

Digital image processing

Prof. P. K. Biswas

Department of Electronics and Electrical Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture - 15

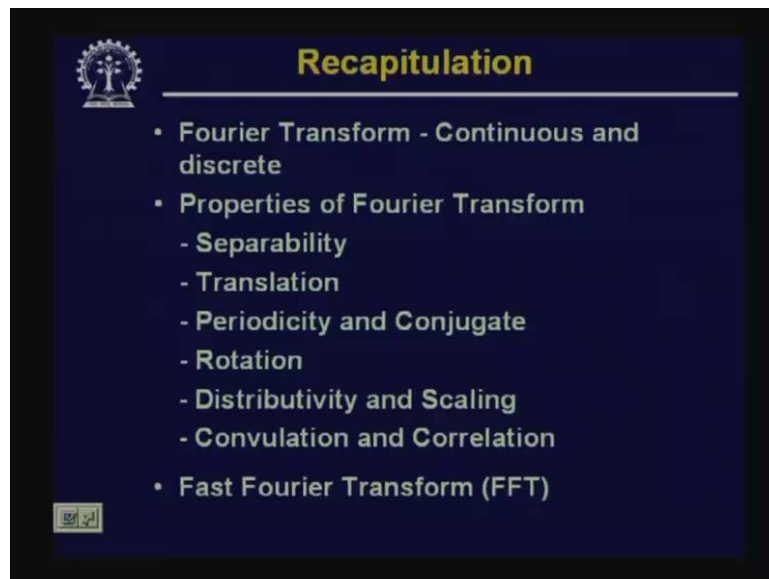
Discrete Cosine Transform

Walsh Transform

Hadamard Transform

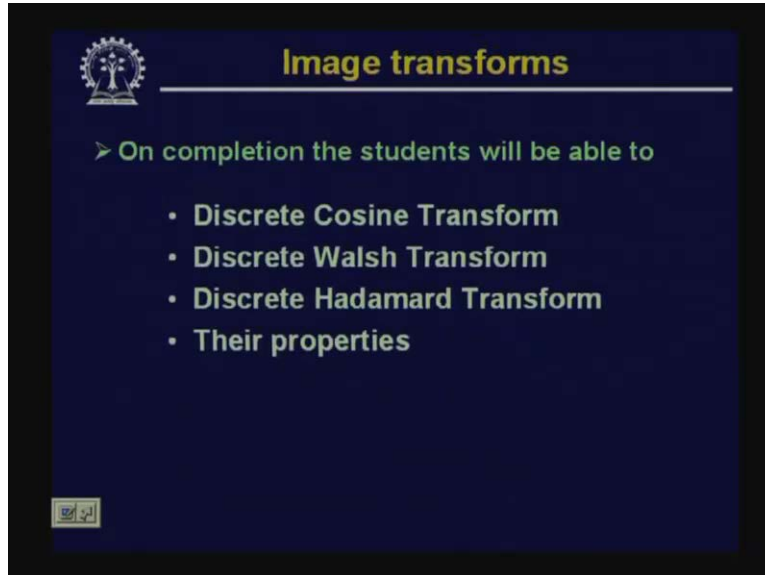
Welcome to this lecture on digital image processing. In our last class, we have discussed about the discrete Fourier transform.

(Refer Slide Time: 00:54)



We have seen both the continuous domain as well as the discrete domain of the Fourier transformation. We have seen the properties of the Fourier transform; the properties, specifically the separability, translation property, periodicity and conjugate property, the rotation property, distributivity and scaling and convolution and the correlation property and then finally, we have seen a fast implementation of the discrete Fourier transform which we have said as the fast Fourier transform or FFT operation.

(Refer Slide Time: 1:36)



In today's lecture, we will talk about some other transformations in the digital domain. We will talk about the discrete cosine transform, we will talk about the discrete Walsh transform, we will talk about discrete Hadamard transform and we will also see some properties of these different transformation techniques.

Now, for during the last 2 classes, when we have talked about the discrete Fourier transformation, you might have noticed one thing that this discrete Fourier transformation is nothing but a special case of a class of transformations or a class of separable transformations.

Some of these discussions, we have done while we have talked about the unitary transformation. Now, before we start our discussion on the discrete cosine transformation or Walsh transformation or Hadamard transform, let us have some more insight on this class of transformations. Now, as we said that discrete Fourier transformation is actually a special case of a class of transformations.

(Refer Slide Time: 2:53)

$$T(u,v) = \sum_{x,y=0}^{N-1} f(x,y) \cdot g(x,y,u,v)$$
$$f(x,y) = \sum_{u,v} T(u,v) \cdot h(x,y,u,v)$$
$$g(x,y,u,v) = g_1(x,u) \cdot g_2(y,v)$$
$$= g_1(x,u) \cdot g_r(y,v)$$

Let us see what is that class of transformation. You will find that if we define a transformation of this form say $T(u, v)$ is equal to double summation $f(x, y)$ where $f(x, y)$ is the 2 dimensional signal into $g(x, y, u, v)$ where both x and y vary from 0 to capital N minus 1.

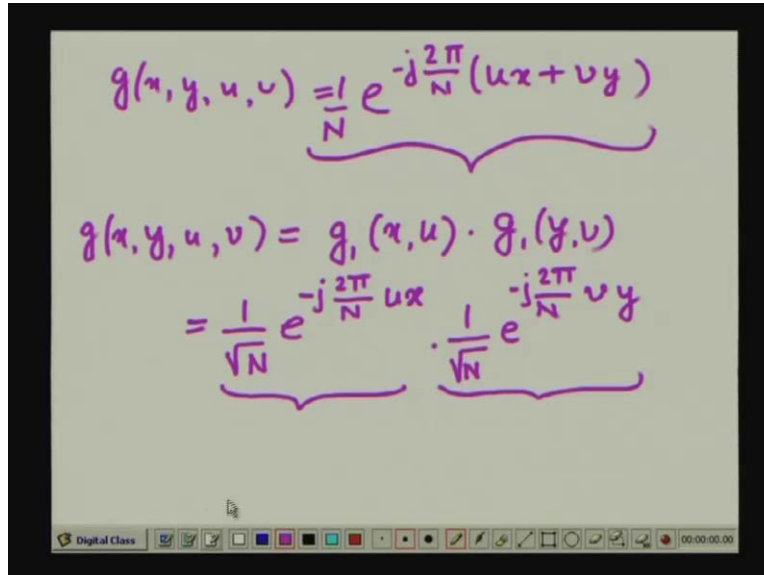
So, we are assuming that our 2 dimensional signal $f(x, y)$ is an N by N array, capital N by capital N array and the corresponding inverse transformation is given by $f(x, y)$ is equal to double summation again, we have this transformation matrix transform coefficients $T(u, v)$ into $h(x, y, u, v)$ where this $g(x, y, u, v)$ $g(x, y, u, v)$ this is called the forward transformation kernel and $h(x, y, u, v)$ is called the inverse transformation kernel or the basis functions.

Now, these transformations, this class of transformation will be separable if we can write $g(x, y, u, v)$ in the form $g_1(x, u)$ into $g_2(y, v)$. So, if $g(x, y, u, v)$ can be written in the form $g_1(x, u)$ into $g_2(y, v)$, then this transformation will be a separable transformation. Moreover, if g_1 and g_2 these are functionally same that means if I can write this as $g_1(x, u)$ into $g_1(y, v)$ that is I am assuming g_1 and g_2 to be functionally same.

So, in that case, these class of transformations will be separable obviously because $g(x, y, u, v)$ we have written as product of 2 functions - $g_1(x, u)$ into $g_2(y, v)$. And since $g_1(x, u)$ and $g_2(y, v)$; so this function g_1 and g_2 , they are functionally same, so this I can write as $g_1(x, u)$ into $g_1(y, v)$ and in this case, the function will be called as symmetric.

So here, what we have is this particular transformation or class of transformations is called separable as well as symmetric and the same is also true for the inverse transformation kernel that is $h(x, y, u, v)$.

(Refer Slide Time: 6:26)



The image shows a digital class screen with handwritten mathematical equations in purple ink. The equations are:

$$g(x, y, u, v) = \frac{1}{N} e^{-j \frac{2\pi}{N} (ux + vy)}$$
$$g(x, y, u, v) = g_1(x, u) \cdot g_1(y, v)$$
$$= \frac{1}{\sqrt{N}} e^{-j \frac{2\pi}{N} ux} \cdot \frac{1}{\sqrt{N}} e^{-j \frac{2\pi}{N} vy}$$

The screen also shows a toolbar at the bottom with various drawing tools and a timer indicating 00:00:00.

Now, find that for a 2 dimensional discrete Fourier transformation, we had $g(x, y, u, v)$ which was of this form e to the power minus $j 2 \pi$ by capital N into ux plus vy and of course, we had this multiplicity term 1 upon capital N . So, this was in the forward transformation kernel in case of 2 dimensional discrete Fourier transform or 2D DFT.

Obviously, this transformation is separable as well as symmetric because I can now write this $g(x, y, u, v)$ as $g_1(x, u)$ multiplied by $g_1(y, v)$ which is nothing but 1 over square root of capital N e to the power minus $j 2 \pi$ by capital N ux into 1 over square root of N e to the power minus $j 2 \pi$ by capital N vy .

So, you find that the first product $g_1(x, u)$ and the second term that is $g_1(y, v)$, they are functionally same but only the arguments; x in 1 case, it is ux and in the other case, it is vy . So obviously, this 2 dimensional discrete Fourier transformation is separable as well as symmetric. So, as we said that this **represents a specific case of** the 2 dimensional discrete Fourier transformations represents a specific case of a class of transformations and we had also discussed it, discussed the same when we have talked about the unitary transformation.

In today's lecture, we will talk about some other transformations belonging to the same class. The first transformation belonging to this class that we will talk about is called the discrete Fourier transformation or DFT. Let us see what are the forward as well as inverse transform kernels of this discrete Fourier transform.

(Refer Slide Time: 9:21)

$$\begin{aligned} & \text{DCT.} \\ g(x, y, u, v) &= \\ & \alpha(u) \cdot \alpha(v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cdot \\ & \cos\left[\frac{(2y+1)v\pi}{2N}\right] \\ & = h(x, y, u, v) \end{aligned}$$

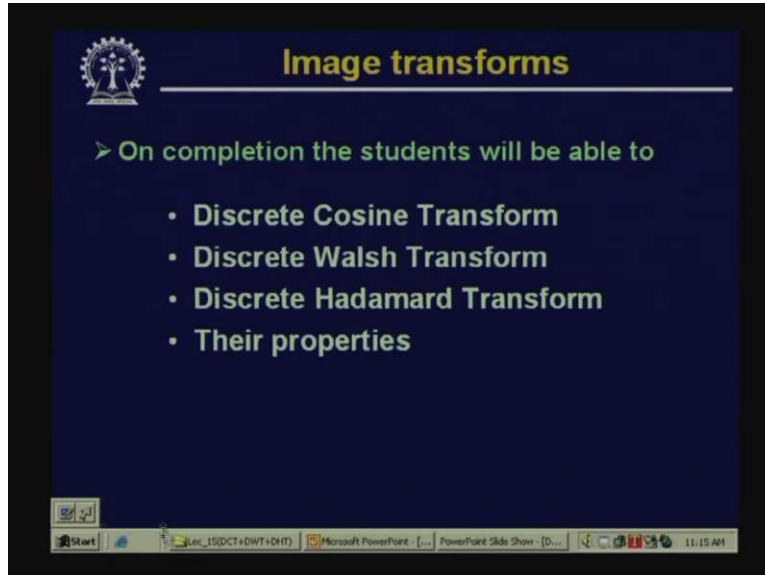
So now, let us talk about the discrete Fourier transform or DCT. In case of discrete Fourier transformation, the forward kernel forward transformation kernel $g(x, y, u, v)$ is given by alpha times u into alpha times v into cosine $2x$ plus 1 u pi upon $2N$ into cosine $2y$ plus 1 into u pi upon twice N which is same as the inverse transformation kernel which is given by $h(x, y, u, v)$.

So, you find that in case of discrete cosine transformation, if you analyze this, you find that both the forward transformation kernel and also the inverse transformation kernel, they are identical and not only that, these transformations transformation kernels are separable as well as symmetric because in this I can have $g_1(x, u)$ equal to alpha u cosine $2x$ plus 1 u pi divided by twice N and $g_1(y, v)$ can be alpha times v into cosine $2y$ plus 1 v pi upon twice N .

So, this transformation that is discrete cosine transformation is separable as well as symmetric and the inverse forward inverse transformation kernel and the forward transformation kernel, they are identical. Now, we have to see what the values of alpha u and alpha v. Here, alpha u is given by square root of 1 upon capital N where u is equal to 0 and it is equal to square root of twice by capital N for values of u equal to $1, 2$ to capital N minus 1 .

So, these are the values of alpha u for different values of u and similar is the values of alpha v for different values of v. Now, using these forward and inverse transformation kernels, let us see how the basis functions or the basis images look like in case of discrete cosine transform.

(Refer Slide Time: 12:40)



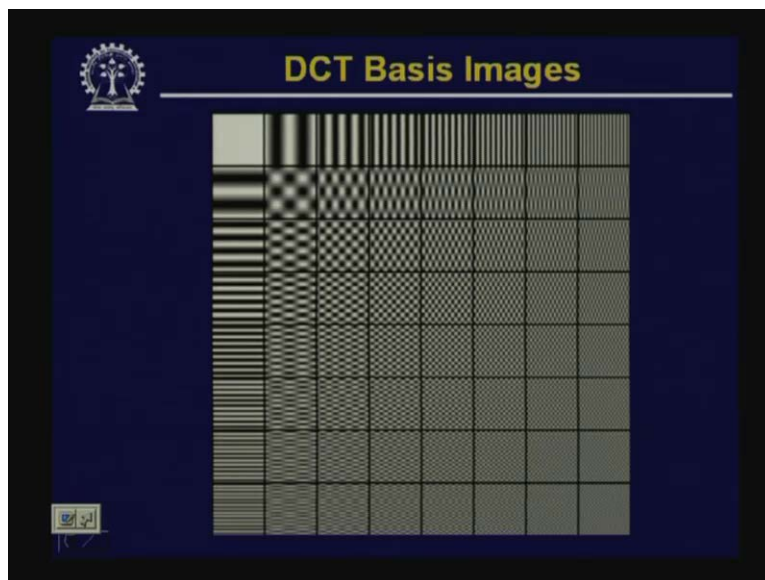
The slide features a dark blue background with a white logo in the top left corner. The title 'Image transforms' is centered at the top in a yellow font. Below the title, a green arrow points to the text 'On completion the students will be able to'. A bulleted list follows, listing four items: 'Discrete Cosine Transform', 'Discrete Walsh Transform', 'Discrete Hadamard Transform', and 'Their properties'. At the bottom, a Windows taskbar is visible with the Start button, a file explorer icon, and several open application windows including 'Lec_15(DCT+DWT+DHT)', 'Microsoft PowerPoint', and 'PowerPoint Slide Show'. The system clock shows '11:15 AM'.

Image transforms

➤ On completion the students will be able to

- Discrete Cosine Transform
- Discrete Walsh Transform
- Discrete Hadamard Transform
- Their properties

(Refer Slide Time: 12:42)



The slide has a dark blue background with a white logo in the top left corner. The title 'DCT Basis Images' is centered at the top in a yellow font. The main content is a grid of 8x8 small images, each representing a different basis function for the 2D Discrete Cosine Transform. The patterns in the grid vary from simple horizontal and vertical lines to complex, high-frequency oscillations. At the bottom left, there is a small icon and the number '17'.

DCT Basis Images

So, this figure shows the 2 dimensional basis images or basis functions in case of discrete cosine transformation where we have shown the basis images for an 8 by 8 discrete cosine transformation or 8 by 8 2 dimensional discrete cosine transformations.

(Refer Slide Time: 13:12)

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

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

Now, using these kernels, now we can write the expressions for the 2 dimensional discrete cosine transformation in the form of $c(u, v)$ is equal to $\alpha(u)\alpha(v)$ double summation $f(x, y)$ into cosine of $2x$ plus 1 into πu upon twice N into cosine of twice y plus 1 πv upon twice N where both x and y vary from 0 to capital N minus 1 .

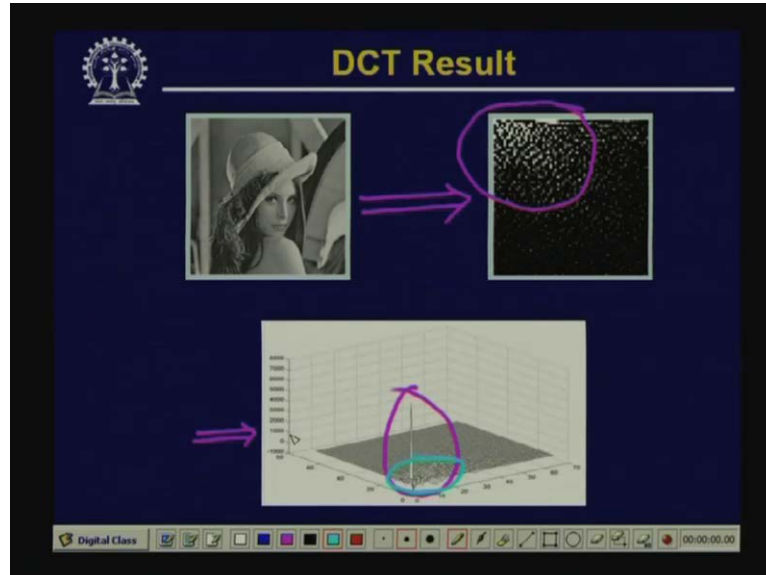
Similarly, the inverse discrete cosine transformation can be written as $f(x, y)$ is equal to double summation $\alpha(u)$ times $\alpha(v)$ times $c(u, v)$. So, $c(u, v)$ is the coefficient matrix into cosine of $(2x + 1)u\pi$ upon twice capital N into cosine of $(2y + 1)v\pi$ upon twice capital N and now u and v vary from 0 to capital N minus 1 . So, this is the forward 2 dimensional discrete cosine transformation and this is the inverse 2 dimensional discrete cosine transformation.

Now, you find that there is one difference in case of forward discrete cosine transformation. The terms $\alpha(u)$ and $\alpha(v)$ were kept outside the summation, double summation whereas in case of inverse discrete cosine transformation, the terms $\alpha(u)$ and $\alpha(v)$ are kept inside the double summation.

The reason being, in case of forward transformation because the summation is taken over x and y varying from 0 to capital N minus 1 , so $\alpha(u)$ and $\alpha(v)$, these terms are independent of this summation operation whereas, in case of inverse discrete cosine transformation, the double summation is taken over u and v varying from 0 to capital N minus 1 . So, these terms $\alpha(u)$ and $\alpha(v)$ are kept inside the double summation operation.

So, using this discrete cosine transformation 2 dimensional discrete cosine transformation, let us see that for a given image, what kind of output we get.

(Refer Slide Time: 17:00)



So this shows, this figure shows the **discrete Fourier** discrete cosine transformation coefficients for the same image which is very popular in image processing community, the image of length. The results are shown in 2 forms. The first figure, this is the coefficients which is shown in the form of intensity plots in the form of a 2 dimensional array whereas, the third figure shows the same coefficients which are plotted **in the form of a 3 dimensional** in the form of a surface in 3 dimension.

Now, if you closely look at this output coefficients, you find that in case of discrete cosine transformation, the energy of the coefficients are concentrated mostly in a particular region where the coefficients are near the origin that is $(0, 0)$ which is more visible in the case of a 3 dimensional **pot**. So, you find that here in this particular case, the energy is concentrated in a small region in the coefficients space near about the $(0, 0)$ coefficients. So, this is a very very important property of the discrete cosine transformation which is called energy compaction property.

Now, among the other properties of discrete cosine transformation which is obviously similar to the discrete Fourier transformation; as we have said that the discrete cosine transformation is separable as well as symmetric, it is also possible to have a faster implementation of discrete cosine transformation or FDCT in the same manner as we have implemented FFT incase of discrete Fourier transformation.

The other important property of the discrete cosine transformation is the periodicity property. Now, in case of discrete cosine transformation, you will find that the periodicity is not same as incase of discrete Fourier transformation. In case of Fourier transformation, we have said that the discrete Fourier transform is periodic with period capital N where N is the number of samples. In case of discrete cosine transformation, the magnitude of the coefficients are periodic with a period twice N where N is the number of samples.

So, the periodicity in case of discrete cosine transformation is twice or the period in case of discrete cosine transformation is twice of the period in case of discrete Fourier transformation and we will see later that this particular property helps to obtain data compression, the smother data compression using the discrete cosine transformation and not using the discrete Fourier transformation.

The other property which obviously helps the data compression using discrete cosine transformation is the energy compaction property because most of the signal energy or image energy is concentrated in a very few number of coefficients near the origin or near the (0, 0) value in the frequency domain in the uv plane.

So, by coding few numbers of coefficients, we can represent or we can represent most of the energy, most of the signal energy or most of the image energy. So, that also helps in the data compression using discrete cosine transformation, a property which is not normally found in case of discrete Fourier transformation.

So, after discussing about all these different properties of the discrete cosine transformation, let us go to the other transformation which we have said as Walsh transformation.

(Refer Slide Time: 21:05)

Walsh Transform

$$\text{1-D } g(x, u) = \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{b_i(x) b_{n-1-i}(u)}$$

$N \rightarrow$ no. of samples
 $n \rightarrow$ no. of bits x/u .
 $b_k(z) \rightarrow k^{\text{th}}$ bit in digital/binary representation of z

So now, let us discuss about the Walsh transform. In case of 1 D, the discrete Walsh transform kernels are given by $g(x, u)$ is equal to 1 upon capital N into product minus 1 to the power $b_i(x)$ into $b_{n \text{ minus } 1 \text{ minus } i}(u)$ where the product is taken over i equal to 0 to $n \text{ minus } 1$. So, you find in this particular case, the capital N gives you the number of samples and the lower case n is the number of bits needed to represent x as well as u sorry this is u not x .

So, capital N is the number of samples and the lower case n is the number of bits needed to represent both x and u and in this case, the forward transformation kernel is given by $g(x, u)$

equal to 1 upon capital N into product i equal to 0 to lower case n minus 1, minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$.

Now, in this particular case, the convention is; see, if I represent $b_k(Z)$, $b_k(Z)$ represents the k'th bit in the digital representation of z, digital or binary representation of z. So, that is the interpretation of $b_i(x)$.

(Refer Slide Time: 24:00)

$$w(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) \cdot \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}$$

[Inv. kernel]

$$h(x, u) = \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}$$

$$f(x) = \sum_{u=0}^{N-1} w(u) \prod_{i=0}^{n-1} (-1)^{b_i(x)b_{n-1-i}(u)}$$

So, using this, the forward discrete Walsh transformation will be given by $w(u)$, in case of 1 dimension will be given by 1 upon capital N summation $f(x)$ into product i equal to 0 to lower case n minus 1, minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$ where x varies from 0 to capital n minus 1.

The inverse transformation kernel, in case of this discrete Walsh transformation is identical with the forward transformation kernel. So, $h(x, u)$, the inverse transformation kernel is same as product i equal to 0 to lower case n minus 1 into minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$ and using this inverse transformation kernel, we can get the inverse Walsh transformation as $f(x)$ equal to summation u equal to 0 to capital N minus 1 $W(u)$ product i equal to 0 to lower case n minus 1, minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$ So, this is the inverse kernel and this is the inverse transformation.

So here, you find that the discrete Walsh transformation, both the forward transformation and the inverse transformation, they are identical. Only thing **is there** is the difference of the multiplicity factor 1 upon capital N but otherwise because the transformations are identical, so the algorithm used to perform the forward transformation, the same algorithm can also be used to perform the inverse Walsh transformation.

(Refer Slide Time: 26:35)

2-D

$$g(x, y, u, v) = \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{\{b_i(x) \cdot b_{n-1-i}(u) + b_i(y) \cdot b_{n-1-i}(v)\}}$$

$$f(x, y, u, v) = \frac{1}{N} \prod_{i=0}^{n-1} (-1)^{\{b_i(x) \cdot b_{n-1-i}(u) + b_i(y) \cdot b_{n-1-i}(v)\}}$$

Now, in case of 2 dimensional signal; so in case of 2 dimensional signal, we will have the transformation kernel as $g(x, y, u, v)$ which is equal to 1 upon capital N product i equal to 0 to lower case n minus 1, minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$ plus $b_i(y)$ into $b_{n-1-i}(v)$ and the inverse transformation kernel in this case is identical with the forward transformation kernel.

So, the inverse transformation kernel is given by 1 upon capital N product again i equal to 0 to lower case n minus 1, minus 1 to the power $b_i(x) \cdot b_{n-1-i}(u)$ plus $b_i(y) \cdot b_{n-1-i}(v)$.

(Refer Slide Time: 28:35)

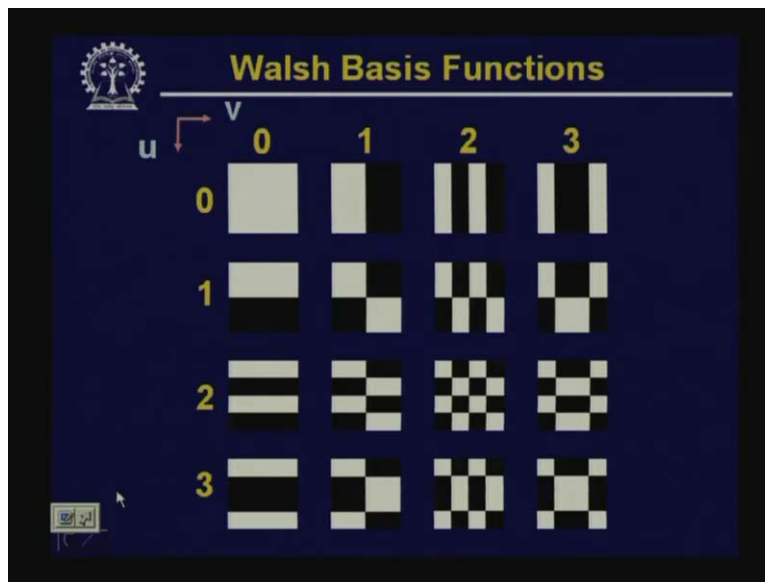
$$W(u, v) = \frac{1}{N} \sum_{x, y=0}^{N-1} f(x, y) \prod_{i=0}^{n-1} (-1)^{\{b_i(x) \cdot b_{n-1-i}(u) + b_i(y) \cdot b_{n-1-i}(v)\}}$$

So, using this forward transformation kernel and the inverse transformation kernel, now you find that the inverse as well as the forward discrete Walsh transformation can now be implemented as $W(u, v)$ is equal to 1 upon capital N double summation $f(x, y)$ into product i equal to 0 to n minus 1, minus 1 to the power $b_i(x)$ into $b_{n \text{ minus } 1 \text{ minus } i}(u)$ plus $b_i(y)$ into $b_{n \text{ minus } 1 \text{ minus } i}(v)$ and the summation has to be taken over x and y varying from 0 to capital N minus 1.

And in the same manner, because the forward transformation as well as the inverse transformation, they are identical in case of discrete Walsh transformation; the same expression if I replace $f(x, y)$ by $W(u, v)$ and the summation is taken over u, v varying from 0 to capital N minus 1, what I get is the inverse Walsh transformation and I get back the original signal $f(x, y)$ from the transformation coefficients $W(u, v)$.

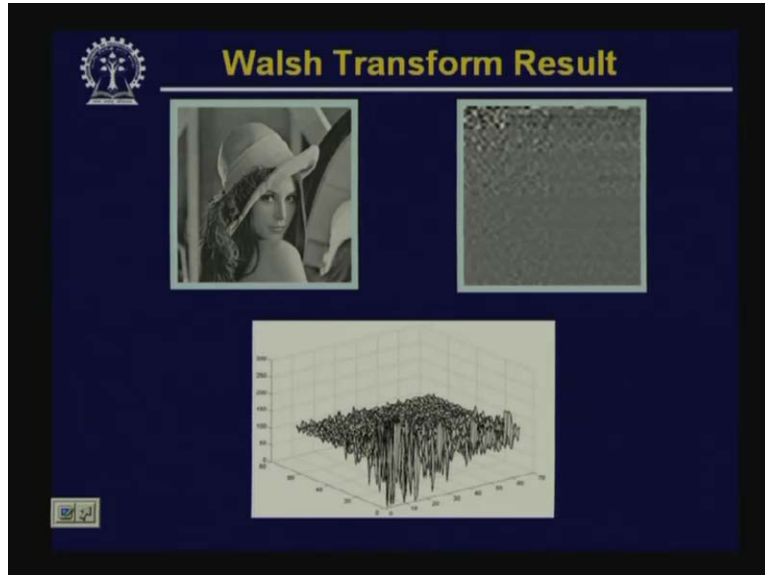
So, you find that here, the same algorithm which is used for computing the forward Walsh transformation can also be used for computing the inverse Walsh transformation. So now, let us see that what are the basis functions of this Walsh transformation and what are the results on some image.

(Refer Slide Time: 30:34)



So, for Walsh transformation, the basis function appears like this or the state of basis images appear like this. Here, the basis images are given for a 4 by 4 2D Walsh transformation and if I apply this Walsh transformation on the same image say Lena, you find that this is the kind of result that we get.

(Refer Slide Time: 30:51)

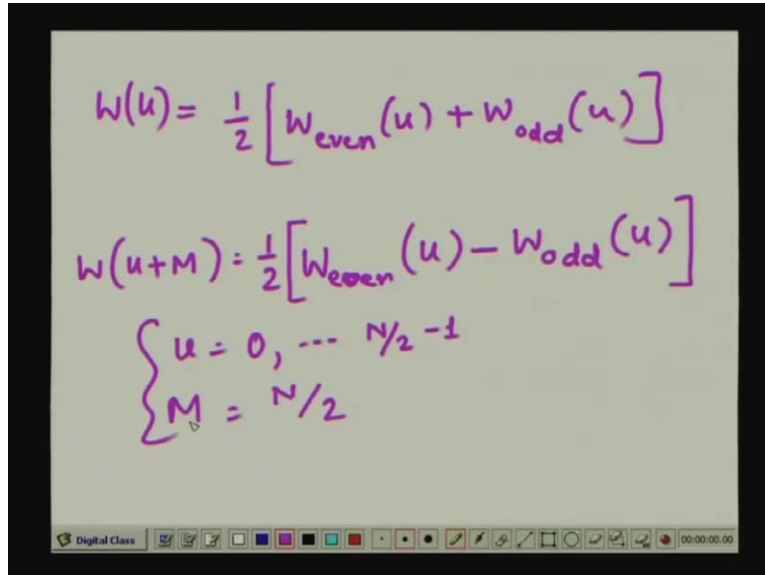


So, here again, you find that the property of this cosine transformations that is the coefficients neared 0, they are having the maximum energy and as you go away from the origin in the uv plane the energy of the coefficients reduces. So, this transformation also has the energy compaction property. But here you find that the energy compaction property is not as strong as in case of the discrete cosine transformation.

So here, the coefficient energies which is mostly concentrated in this particular region is not that strong as the compaction of energy in case of discrete cosine transformation and by analyzing this forward as well as the inverse transformation, **Walsh transformation kernels** Walsh transformation kernels; you can again find out thus that this Walsh transformation is separable as well as symmetric.

Not only that, for this Walsh transformation, it is also possible to have a fast implementation of 2D Walsh transformation almost in the same manner as we have done in case of the discrete Fourier transformation where we have computed the first Fourier transform or FFT.

(Refer Slide Time: 32:29)



The image shows a digital class screen with handwritten mathematical equations in purple ink. The equations are:

$$W(u) = \frac{1}{2} [W_{\text{even}}(u) + W_{\text{odd}}(u)]$$
$$W(u+M) = \frac{1}{2} [W_{\text{even}}(u) - W_{\text{odd}}(u)]$$
$$\begin{cases} u = 0, \dots, N/2 - 1 \\ M = N/2 \end{cases}$$

At the bottom of the screen, there is a toolbar with various icons and a timer showing 00:00:00.

So, in case of discrete Walsh transformation, the first implementation of the Walsh transformation will be even simpler and in this case, the first transformation can be implemented as $W(u)$. So here, we are saying that because the Walsh transformation is separable; so the same way in which we have done the Fourier transformation, 2D Fourier transformation that the Walsh transformation, 2D Walsh transformation can be implemented by using a sequence of 1 dimensional Walsh transformation and that is also true in case of discrete cosine transformation.

So, first you perform 1 dimensional Walsh transformation along the rows of the image and then the intermediate result that you get, on that you perform 1 dimensional Walsh transformation along the columns of the intermediate matrix. So, you get the final transformation coefficients. The same is also true in case of discrete cosine transformation because the discrete cosine transformation is also separable.

So, to illustrate the first implementation of the Walsh transformation, I take the 1 dimensional case. So here, the first implementation can be done in this form. I can write $W(u)$ is equal to half of $W_{\text{even}}(u)$ plus $W_{\text{odd}}(u)$ and $W(u)$ plus capital M is equal to half of $W_{\text{even}}(u)$ minus $W_{\text{odd}}(u)$. So, you find and in this case, u varies from 0 to capital N by 2 minus 1 and M is equal to N by 2.

So, we find that almost in the same manner in which we have implemented the first Fourier transformation, the discrete 2 dimensional or discrete Walsh transformations, first discrete Walsh transformation can also be implemented in the same manner. Here, we divide all the samples for which the Walsh transformation has to be taken into even numbered samples and odd numbered samples, compute the Walsh transformation of the even numbered samples, compute the Walsh transform of the odd numbered samples, then combine these 2 intermediate results to give you the Walsh transformation of the total number of samples.

And because this division can be recursive, so first I have N number of samples. I divide them into N by 2 odd samples and N by 2 even samples, even N by 2 odd samples can be divided into N by 4 number of odd samples and even samples and if I continue this and finally I come to a stage where I am left with only 2 samples, I perform the Walsh transformation of those 2 samples, then hierarchical combine those intermediate results to get the final Walsh transformation.

So here again, by using this fast implementation of the Walsh transformation, you may find that the computational complexity will be reduced drastically.

(Refer Slide Time: 35:53)

The image shows a whiteboard with the following handwritten text:

Hadamard Transform.

$$g(x, u) = \frac{1}{N} \sum_{i=0}^{n-1} (-1)^{i \cdot b_i(x) \cdot b_i(u)}$$

$$H(u) = \frac{1}{N} \sum_{x=0}^{N-1} f(x) (-1)^{\sum_{i=0}^{n-1} b_i(x) \cdot b_i(u)}$$

The whiteboard also features a toolbar at the bottom with various drawing tools and a timer showing 00:00:00.00.

So, after discussing about the Walsh transformation, let us go to the next transformation which is called the Hadamard transform. So, the next transformation that we discuss is Hadamard transform. In case of Hadamard transform, first let us say consider the case in 1 dimension. The forward transformation kernel is given by $g(x, u)$ equal to $\frac{1}{N}$ upon capital N minus 1 to the power summation $b_i(x)$ into $b_i(u)$ where this i varies from 0 to lower case n minus 1.

So again, the capital N as well as lower case N, they have the same interpretation as in case of Walsh transformation and using this forward transformation kernel, **the forward Walsh transformation** forward Hadamard transformation can be obtained as $H(u)$ equal to $\frac{1}{N}$ upon capital N summation x varies from 0 to capital N minus 1 $f(x)$ into minus 1 to the power summation $b_i(x)$ into $b_i(u)$ where i varies from 0 to lower case n minus 1.

And for Hadamard transform also, the forward transformation as well as the inverse transformation, they are identical. That is the forward transformation kernel and the inverse transformation kernel, they are identical. So here again, the same algorithm can be used for forward transformation as well as the inverse transformation.

(Refer Slide Time: 37:53)

The image shows a digital whiteboard with two equations written in purple ink. The first equation is
$$h(x, u) = (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)}$$
 and the second equation is
$$\Rightarrow f(x) = \sum_{u=0}^{N-1} H(u) (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u)}$$
. At the bottom of the whiteboard, there is a toolbar with various drawing tools and a timer showing 00:00:00.00.

So here, the inverse transformation kernel is given by $h(x, u)$ is equal to minus 1 to the power summation $b_i(x)$ into $b_i(u)$ where i varies from 0 to lower case n minus 1 and using this, the inverse Hadamard transformation is obtained as $f(x)$ is equal to summation u varying from 0 to capital N minus 1 $H(u)$ minus 1 to the power summation $b_i(x)$ into $b_i(u)$ where i varies from 0 to lower case n minus 1. So, these are the forward and inverse Hadamard transformation in case of 1 dimension.

(Refer Slide Time: 39:03)

The image shows a digital whiteboard with two equations written in purple ink. The first equation is
$$g(x, y, u, v) = \frac{1}{N} (-1)^{\sum_{i=0}^{n-1} b_i(x) b_i(u) + b_i(y) b_i(v)}$$
 and the second equation is
$$h(x, y, u, v) = \frac{1}{N} (-1)^{\sum_{i=0}^{n-1} [b_i(x) \cdot b_i(u) + b_i(y) b_i(v)]}$$
. At the bottom of the whiteboard, there is a toolbar with various drawing tools and a timer showing 00:00:00.00.

Obviously, this can easily be extended into 2 dimension as in the other cases where the 2 dimensional forward and inverse transformation will be given by $g(x, y, u, v)$ is equal to 1 upon

capital N into minus 1 to the power summation $b_i(x)$ into $b_i(u)$ plus $b_i(y)$ into $b_i(v)$ where this summation is taken from over i equal to 0 to lowercase n minus 1 and similarly, the inverse transformation kernel is also given by $h(x, y, u, v)$ which is same as $g(x, y, u, v)$ that is 1 upon capital N minus 1 to the power summation i equal to 0 to lower case n minus 1 into $b_i(x)$ into $b_i(u)$ plus $b_i(y)$ into $b_i(v)$.

So, you find that the forward transformation kernel and the inverse transformation kernel in case of 2 dimensional discrete Hadamard transformations are identical. So, that gives us the forward transformation and the inverse transformation for the 2 dimensional discrete Hadamard transformations to be same which enables us to use the same algorithm or same program to compute the forward transformation as well as the inverse transformation.

And, if you analyze this, you find that this Hadamard transformation is also separable and symmetric. That means in the same manner, this 2 dimensional Hadamard transformation can be implemented by using a sequence of 1 dimensional Hadamard transformations. So, for the image first we implement 1 dimensional Hadamard transformation over the rows of the image and then implement the 1 dimensional Hadamard transformations over the columns of this intermediate matrix and that gives you the final Hadamard transformation output.

Now, if I further analyze the kernels of this Hadamard transformation and because we have said that 2 dimensional Hadamard transformations can now be implemented in the form of a sequence of 1 dimensional Hadamard transformations, so we analyze further with respect to an 1 dimensional Hadamard transformation.

(Refer Slide Time: 42:09)

The image shows a handwritten equation on a digital whiteboard. The equation is:
$$g(x, u) = \left(\frac{1}{N}\right) \sum_{i=0}^{n-1} b_i(x) b_i(u)$$
The term $\left(\frac{1}{N}\right)$ is circled in blue. The whiteboard interface includes a toolbar at the bottom with various drawing tools and a timer showing 00:00:00:00.

So, **as you** as you have seen that 1 dimensional Hadamard transformation is given by $g(x, u)$ is equal to 1 upon capital N minus 1 to the power summation $b_i(x)$ into $b_i(u)$ where i varies from 0 to lower case n minus 1 and let me mention here that all these additions that we are doing, this

summations follow modular 2 arithmetic that means this summations are actually nothing but ORing operation of different bits.

Now, if I analyze this 1 dimensional forward Hadamard transformation kernel, you will find that if I omit the multiplicative term 1 upon N, so if I omit from this, this multiplicative term 1 upon capital N; then this forward transformation actually forms, leads to a matrix which is known as Hadamard matrix.

(Refer Slide Time: 43:22)

Hadamard Matrix (N=8)

		$u \longrightarrow$							
		0	1	2	3	4	5	6	7
$x \downarrow$	0	+	+	+	+	+	+	+	+
	1	+	-	+	-	+	-	+	-
	2	+	+	-	-	+	+	-	-
	3	+	-	-	+	+	-	-	+
	4	+	+	+	+	-	-	-	-
	5	+	-	+	-	-	+	-	+
	6	+	+	-	-	-	-	+	+
	7	+	-	-	+	-	+	+	-

So, to see that what is this Hadamard matrix, you will find that for different values of x and u, the Hadamard matrix will look like this. So, here we have shown a Hadamard matrix for Hadamard transformation of dimension 8. So, for N equal to 8, this Hadamard matrix has been formed and here plus means it is equal to plus 1 and minus means it is equal to minus 1.

Now, if you analyze this particular Hadamard matrix, you will find that it is possible to generate a recursive relation; it is possible to formulate a recursive relation to generate the transformation matrixes. Now, how that can be done?

You will find that this particular path, if I consider say these 4 by 4 elements, these 4 by 4 elements, these 4 by 4 elements, they are identical. Whereas, these 4 by 4 elements of this matrix is just negative of this and the same is followed, the same pattern can be observed in all other parts of this matrix. So, by observing this, now we can formulate a recursive relation to generate these transformation matrixes.

(Refer Slide Time: 45:02)

The image shows a digital whiteboard with two equations written in purple ink. The first equation is $H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ with $N=2$ written above it. The second equation is $H_{2N} = \begin{bmatrix} H_N & H_N \\ H_N & H_{-N} \end{bmatrix}$. At the bottom of the whiteboard, there is a toolbar with various drawing tools and a timer showing 00:00:00.

So, to have that recursive relation, let us first have a Hadamard matrix of the lowest order that is for value of n equal to 2 and for this lowest order, we have a Hadamard matrix H_2 which is nothing but (1, 1, 1, minus 1) and then using this recursively, a Hadamard matrix of dimensional $2N$ can be obtained from a Hadamard matrix of dimensional N which is given by the relation (H_N , H_N , H_N and H minus N). So, a Hadamard matrix of higher dimension can recursively formed from a Hadamard matrix of lower dimension. So, this is a very very important property of the Hadamard transformation.

Now, if we analyze the Hadamard matrix further; let us see, suppose I want to analyze, I want to analyze this particular Hadamard matrix, here you find that if I consider the number of sign changes along a particular column, so you find that the number of sign changes along column number 0 is equal to 0. Number of sign changes along column number 1 is equal to 7, number of sign changes along column number 2 is equal to 3, along column number 3, the number of sign changes equal to 4, along column 4 it is equal to 1, along column 5 it is equal to 6, along column 6 it is equal 2, along column 7 it is equal to 5.

So, if I define the number of sign changes along a particular column as the sequence which is similar to the concept of frequency in case of discrete Fourier transformation or in case of discrete cosine transformation; so in case of Hadamard matrix, we are defining the number of sign changes along a particular column or for a particular value of u as the sequence.

So here, you will find that for value of u is equal to 0, the sequence is equal to 0; u equal to 1, the sequence equal to 7; u equal to 2, the sequence equal to 3. So, there is no straight forward relation between the value of u and the corresponding sequence unlike in case of discrete Fourier transform or in case of discrete cosine transform or we have seen that increasing values of the frequency variable u corresponds to increasing values of the frequency components.

(Refer Slide Time: 48:27)

$$g(x, u) = \frac{1}{N} (-1)^{\sum_{i=0}^{n-1} b_i(x)} \underbrace{p_i(u)}$$
$$\left. \begin{aligned} p_0(u) &= b_{n-1}(u) \\ p_1(u) &= b_{n-1}(u) + b_{n-2}(u) \\ p_2(u) &= b_{n-2}(u) + b_{n-3}(u) \\ &\vdots \\ p_{n-1}(u) &= b_1(u) + b_0(u) \end{aligned} \right\}$$

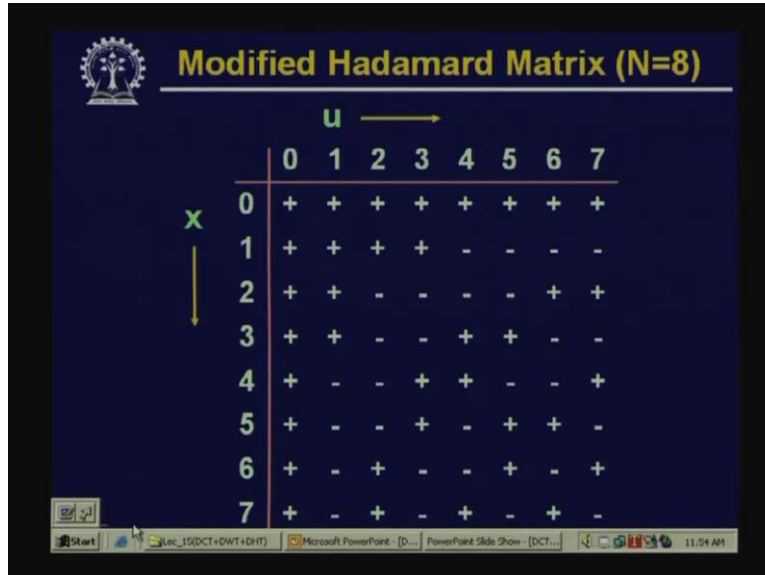
So, if we want to have similar type of concepts in case of Hadamard transformation also, then what we need is we need some sort of reordering of this hadamard matrix and that kind of reordering can be obtained by another transformation where that particular transformation, the kernels of that particular transformation will be given by $g(x, u)$ is equal to 1 upon capital N minus 1 to the power summation $b_i(x)$ into $p_i(u)$. Now, instead of $b_i(u)$, we are writing $p_i(u)$ where this summation is taken over i equal to 0 to lower case n minus 1 and this particular term $p_i(u)$ can be obtained from $b_i(u)$ using these relations.

$p_0(u)$ will be given by $b_{n \text{ minus } 1}(u)$, $p_1(u)$ will be given by $b_{n \text{ minus } 1}(u)$ plus $b_{n \text{ minus } 2}(u)$, $p_2(u)$ will be given by $b_{n \text{ minus } 2}(u)$ plus $b_{n \text{ minus } 3}(u)$ and continuing like this, $p_{n \text{ minus } 1}(u)$ will be given by $b_1(u)$ plus $b_0(u)$ where all these summations are again modulo 2 summations. That is they can also be explained, they can be implemented using the binary OR operations.

Now, by using this modification, again this particular forward transformation kernel that you get, the modified forward transformation kernel that you get that leads to a modified Hadamard matrix. So, let see what is this modified Hadamard matrix.

So, the modified Hadamard matrix that you get is of this particular form and if you look at to this particular modified Hadamard matrix, you find that here sequence for u equal to 0 is again equal to 0, sequence for u equal to 1 is equal 1, sequence for u equal to 2 is equal 2 and now for increasing values of u , we have increasing values of sequencing.

(Refer Slide Time: 50:30)

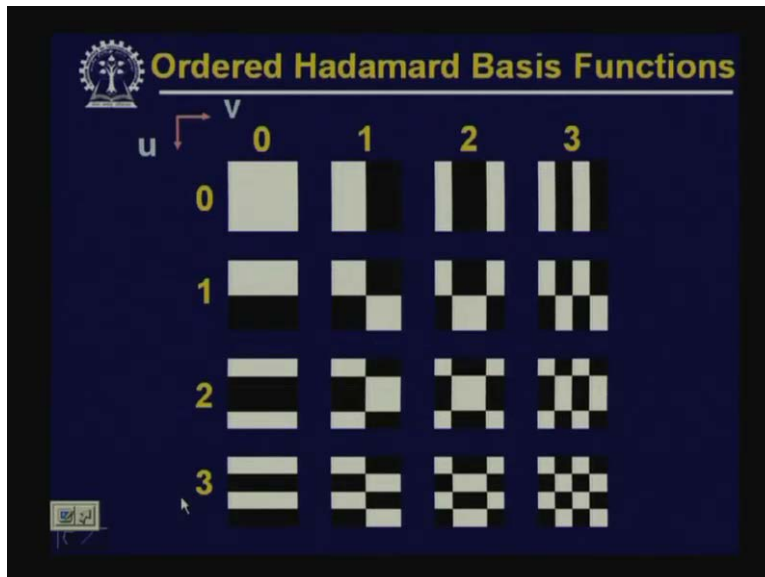


The slide displays an 8x8 matrix with a title "Modified Hadamard Matrix (N=8)". The columns are labeled 'u' and indexed 0 to 7. The rows are labeled 'x' and indexed 0 to 7. The matrix contains '+' and '-' signs in a specific pattern.

	u	0	1	2	3	4	5	6	7
x	0	+	+	+	+	+	+	+	+
1	+	+	+	+	-	-	-	-	-
2	+	+	-	-	-	-	+	+	+
3	+	+	-	-	+	+	-	-	-
4	+	-	-	+	+	-	-	+	+
5	+	-	-	+	-	+	+	-	-
6	+	-	+	-	-	+	-	+	+
7	+	-	+	-	+	-	+	-	-

So, using this modified or ordered hadamard matrix forward kernel; in case of 2 dimensions, the ordered Hadamard basis functions are obtained in this particular form.

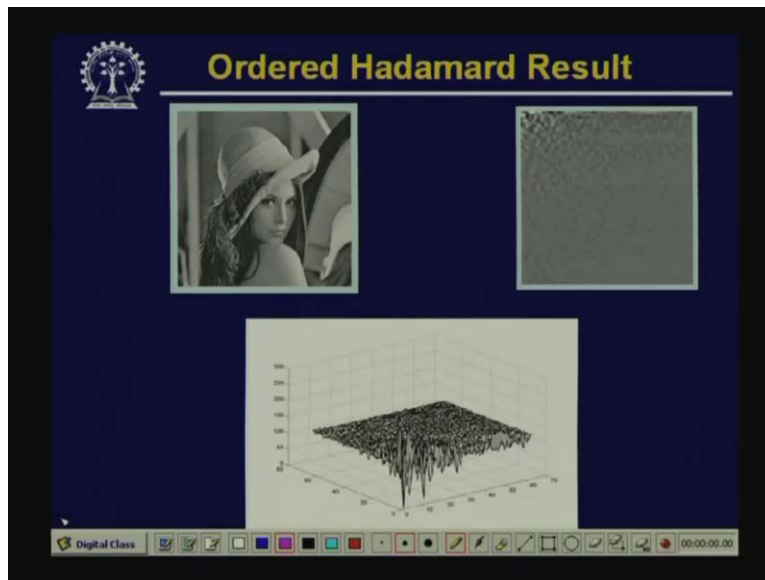
(Refer Slide Time: 51:14)



And using this ordered Hadamard basis functions, if I compare this with the Walsh basis functions, you will find that basis functions or basis images in case of Walsh transformation and the basis images in case of ordered Hadamard transformation; the basis functions are identical. But there is a difference of ordering of the Walsh basis functions and the ordered Hadamard basis functions.

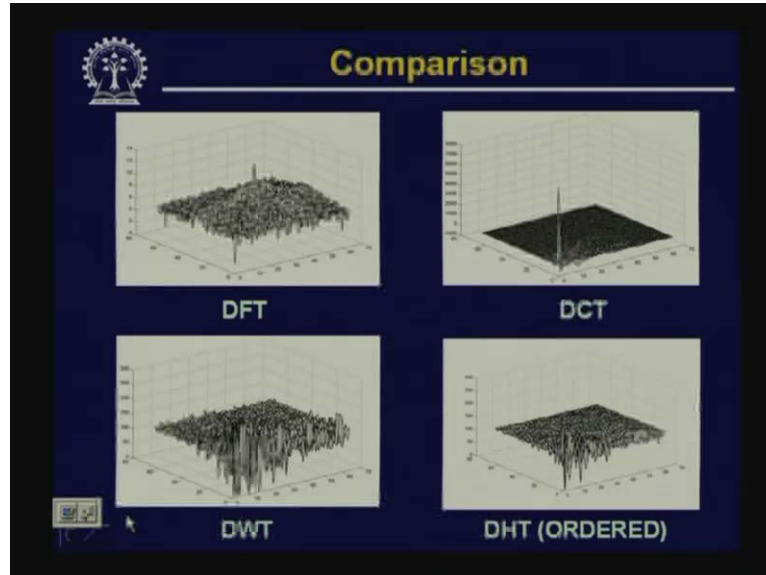
Otherwise, the basis functions for Walsh transformation and the ordered Hadamard transformation, they are identical and because of this, in many cases, a term which is used is called Walsh Hadamard transformation and this term Walsh Hadamard transformation is actually used to mean either Walsh transformation or Hadamard transformation. So, this uses, this means 1 of the 2. Now, using this ordered Hadamard transformation, the result on the same image that you get is something like this.

(Refer Slide Time: 52:17)



So here you find, again, if I look at the energy distribution of different Hadamard coefficients, the ordered Hadamard coefficients; you will find that here the energy is concentrated more towards 0 compared to the Walsh transformation. So, the energy compaction property of the ordered Hadamard transformation is more than the energy compaction property of Walsh transformation.

(Refer Slide Time: 53:07)



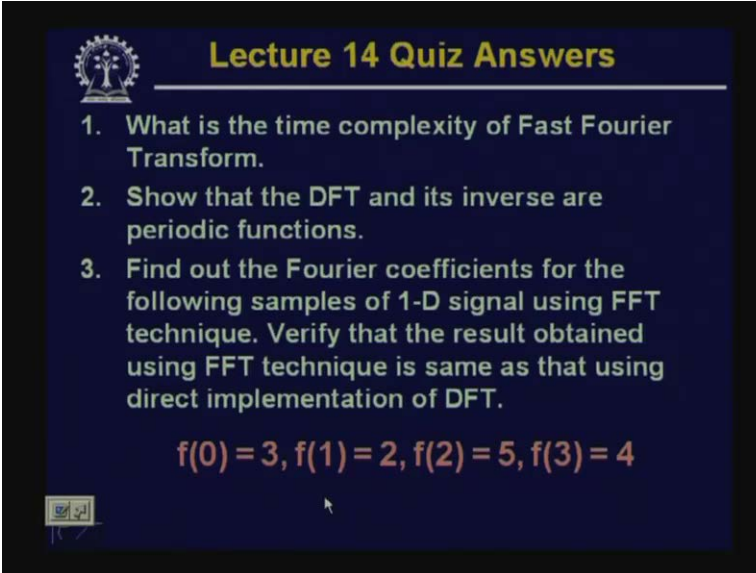
Now, this slide shows the comparison of the transformation coefficients of this different transformation. The first one shows the coefficient matrix for the discrete Fourier transformation, the second one shows the coefficient matrix for the discrete cosine transformation, third one is for discrete Walsh transformation and fourth one is for discrete Hadamard transformation, it is ordered Hadamard transformation.

Now, by comparing all these 4 different results; you find that in case of discrete cosine transformation, the discrete cosine transformation has the property of strongly concentrating the energy in very few numbers of coefficients. So, the energy compaction property of discrete cosine transformation is much more compared to the other transformations and that is why this discrete cosine transformation is very popular for the data compression operations unlike the other cases.

And, in case of discrete Fourier transformation and discrete cosine transformation, though we can associate the frequency term with the transformation coefficients, it is not possible to have such a physical interpretation of the coefficients of the discrete Walsh transformation nor in case of discrete Hadamard transformation.

So, though we cannot have such a kind of physical interpretation but still because of this energy compaction property, the Hadamard transform as well as the Walsh transform can have some application in data compression operations. So, with this we come to our end of our discussion on the discrete cosine transformation, discrete Walsh transformation and discrete Hadamard transformation and we have also seen some comparison with the discrete Fourier transformation.

(Refer Slide Time: 55:06)



The slide is titled "Lecture 14 Quiz Answers" and features a logo of a tree in a circle on the left. It contains three numbered questions:

1. What is the time complexity of Fast Fourier Transform.
2. Show that the DFT and its inverse are periodic functions.
3. Find out the Fourier coefficients for the following samples of 1-D signal using FFT technique. Verify that the result obtained using FFT technique is same as that using direct implementation of DFT.

Below the questions, the signal samples are defined as: $f(0) = 3, f(1) = 2, f(2) = 5, f(3) = 4$

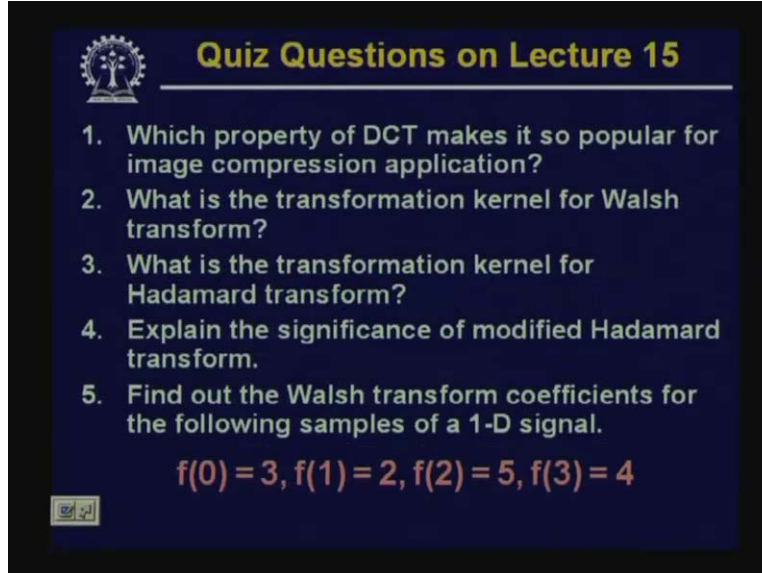
Now, let us discuss about the questions that we have discussed, we have given in the last class.

The first 1 is the time complexity of the first Fourier transformation and if you analyze the way in which the first Fourier transform is implemented, you can easily find out that the complexity, the computational complexity of first Fourier transformation will be in $n \log n$ where n is the data size.

The second question show that DFT and its inverse are periodic functions where the periodicity is N capital N . So, in the expression if you replace u by u plus capital N or v by v plus capital N , you will find that we will get back the same expression which means that the DFT as well as the inverse DFT, they are periodic with a periodicity of capital N .

The third question is find out the **Fourier coefficients for** discrete Fourier coefficients for the sequence of samples as given here. We have to compute the first Fourier transformation and verify that the discrete Fourier transform coefficients as well as FFT coefficients, they are identical. It is also quite straight forward by using the expressions that we have discussed when we have talked about the Fourier transformation and discrete Fourier transformation, these coefficients can be easily computed and it can be verified that DFT coefficients and FFT coefficients, they are identical.

(Refer Slide Time: 56:37)



The image shows a slide titled "Quiz Questions on Lecture 15" with a list of five questions and a signal definition. The slide has a dark blue background with white and yellow text. A small logo is in the top left corner. The questions are numbered 1 to 5. The signal definition is written in red text.

Quiz Questions on Lecture 15

1. Which property of DCT makes it so popular for image compression application?
2. What is the transformation kernel for Walsh transform?
3. What is the transformation kernel for Hadamard transform?
4. Explain the significance of modified Hadamard transform.
5. Find out the Walsh transform coefficients for the following samples of a 1-D signal.

$f(0) = 3, f(1) = 2, f(2) = 5, f(3) = 4$

Now, coming to today's quiz questions; the first question is which property of discrete cosine transform makes it so popular for image compression applications. The second one, what is the transformation kernel for Walsh transformation. Third question - what is the transformation kernel for Hadamard transform.

Fourth question - explain the significance of modified Hadamard transform. Now, in this case, by modified Hadamard transform I mean the ordered Hadamard transform. So, explain the significance of ordered Hadamard transform. The fifth question - find out the Walsh transform coefficients for the following samples of a 1 dimensional signal where sample values are $f(0)$ equal to 3, $f(1)$ equal to 2, $f(2)$ equal to 5 and $f(3)$ equal to 4.

Thank you.