\
 \alpha(0) &= \frac{1}{\sqrt{2}} \\
 \alpha(0) &= 1 \\
 C(0,1) &= \frac{1}{\sqrt{2}} \sum_{n=0}^1 \sum_{m=0}^1 f(n,m) \cos\left(\frac{\pi(2n+1)}{2} \frac{\pi(2m+1)}{2}\right) \\
 &= \frac{1}{\sqrt{2}} \sum_{n=0}^1 f(n,0) \cos\frac{\pi}{4} + f(n,1) \cos\frac{3\pi}{4} \\
 &= \frac{1}{\sqrt{2}} \left( f(0,0) \cos\frac{\pi}{4} + f(0,1) \cos\frac{3\pi}{4} + f(1,0) \cos\frac{\pi}{4} + f(1,1) \cos\frac{3\pi}{4} \right)
 \end{aligned}$$

So, I want to calculate  $C(0,1)$ . So,  $u$  is equal to 0  $v$  is equal to 1. So, what is my formula? So, I know  $\alpha(0)$   $u$  is equal to 0  $1$  by root 2, which is not. So, the  $\alpha(v)$  for  $v$  is equal to 1. So, this is nothing but a 1. So, my constant term  $C(0,1)$  is  $1$  by root 2 into 1, and then my summation  $x$  equals 0 to 1,  $y$  equals 0 to 1. What is there?  $f$  of  $x$   $y$  is ok. What is their cos? What is the  $\cos 2\pi$  into 2  $x$  plus 1 into  $u$ .

So,  $\pi$  into 2  $x$  plus  $u$  is equal to 0  $u$  is 0 I have taken  $u$  is 0. So,  $u$  is 0 means, this is 1. And this will be  $\cos \pi$  into 2  $y$  plus 1  $\cos \pi$  into 2  $y$  plus 1 divided by 2  $n$  2  $n$  means 2 into 2 2  $n$  into  $v$  is equal to 1. So, I get 1 by root 2  $x$  equal to 0 to 1  $y$  equal to 0 to 1  $f$  of  $x$   $y$   $\cos$  of  $\pi$  into 2  $\pi$  2  $y$  plus 1 divided by 4. So, which is nothing, but I just put 1 by root 2. This sum  $x$  equals 0 to 1, and let us remain.

So, let us say this sum is do  $y$  equal to 0. So, I can say this is nothing, but a  $f$  of  $x$  0 into  $\cos \pi$   $y$  is equal to 0. So, if this is 0, this is 1. So,

$$C(0,1) = \frac{1}{\sqrt{2}} \left( f(0,0) \cos\left(\frac{\pi}{4}\right) + f(0,1) \cos\left(\frac{3\pi}{4}\right) + f(1,0) \cos\left(\frac{\pi}{4}\right) + f(1,1) \cos\left(\frac{3\pi}{4}\right) \right)$$

So, I get that. I can write a program instead of doing hand calculations. So, if I said the 8 cross 8 matrix DCT. So, instead of that, I write a for loop only for the cos part and a for loop only for all the conditions. So, it's very easy. You can write a C program for that DCT calculation.

Thank you.