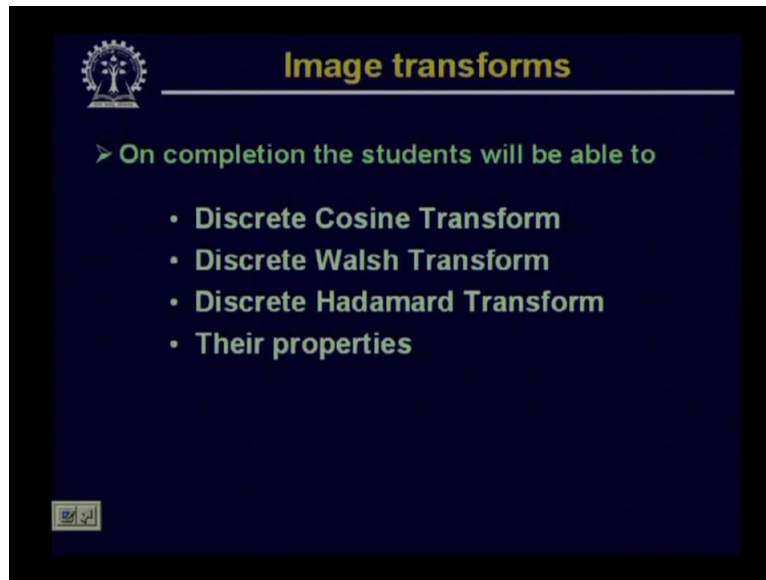


Digital Image Processing.
Professor P. K. Biswas.
Department of Electronics and Electrical Communication Engineering.
Indian Institute of Technology, Kharagpur.
Lecture-28.
DCT and Walsh Transform.

(Refer Slide Time: 0:27)



Hello. Welcome to the video lecture series on Digital Image Processing. We will talk about the Discrete Cosine Transform, we will talk about the Discrete Walsh Transform, we will talk about the Discrete Hadamard Transform and we will also see some properties of these different transformation techniques.

Now during the last two classes, when we have talked about Discrete Fourier Transformation, you might have noticed one thing, that this Discrete Fourier Transformation is nothing but a special case of class of transformations or a class of separable transformations. Some of these discussions, we have done while we have talked about the unitary transformations.

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: 1:36)

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

$$f(x,y) = \sum_{u,v} T(u,v) \cdot h(x,y,u,v)$$

$$g(x,y,u,v) = g_1(x,u) \cdot g_2(y,v)$$

$$= g_1(x,u) \cdot g_2(y,v)$$

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 of 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, this class of transformations will be separable obviously because $g(x,y,u,v)$, we have written as product of two 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 transformations or class of transformations are called separable as well as symmetric. And the same is also true for the inverse transformation kernel that is $h(y,u) h(x,y,u,v)$.

(Refer Slide Time: 5:13)

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

$$g(x, y, u, v) = g_1(x, u) \cdot g_1(y, v)$$

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

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 multiplicative term 1 upon capital N . so this was 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 the same but only the arguments, x in one 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 Transformation represents a specific case of a class of transformation and we had also discussed it uhh 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 DCT. Let us see what are the forward as well as inverse transform kernels of this Discrete Fourier Transformation. So now let us talk about the discrete Fourier Transformation or DCT.

(Refer Slide Time: 8:10)

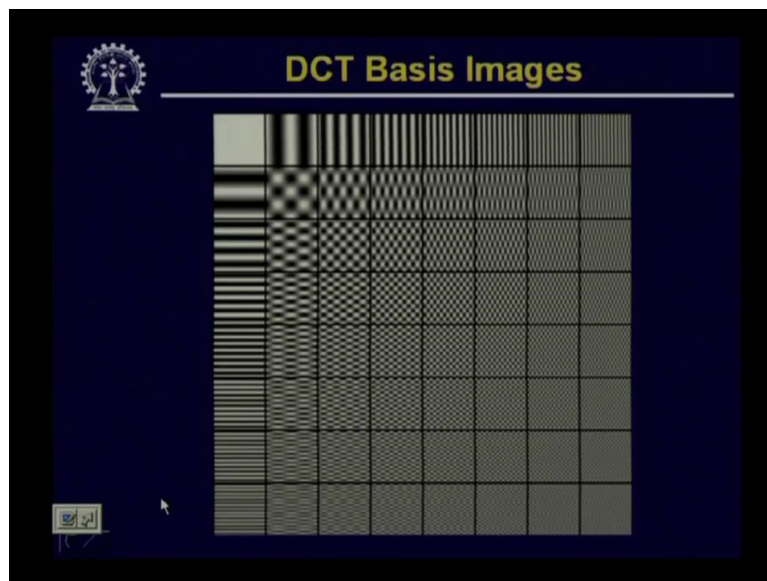
The image shows a handwritten derivation on a digital whiteboard. At the top, 'DCT.' is written and underlined. Below it, the kernel $g(x, y, u, v)$ is defined as the product of $\alpha(u)$, $\alpha(v)$, and two cosine terms: $\cos\left[\frac{(2x+1)u\pi}{2N}\right]$ and $\cos\left[\frac{(2y+1)v\pi}{2N}\right]$. This is then equated to $h(x, y, u, v)$. Finally, the definition of $\alpha(u)$ is given as a piecewise function: $\alpha(u) = \begin{cases} \sqrt{1/N} & u=0 \\ \sqrt{2/N} & u=1, 2, \dots, N-1 \end{cases}$. A digital toolbar is visible at the bottom of the whiteboard.

In case of Discrete Fourier Transformation, the forward kernel, the 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 v pi upon twice N , which is same as the Inverse Transformation kernel which is given by x $h(x,y,u,v)$. So you find that in case of Discrete Cosine Transformation, if you analyse this, you will 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 the 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 are the values of αu and αv . here αu is given by square root of 1 upon capital N where u is equal to 0 and it is equal to square root of 2 by capital N for values of u equal to $1, 2$ to capital N minus 1 . So these are the values of αu for different values of u and similar is the values of αv for different values of v .

(Refer Slide Time: 11:31)



Now using this forward and inverse uhh Transformation kernels, let us see how the basis functions or the basis images look like in case of Discrete Cosine Transformation. 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 Transformation.

(Refer Slide Time: 11:59)

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

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

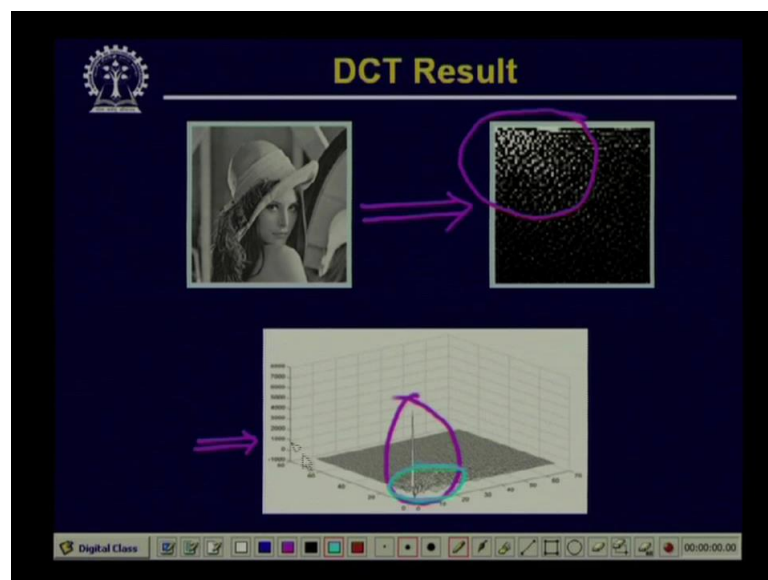
Now using these kernels, now we can write the expressions for the 2 dimensional Discrete Cosine Transformation in the form of C(u,v) is equal to alpha u alpha v, double summation f(x,y) into cosine of 2x plus 1 into pi u upon twice N into cosine of twice y plus 1 pi v upon twice N where both x and y 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 α_u times α_v times $C(u,v)$, so $C(u,v)$ is the coefficient matrix into Cosine of twice x plus $1/2$ π upon twice capital N into cosine of twice y plus $1/2$ π 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 Discrete Cosine Transformation.

Now you find that there is one difference. In case of Forward Discrete Cosine Transformation, the terms α_u and α_v were kept outside the summation, double summation whereas in case of inverse Discrete Cosine Transformation, the terms α_u and α_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 α_u and α_v , these terms are independent of these 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 α_u and α_v were kept or are kept inside the double summation operation.

(Refer Slide Time: 15:40)



So using this uhh Discrete Cosine Transformation 2 dimensional Discrete Cosine Transformation uhh let us see that for a given image what kind of output we get. So this shows this figure shows the Discrete uhh Discrete Cosine Transformation coefficients for the same image which is very popular in image processing community the image of Lena.

The results are shown in two forms, the first figure, this is the coefficients which are 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 uhh in the form of a surface in 3 dimension. Now if you closely look at these output coefficients, you will find that in case of Discrete Cosine Transformation, the energy of the uhh 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 plot.

So you find that here, in this particular case, the energy is concentrated in a small region in the coefficient 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 in case 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 in case 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 of the period in case of Discrete Fourier Transformation. And we will see later that this particular property helps uhh to obtain data compression and a smoother data compression using the Discrete Cosine Transformation and not using the Discrete Fourier Transformation.

The other property which obviously helps uhh 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) uhh value in the frequency domain in the uv-plane.

So by coding few number of coefficients we can represent or we can uhh represent most of the energy most of the signal energy or most of the image energy, so that also helps uhh in the data compression using Discrete Cosine Transformation, a property which is not normally found in case of Discrete Fourier Transformation.

(Refer Slide Time: 19:50)

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 after discussing about all these different properties of the Discrete Cosine Transformation, let us go to the uhh other transformation which we have said as Walsh Transformation. so now let us discuss about the Walsh Transform. In case of 1D, 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 $b_{n-1-i}(u)$ where the product is taken over i equal to 0 to $n-1$.

So you will find in this particular case that capital N gives you the number of samples and the lowercase 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 lowercase n is the number of bits needed to represent both x and u and in this case, the Forward Transformation kernel is given by $g(x, u)$ is equal to 1 upon capital N into product i equal to 0 to lowercase $n-1$ minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$.

(Refer Slide Time: 22:46)

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

[inv. kernel]

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

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

Now in this particular case, the convention is say 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 $b_i(x)$. So using this, the Forward Discrete Walsh Transformation will be given by $W(u)$ in case of 1 dimension will be given by $\frac{1}{N}$ upon capital N , summation $f(x)$ into product i equal to zero to lowercase 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 lowercase 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 lowercase 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 uhh only thing is the difference of the multiplicative factor $\frac{1}{N}$ 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: 25:19)

2-D

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

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

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 zero to lowercase 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 kernels so the inverse transformation kernel is given by 1 upon capital N product again i equal to 0 to lowercase 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: 27:17)

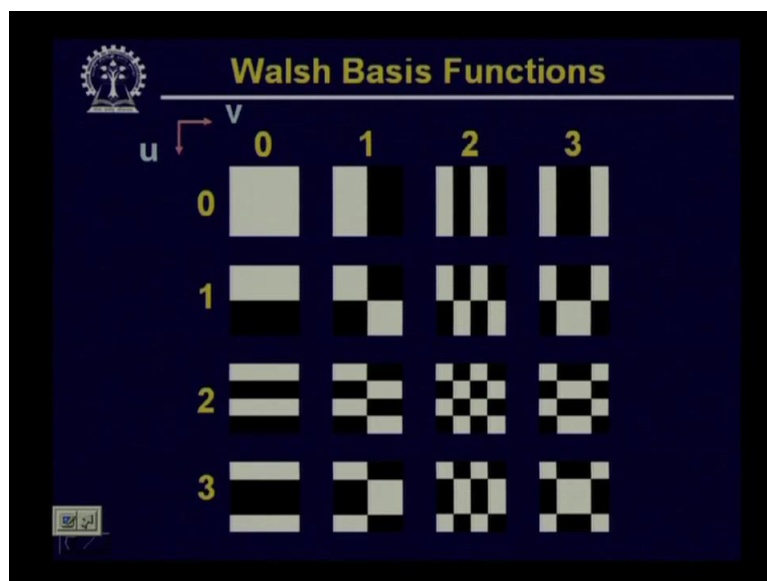
$$W(u,v) = \frac{1}{N} \sum_{x,y=0}^{N-1} \prod_{i=0}^{n-1} (-1)^{\{b_i(x) \cdot b_{n-1-i}(u) + b_i(y) \cdot b_{n-1-i}(v)\}}$$

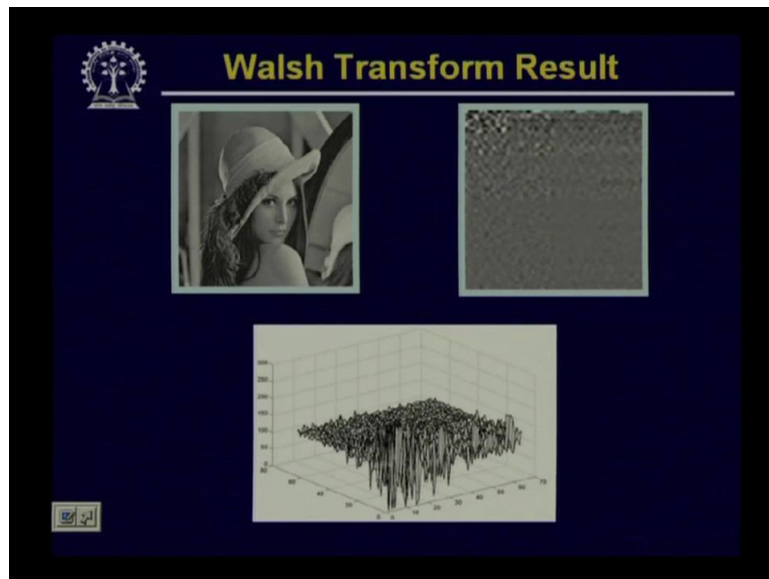
So using this Forward Transformation kernel and the inverse transformation kernel, now we find that the inverse as well as the forward Discrete Walsh Transformation can now be implemented as $W(u,v)$ is equal to 1 upon capital N , double summation $f(x,y)$ into product i equal to 0 to n minus 1 minus 1 to the power $b_i(x)$ into $b_{n-1-i}(u)$ plus $b_i(y)$ into $b_{n-1-i}(v)$.

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? It 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 uhh 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: 29:25)





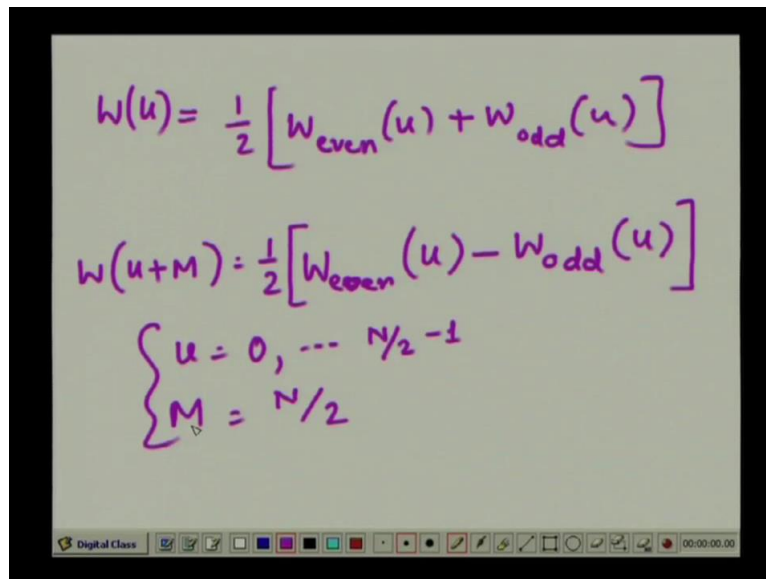
So for Walsh transformation, the basis function appears like this or the set of (basi-) basis images appear like this. Here the basis images are given for a 4 by 4 uhh 2D Walsh Transformation and if I apply this Walsh Transformation on the same image say Lena, you find that this is the kind of result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

So here again, you find that the property of this Cosine Transformations that is the coefficients near 0, they are having the maximum energy and as you go away from the origin in 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 uhh is not that strong as the compaction of energy in case of discrete cosine transformation. And by analysing this forward as well as inverse transformation Walsh Transform uhh kernels Walsh Transform Kernels. You can again find out this 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 of FFT.

(Refer Slide Time: 31:14)



The image shows a digital whiteboard 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 whiteboard, there is a toolbar with various drawing tools and a timer showing 00:00:00.00.

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 seeing that because the Walsh transformation is separable, so the same way in which you 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 Transformations.

Because the Discrete Cosine Transformation is also separable. So to illustrate the faster implementation of the Walsh Transformation, I take the 1 dimensional case, so here the fast 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 $W(u + \text{capital } M)$ 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 you find that almost in the same manner in which we have implemented the first Fourier Transformation. The Discrete 2 dimensional uhh or Discrete Walsh Transformation Fast Discrete Walsh Transformation can also be implemented in the same manner.

Here we divide all the samples of $x[n]$ of 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 two 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 $N/2$ odd samples and $N/2$ even samples. Even and odd samples can be divided into four 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 two samples, then hierarchically 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. Thank you.