

Real-Time Digital Signal Processing
Prof. Rathna G. N.
Department of Electrical Engineering
Indian Institute of Science – Bengaluru

Lecture – 48
M3U30 - Discrete Cosine Transform - II

Welcome back to Real-Time Digital Signal Processing Course. So, we are discussing about Discrete Cosine Transform in Image Processing. So, we will continue in today's class also DCT basically.

(Refer Slide Time: 00:37)



So, in the last class we discussed about discrete cosine transform way it is being used. We know that it is in jpeg compression.

(Refer Slide Time: 00:50)

JPEG Modes



Sequential Mode:

- Each image is encoded in a single left-to-right, top-to-bottom scan.
 - The technique we have been discussing so far is an example of such a mode, also referred to as the **Baseline Sequential Mode**.
 - It supports only 8-bit images as opposed to 12-bit images as described before.



Rathna G N



So, today we will continue with the some of the properties what we are discussing about jpeg? So, here we will see the jpeg modes, so that is the first one is we can have sequential mode that is each image is encoded in a single left-to-right, top-to-bottom scan. So, this technique we have been discussing so far in is an example of such a mode also referred to as the baseline sequential mode.

And it supports only 8-bit images as opposed to 12-bit images, as we have discussed it earlier.

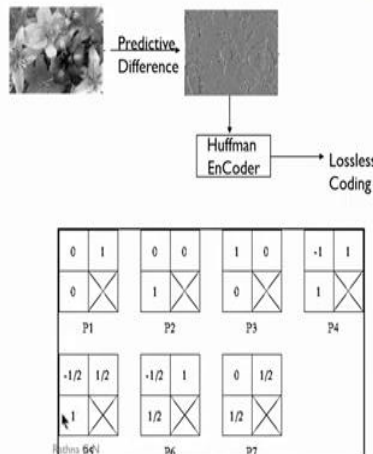
(Refer Slide Time: 01:28)

JPEG Modes (2)



Lossless Mode:

- Truly lossless
- It is a predictive coding mechanism as opposed to the baseline mechanism which is based on DCT and quantization (the source of the loss).
- Here is the simple block diagram of the technique:



Coming to the second mode so we will see that it is lossless mode that is truly lossless what we call it. What do you mean by that? So, we will be doing the predictive coding mechanism opposed to the baseline mechanism which is based on DCT and quantization that is source of

the loss. So, here the simple block diagram of the technique is what it is shown in this. So, what we have is? This is the image what we have it.

So, we will find the predictive difference so which is shown in this diagram. Then we will do the Huffman encoder. Then we will be calling this as the lossless coding. So, as you can see in this figure, there are seven predictive coding techniques which is going to be used. Here, it is going to be unknown parameter in the P1 and the surrounding images pixels, what we call it which are 0, 0 and 1.


In this case, in the P2 mode, it is going to be 1, 0, 0 and so on what will be using it. As you will be seeing in the P5, we have 1 minus half and then half same way with P7 we have half 0 and then half.

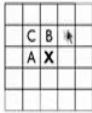
(Refer Slide Time: 02:56)

Lossless Mode

Predictive Difference:

- For each pixel a predictor (one of 7 possible) is used that best predicts the value contained in the pixel as a combination of up to 3 neighboring pixels.
- The difference between the predicted value and the actual value (X) contained in the pixel is used as the predictive difference to represent the pixel.
- The predictor along with the predictive difference are encoded as the pixel's content.
- The series of pixel values are encoded using Huffman coding





Predictor	Prediction
P1	A
P2	B
P3	C
P4	A+B-C
P5	$A + (B-C)/2$
P6	$B + (A-C)/2$
P7	$(A+B)/2$

Notes:

- The very first pixel in location (0, 0) will always use itself.
- Pixels at the first row always use P1,
- Pixels at the first column always use P2.
- The best (of the 7) predictions is always chosen for any pixel.

5
Rathna G N

So, we will see predictive coding how it is going to be used? Only we are going to consider the difference in this case. So that is for each pixel, a predictor that is what we shown one of the seven which is listed in the table here, is possible. So that is used best predicts the value contained in the pixel as a combination of up to three neighboring pixels, as you are seeing it. This is the pixel, so, it can be A C B R, these three.

So, the predictor with respect to that if it is in the P1 only A what you will be selecting it P2 B and P3 C. And then the other combinations what you can see the thing up to P4 to P7 with respect to A B and then C. What we say is the difference between the predicted value and the

actual value that is X contained in the pixel is used as the predictive difference to represent the pixel instead of directly sending the value of that pixel.

So, we will be computing predict the difference and then what it belongs to, so that numbers of bits are going to be less to represent them. The predictor, along with the predictive difference are encoded as the pixels content. So, the series of pixel values are encoded using, as in the previous slide it depicted using the Huffman coding. So, one has to take care of it that is what is this notes?

The very first pixel in location 0, 0 will always use itself. And pixels at the first row always use P_1 that is A pixel in row. They will use P_1 and pixel at the first column, always in this use, P_2 technique. And then the best that is what of the seven predictions is always chosen for any of the pixel in this portion. So, here it is selected as 5 by 5 pixel what it is shown? But usually, we will take 8 by 8 pixel as the sample.

(Refer Slide Time: 05:11)

JPEG Modes

Progressive Mode: It allows a coarse version of an image to be transmitted at a low rate, which is then progressively improved over subsequent transmissions.

- **Spectral Selection :** Send DC component and first few AC coefficients first, then gradually some more ACs.

Image Pixels

Spectral Selection:

First Scan: [Red bar]

Second Scan: [Green bar]

Third Scan: [Green bar]

Nth Scan: [Green bar]

6

Rathna G N

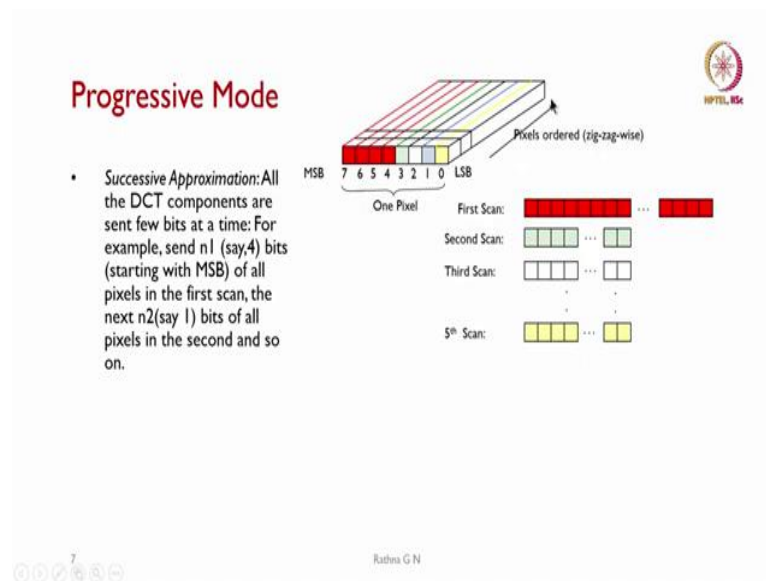
So then how we are going to continue with the progressive mode? So, from the previous one predictive, so, we will see progressive how it is going to go? So, it allows a coarse version of an image to be transmitted at a low rate which is then progressively improved over subsequent transmissions. So, as you will be seeing, spectral selection send DC component and first few AC coefficients first.

And then gradually some more AC coefficients what you will be selecting it? So, what is it? So, you will be seeing that in the 8 by 8. So, these are your red depicts your DC coefficients,

so, first scan you will be sending all the DC coefficients. In the second scan, what you will be seeing it. So, you will send some few AC coefficients. Like that you can send some more AC coefficients which are dominant.

As you can see the thing the last one is yellow that is nth scan you will be sending these coefficients. This is how in the progressive mode, you will be sending the coefficients.

(Refer Slide Time: 06:27)



So, as you can see that in the progressive mode, successive approximation, all the DCT components are sent few bits at a time. For example, send n_1 what it says is 4. So, you are seeing that these are the four pixels which are sent bit starting with MSB of all pixels in the first scan. The next n_2 say in this case only 1-bit what it will be sent here? That is the complete length of it.

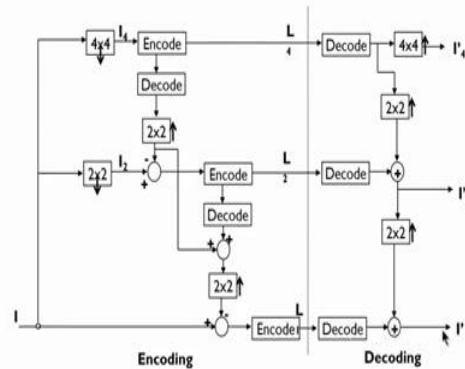
So, you will be seeing that this 4-bits that whatever the length of it is going to be our first scan. And then the second scan, what you are seeing it here. And then bits of all pixels in the second scan and then so on what you will be doing it. So, you are seeing that till the end that is LSB bit, so that is one pixel for the rest of the thing what you will be sending it.

Last one is that is in this case it becomes fifth scan which you will be transmitting yellow. And then this is, what you have ordered the pixels in the zigzag wise, basically on this axis.

(Refer Slide Time: 07:52)

Hierarchical Mode

- Used primarily to support multiple resolutions of the same image which can be chosen from depending on the target's capabilities.
- The figure here shows a description of how a 3-level hierarchical encoder/decoder works:



So, the other mode is the hierarchical mode What you have to incorporate. So, this is used primarily to support multiple resolutions of the same image which can be chosen from or depending on the targets capabilities. So, the figure shows here a description of how a 3-level hierarchical, encoder or decoder is going to work. So, what is it? So, you have I is the image input here, so, you have a 4 by 4. You call it as I_4 then do the encoding of it.

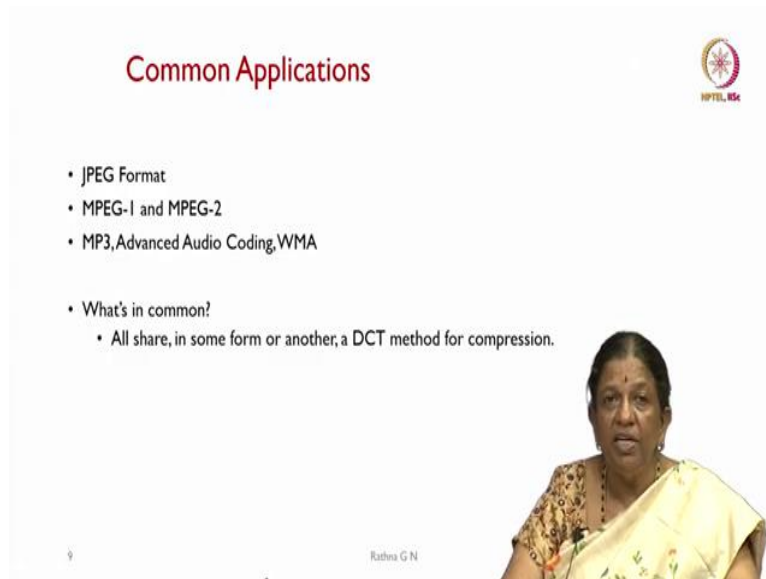
And then you will call it as L_4 basically. And then you will be at the decoder end that is you will be doing the decode and do the conversion to 4 by 4 and then output will be I_4 what you will be getting it? So, the same thing encoded at the encoding side also, you can decode and then convert it into 2 by 2. So, you will be seeing that the next 2 by 2 whatever image thing what you have taken you call it as I_2 .

Then find the difference between these two. And then you will be encoding this value and then calling it as L_2 and you will be transmitting it. So, the same thing the next level is, you are passing this encoder to decode and then you will be doing the addition of the thing from whatever 2 by 2 what you have it. And then you will call it as the 2 by 2 and this will be passed on to the third level. So, which will be subtracted with respect to the original I .

And then do this encode and call it as L alone and again at the decoding place you are doing the decode of these three level, whatever encoded message. So, we got I_4 dash now the same thing what you will be taking it and then doing it as 2 by 2 then you will be adding with whatever you are getting from L_2 and that you call it as I_2 dash. The same thing will proceed

to the next stage that is 2 by 2. And then whatever decoded at this last stage is going to be added and then you will be calling it as I dash.

(Refer Slide Time: 10:41)



Common Applications

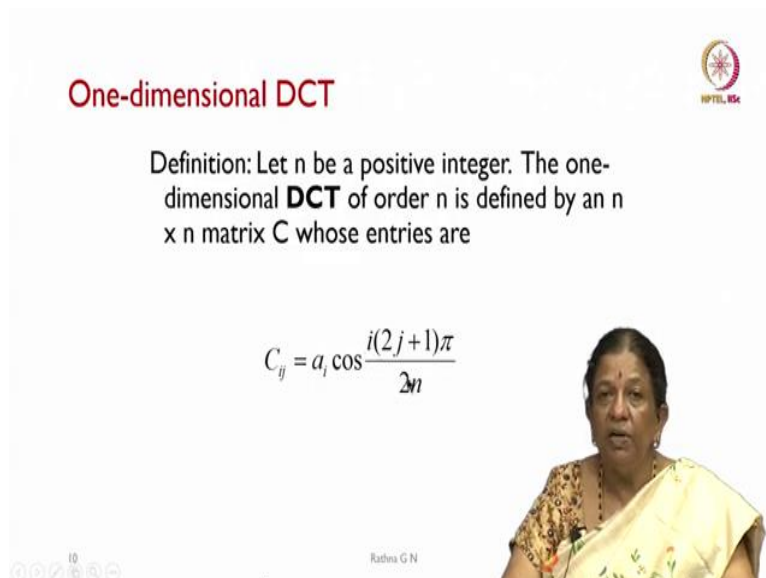
- JPEG Format
- MPEG-1 and MPEG-2
- MP3, Advanced Audio Coding, WMA
- What's in common?
 - All share, in some form or another, a DCT method for compression.

9 Rathna G N

So, where this jpeg common applications lie, so, one of the application is the jpeg format, what we have it? And in MPEG-1 this is the joint distinct group. What you have it? This is MPEG is the moving average, what you have 1 and then 2 and then MP3 is the audio player and even in the advanced audio coding and then in WMA most of them use our DCT method for compression.

So that is what, what is common in all these applications what you have the DCT coming into the thing?

(Refer Slide Time: 11:33)



One-dimensional DCT

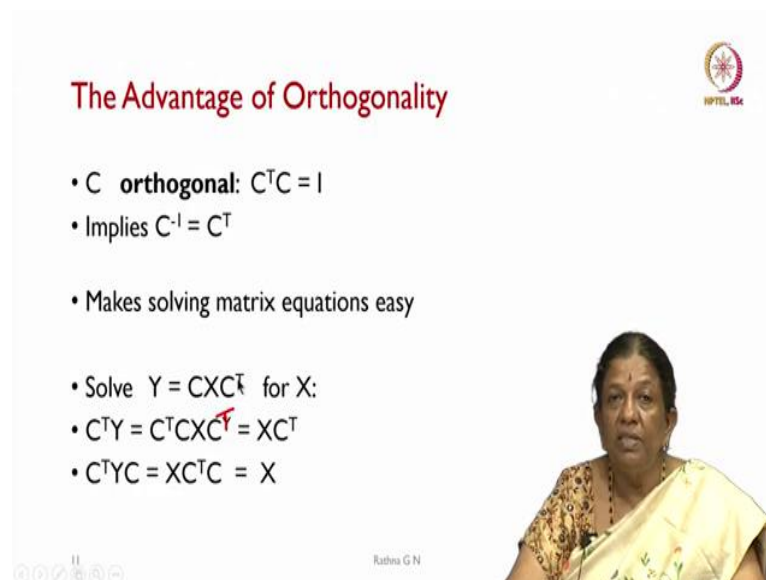
Definition: Let n be a positive integer. The one-dimensional **DCT** of order n is defined by an $n \times n$ matrix C whose entries are

$$C_{ij} = a_i \cos \frac{i(2j+1)\pi}{2n}$$

10 Rathna G N

So now, we will see a little bit on one-dimensional DCT. How does it look like? So, we say n be a positive integer. The one-dimensional DCT of order n is defined by an n by n matrix, C will be using A and C interchangeably. So, wherever C matrix in some of the applications, we may call it as A matrix. So, whose entries are given by that is C_{ij} is given by $\frac{1}{\sqrt{2n}} \cos \left(\frac{(i-1)(j-1)\pi}{2n} \right)$.

(Refer Slide Time: 12:17)



The Advantage of Orthogonality

- C orthogonal: $C^T C = I$
- Implies $C^{-1} = C^T$
- Makes solving matrix equations easy
- Solve $Y = CXC^T$ for X :
- $C^T Y = C^T CXC^T = XC^T$
- $C^T Y C = XC^T C = X$

II
Rathna G N

What is the advantage of orthogonality? We will see the thing if C is orthogonal then we say that C transpose C is going to be identity matrix I that implies that when you take a inverse of C which is equivalent to C transpose itself. So, this makes a solving the matrix equation easy that is shown here that is Y is C into X into C transpose for X . Then what happens to C transpose Y ?

When I take C transpose of Y and then C transpose on the right hand side also. Then it becomes C transpose C into X into C a transpose here which is nothing but X into C transpose. If you multiply with C transpose Y into C is nothing but X into C transpose C which becomes X . So now, you will be seeing that how Y and X can be computed interchangeably, basically that if our matrix is orthogonal.

So, X will be what is it? C transpose Y into C whereas original Y is nothing but CX into C transpose.

(Refer Slide Time: 13:41)

One-dimensional DCT

The discrete cosine transform, C, has one basic characteristic: it is a real orthogonal matrix.

$$C = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \dots & \frac{1}{\sqrt{2}} \\ \cos \frac{\pi}{2n} & \cos \frac{3\pi}{2n} & \dots & \cos \frac{(2n-1)\pi}{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \cos \frac{(n-1)\pi}{2n} & \cos \frac{(n-1)3\pi}{2n} & \dots & \cos \frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

$$C^{-1} = C^T = \sqrt{\frac{2}{n}} \begin{bmatrix} \frac{1}{\sqrt{2}} & \cos \frac{\pi}{2n} & \dots & \cos \frac{(n-1)\pi}{2n} \\ \frac{1}{\sqrt{2}} & \cos \frac{3\pi}{2n} & \dots & \cos \frac{(n-1)3\pi}{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\sqrt{2}} & \cos \frac{(2n-1)\pi}{2n} & \dots & \cos \frac{(n-1)(2n-1)\pi}{2n} \end{bmatrix}$$

Rathna G N



So, it does help in computation that is both the synthesis and analysis equation becomes easy for us to do the computation. Now, we will see for the discrete cosine transform that is we have taken here n length, basically. So, one basic characteristic is it is a real orthogonal matrix. So, what is it? Which is given by $C = \sqrt{2/n}$ into the all the first columns are 1 by $\sqrt{2}$. So, the next one is $\cos \pi$ by $2n$ and then $\cos 3\pi$ by $2n$ and so on.































So, the last one entry will be $\cos (2n-1)$ into π divided by $2n$. And in the last row, what we have? $\cos (n-1)$ into π by $2n$ and so on and the last entry will be $(n-1)$ into π divided by $2n$. Now, we will see either C inverse C transpose how does it look like? That is $\sqrt{2/n}$ and we know that row becomes column in this case and you will be seeing $\cos 2\pi$ by n and then this row is becoming this column.

And you will be seeing this last column entries \cos of $(n-1)$ into π by $2n$ and so on and then this is the last n by n th entry is the same as the original.

(Refer Slide Time: 15:22)


DCT Algorithm Classification

- Direct 2-D Method
 - The 2-D transforms, DCT and IDCT, to be applied directly on the $N \times N$ input data items
- Row-Column Method
 - The 2-D transform can be carried out with two passes of 1-D transforms
 - The separability property of 2-D DCT/IDCT allows the transform to be applied on one dimension (row) then on the other (column)
 - Requires $2N$ instances of N -point 1-D DCT to implement an $N \times N$ 2-D DCT



13

Rathna G N



So, how we are going to do the classification of our DCT algorithm? So, either we can do direct 2-D method. This 2-D transforms what we call it as both DCT and IDCT to be applied directly on the N by N input data items or we can go with row-column method that is, the 2-D transform, can be carried out with two passes of 1-D transforms. So, we will be applying the separability property of 2-D DCT and IDCT allows us to transform apply in this 1-D fashion.

That is one dimension row then on the other column. So, this requires what we say $2N$ instances of N -point 1-D DCT. So, to implement an N by N 2-D DCT that is one of the advantage of using it and then what we call it is, This computation is in place. So, what do I mean by in places? So, whatever the original your matrix, what it has it? So, same thing can be replaced with the new data.

So that I need not have more storage for this computation and it is much faster also we will see in a while.

(Refer Slide Time: 16:51)

Row-Column Decomposition



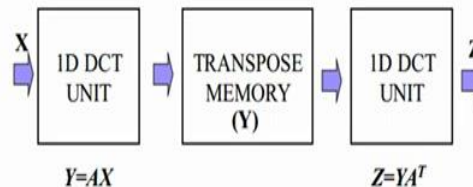
Separable,
row-column decomposition

$$a(k, n) = \sqrt{\frac{2}{N}} c(k) \cos\left[\frac{2\pi(2n+1)k}{4N}\right]$$

$$k, n = 0, 1, \dots, N-1$$

$$Z = AXA^T$$

$$c(0) = \sqrt{\frac{1}{2}} \text{ and } c(k) = 1 \text{ for } k \neq 0$$



Rathna G N

So now, how we are going to do the row column decomposition? So, we said that separable row column decomposition. What will be using it? So, in the matrix form what it is given? As $Z = AX$ into A transpose what we have it and then a k of n is nothing but root 2 by N into c of k N is the order what will be taking it? And then these are the $\cos 2\pi$ into $2n + 1$ into k divided by $4N$.

So, these are the coefficients what we have it, k and n will be varying between 0 to $n - 1$. And we know the first coefficient C of 0 is root 1 by 2 and then the other coefficients $k = 1$ to $n - 1$ that is what, what it says? For k is not equal to 0, c of k coefficient is going to be 1. As you can see in the previous this thing, you will be seeing that this is root 2 by n what we have taken the thing so, this is 1 by root 2.

So, you will be seeing that the first C 0 is 1 by root 2 basically, so which we can take it as 1 by root 2 and then rest of them we can assume it as 1. Then what happens to our equation? So, X is the input our 1-D DCT unit. So, what we are going to compute here? Y is equal to the first stage of the thing A into X . Then we will be doing the transpose the memory of Y so, why we have to do the transpose? So, you will be seeing that this is my transpose of Y .

Here it is C what it has been done? Here, A into X what will be transposing. Then I will do the 1-D DCT same unit. I can apply 1-D DCT unit then I will be getting Y into A transpose. So that is my Z what it is shown here. So, this is how we do the row-column decomposition and then computation.

(Refer Slide Time: 19:15)

Straight Forward Approach



- Carry out the computation as full-matrix-vector multiplications
 - 1-D transform requires N^2N multiplications and $N^2(N-1)$ additions
 - 2-D transform requires N^2N^2N multiplications and $N^2N^2(N^2(N-1))$ additions
 - Although requiring the most number of operations, this method is very regular
 - Most suitable for vector processors or deeply pipelined architectures for high PE utilization
 - 1-D fast algorithms $\Rightarrow O(N^3 \log N)$
 - 2-D fast algorithms $\Rightarrow O(N^6 \log N)$



Rathna G N




Now, why we have to use this? So, carry out the computation as a full matrix, vector multiplications. We know that 1-D transform requires N into N multiplications which is nothing but order of N squared multiplications and then N into $N - 1$ additions what we need it. So, this also order of N squared, whereas our 2-D transform requires, as we know that all is length N which is going to be order of N to the power of 4.

And then multiplications and this one order of N to the power of 4 that is N into N into $N - 1$ additions what we need it if we are using direct 2-D transform. So, although requiring most number of operations, this method is very regular and more suitable for vector processors or deeply pipelined architecture. What we have discussed for high processor engine, basically utilization.


So, what is it? So, we can have 1-D fast algorithms. We can just like our FFT we can configure our DCT also for the fast algorithm which brings down the order of $N \log N$ or instead of order of N squared what we can have computation? Whereas a 2-D fast algorithm we can incorporate it in order of N into $N \log N$ always that is order of N squared $\log N$. What will be getting directly if we are doing the 2-D algorithm? So, one of the applications I will be showing that how this can be incorporated later.

(Refer Slide Time: 21:21)

Image Compression



- Image compression is a method that reduces the amount of memory it takes to store in image.
- We will exploit the fact that the DCT matrix is based on our visual system for the purpose of image compression.
- This means we can delete the least significant values without our eyes noticing the difference.



16


Rathna G N

So now, we will see why we need the compression we have been discussed in the last class also. So, there are different ways of doing the compression. So, we know that is a method that reduces amount of memory takes to store in image. So, we discussed in the last class so, with the 3 millisecond the data, how much video we have to store it? So, how we can reduce it? So that DCT matrix is based on our visual system for the purpose of image compression.

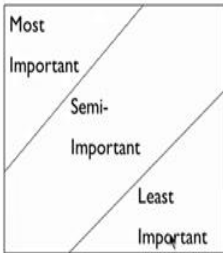

And this means we can delete the least significant values without our eyes noticing the difference. There are different ways of doing the compression.

(Refer Slide Time: 22:10)

Image Compression



- Now we have found the matrix $Y = C(CX^T)^T$
- Using the DCT, the entries in Y will be organized based on the human visual system.
- The most important values to our eyes will be placed in the upper left corner of the matrix.
- The least important values will be mostly in the lower right corner of the matrix.

17

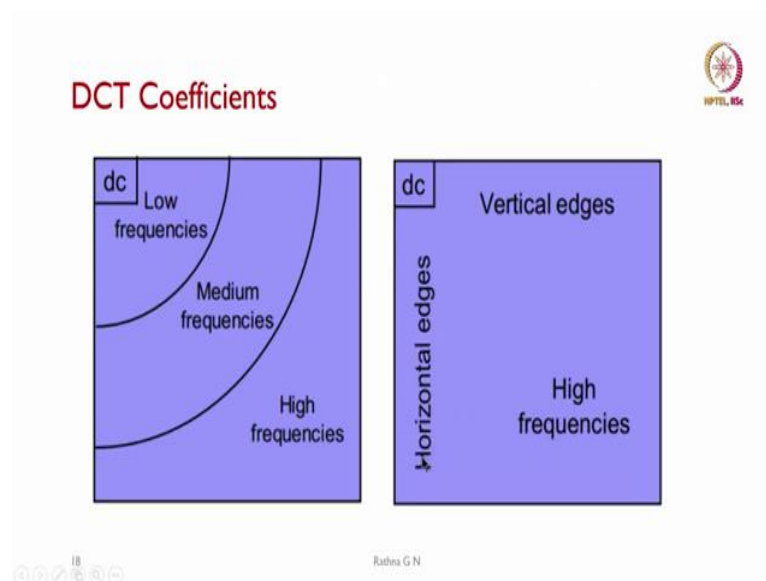
Rathna G N

So, the first one we have discussed we will just go for one more time to see the thing. So, what is it? We have found this matrix that is Y is C transpose whole transpose what we wanted? And

we say in this case this is the most important part in our image, what it is required. And these are semi- important and these are the least important values what we are interested in?

So, using the DCT the entries in Y will be organized based on the human visual system. The most important values to our eyes will be placed in the upper left corner of the matrix. And the least important values will be mostly in the lower right corner of the matrix, as you are seeing it here.

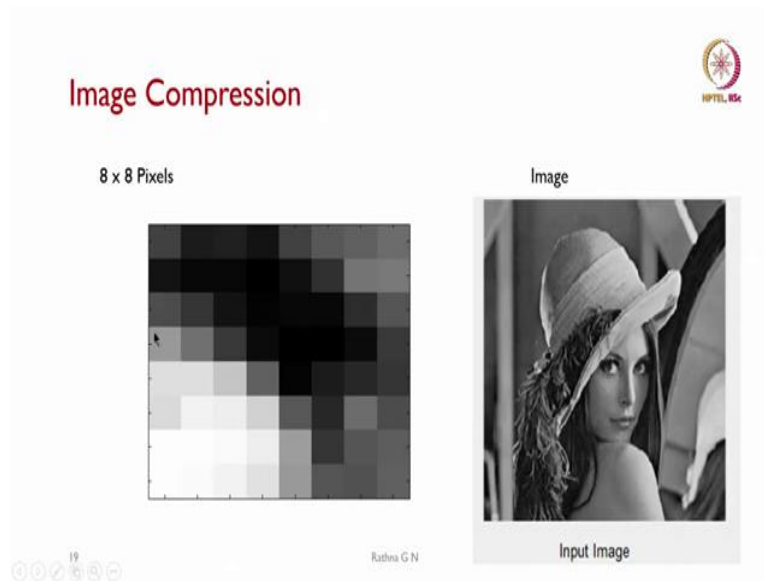
(Refer Slide Time: 23:01)



So, the next one to show that what are the DCT coefficients, how it is ordered? So, you will be seeing that the first corner you will have the DC in terms of frequencies. DC component is here and rest of the frequencies that is low frequencies which are present in the, this domain same as our previous one we have the medium frequencies and these are the high frequencies. Most of us know that high frequencies is a noise so that we can eliminate that.

And then, when we are looking at some of the edges, what we are going to want to put it in the frequency domain? That is in the DCT coefficients. So, we will have the DCT and the vertical edges are represented in the horizontal coefficients, whereas the horizontal edges values are represented as our column wise in this and all these are high frequencies what we will be representing it.

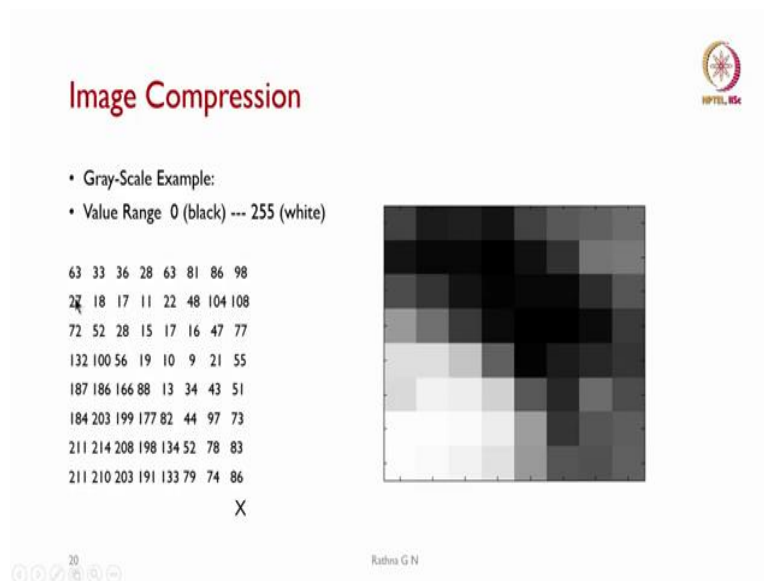
(Refer Slide Time: 24:11)



So, coming to the image compression so, you will be seeing that it is 8 by 8 pixels. So, this is the gray-scale what we have taken the thing that is, some of them are white some of them are black and then some of them are grey. So, this is how the pixels can be taken or one of the best way of doing it is take a checker board which has both white and black components only. And then see how the compression is going to look like?

And then we have taken an image here. All of you know that this is an input image Lena which is there in the Matlab. You can choose this or from anywhere you can download different images and then you can represent. So, we will see the compression how it is going to have it?

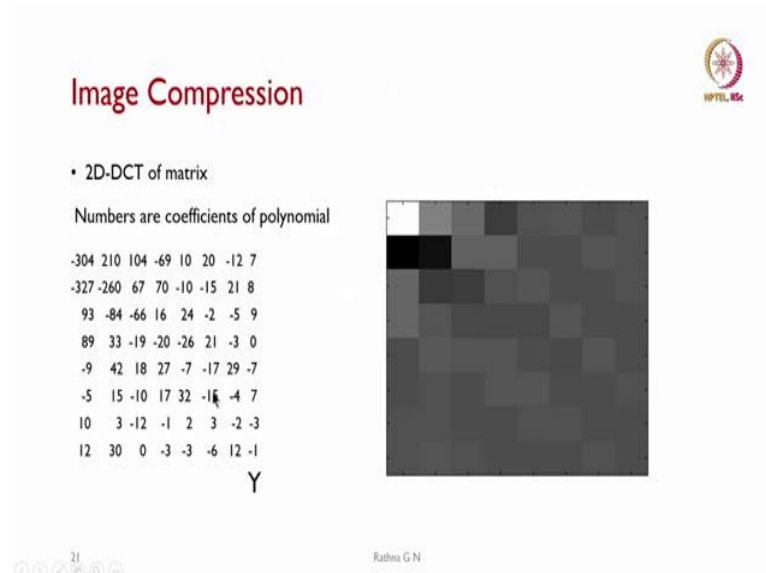
(Refer Slide Time: 25:03)



The first one is, we can do a gray-scale example what it is taken. So, the value of what will be representing is 0 to 255, so, 0 represents black and 255 represent white and in between values,

will be different levels of grey what will have it? So, as an X what you will be seeing is? These are the values. What you will have it in the matrix? So, you will be seeing some of them are 214, 208, 198 and other things. The first one is 63.

(Refer Slide Time: 25:45)



So now, we will have will do the so, 2-D DCT of matrix what we are going to do it. So, numbers are coefficients of a polynomial, what we will take it? And this is after doing the 2-D DCT for this X value will be getting Y. So, these are the values what it is represented. So, we can see that this is DC – 304 which is what we call it as 255 white and then you will be seeing that they will be having different values approximately or this is a – 327 and 260.

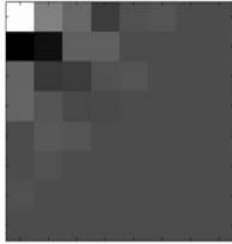
What you will be seeing somewhere here. So, some of them are that is represented as black and then you will be seeing rest of the thing. And then you are seeing that some of them have very less values.

(Refer Slide Time: 26:51)

Image Compression

- Cut the least significant components

-304	210	104	-69	10	20	-12	0
-327	-260	67	70	-10	-15	0	0
93	-84	-66	16	24	0	0	0
89	33	-19	-20	0	0	0	0
-9	42	18	0	0	0	0	0
-5	15	0	0	0	0	0	0
10	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0



As you can see, we save a little over half the original memory.

22

Rathna G N

So, what is it? I can put a threshold, so that is above that I will be making them zeros or below them I will be making zeros and above threshold, I will be making them whatever they have it. So, what is the threshold? You can put the thing here, as you can see that the values are less than arbitrarily here it is chosen. As you can see here, I have a 12 here, 30. So, some of them are getting eliminated.

One way of doing the thing is, I can put a threshold saying that above any value 10 or below that what I am going to make them zeros. I can do that way or one corner of it. What I can do it? Because these are going to be zeros. So, what you are selected is? From here you have made them zeros, basically, all of the lower side of the matrix. So, this is one way of selecting rest of the what we see?

We are going to keep the coefficient as it is that is cut the least significant components. That is what it is done. That is we save a little over half the original memory by doing it.

(Refer Slide Time: 28:11)



So, this is one way of doing our compression. So, in the next class we will see other ways that is, will use the quantization matrix so, to say that will be dividing by that quantization matrix and then we will discuss that how it is done? And then we will take the demo of the DCT also. So, thank you and then happy listening will meet in the next class.