Digital Voice and Picture Communication

Prof. S. Sengupta

Department of Electronics and Communication Engineering

Indian Institute of Technology, Kharagpur

Lecture - 17

Lossy Image Compression: DCT

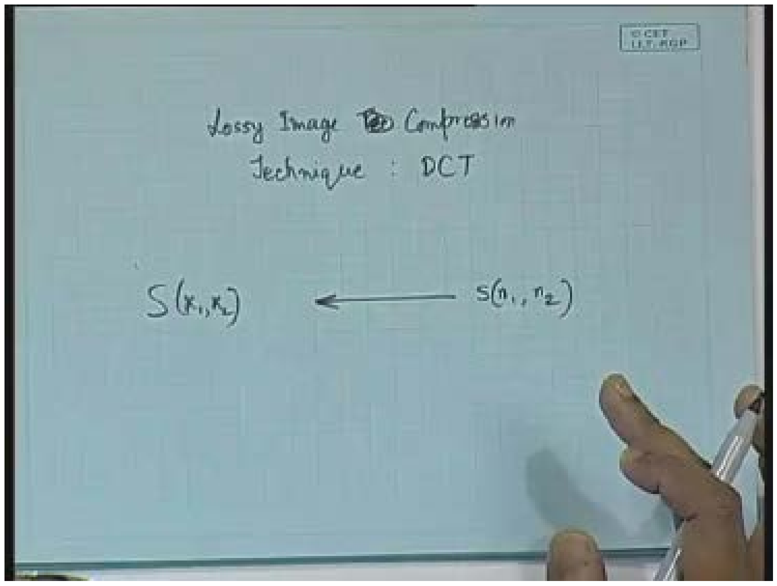
In our last lecture we were having an introduction to the basic concepts behind the image and video coding; we were discussing about certain fundamental aspects and today we will be covering the lossy image compression technique; I mean, we had made a mention of some of the techniques, so in lossy image compression lossy image compression techniques what we are going to discuss today is about the discrete cosine transform or the DCT as it is known as.

Before we can begin the topic of the specific transforms like the DCT, let us first have a basic concept about what we are essentially going to do by the process of the transformation. As I was telling you in the last class, there are two basic purposes behind the transformation. Number one that the transform coefficients means in the transform domain we are going to have the coefficients which are decorrelated and then the second aspect that we are going to exploit is that the transformed coefficients are going to have energy compaction capabilities in the sense that the energy will be contained, the bulk of the energy of the signal will be contained by only a few transformed coefficients so that an efficient compression can be reached by truncating those coefficients which are not carrying any significant energy. That is the basic philosophy with which we proceed for transform coding. And as I was making a mention, some of the transform coding techniques has emerged. Because basically in order to have decorrelation we require the usage of the orthogonal transforms and in that category we have the discrete Fourier transform, the discrete cosine transform, the discrete Haar transform, Hadamard transform, then KL transforms and also the most recently investigated ones that is to say the discrete wavelet transforms.

We have so many techniques with us but for certain reasons the DCT the discrete cosine transform happens to be quite useful in the concept of image compression and it has been adopted in the image compression standards which were proposed to............. at least the first versions of the image coding standards had made use of the discrete cosine transform.

Now essentially what we are going to do for an image transformation is that we have our basic signal as s of (n 1, n 2) so this is the intensity, this is the image intensity at the coordinate position n 1 and n 2. It is basically a two dimensional signal as we were mentioning. And what we want to do is that from this s n 1 and n 2 we would like to transform it to a transform domain where we are representing the signal by capital S and the transformed coefficients in this is we are using as (k 1 and k 2) so essentially we have to make a transformation from s (n 1, n 2) that is to say the original pixel domain or which is in the special domain we have to transform it to the S (k 1, k 2) which is in the transform domain.

(Refer Slide Time: 4:55)



Lossy Image ~~Co~~ Compression
Technique : DCT

$$S(k_1, k_2) \longleftarrow s(n_1, n_2)$$

Now what we want to do is that in the transform domain we want to do the quantization and we want to do the entropy coding on that and then want to send those coefficients over the channel. that will be done at the encoder end that is converting from the special domain to transform domain and then quantization and encoding everything in the transform domain and at the decoder end just the reversal has to take place which means to say that we will be receiving this S (k 1, k 2) and then we will be doing the entropy decoding, we will dequantize it. That means to say that whatever factor we had used in order to divide the transform coefficients we are going to multiply those factors in the dequantization process and then we will be getting back s (n 1, n 2) not exactly s (n 1, n 2) but it will be S cap (n 1, n 2).

So what we require is a reversible transformation. That means to say, a transformation should be so designed that we can go over, in the encoder we should be able to go over from the special domain to the transform domain and at the decoder we should be able to go from the transform domain to the spatial domain so it should be so whatever transformation we should use should be a reversible transformation. And not only that, if we do not use any quantization; that means to say that we do not have any lossy element in our image compression system in that case the process of encoding and decoding should not introduce any errors. That means to say that, in absence of the decoder we must have s (n 1, n 2) and S cap (n 1, n 2) to be one and the same. Now to have {ss} (00:06:49) such a kind of and so

That means to say that the reversible transformation what we had designed what we are going to consider, the process of this transformation it should be completely a reversible process. So whatever transformation we use, in the process of its inversion, in the process of inverse transformation we should not be losing any information.

Therefore, in general this transformation one can write as: S (k 1, k 2) one could express in general as a double summation process I will be putting the limits of this double summation very soon; then small s (n 1, n 2) which is to say the original pixel domain of the spatial components of the signal and then we are going to have a transformation kernel that is what we call and let us denote that kernel by a function g so g is going to be function of (n 1, n 2) (k 1, k 2) everything will be involved in the transformation process. Therefore for g, its arguments should be (n 1, n 2) (k 1, k 2).

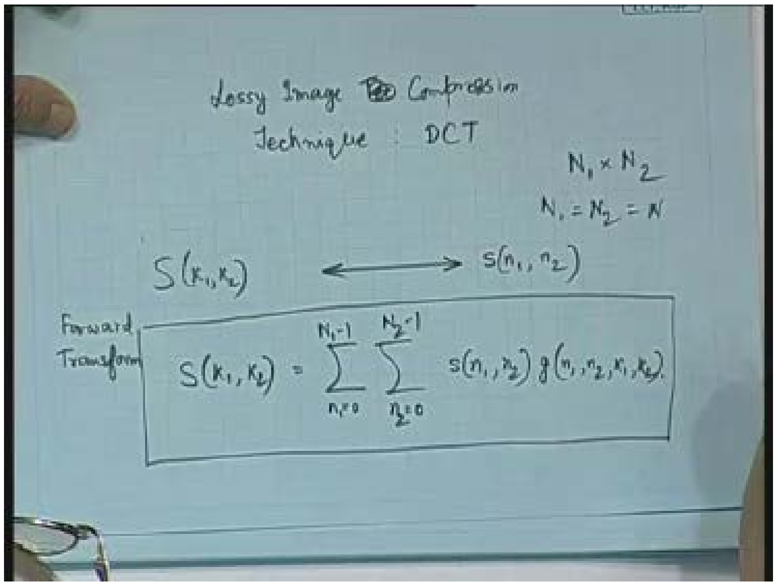
Thus, we are going to have all these things and what should be limits of the double summation process? Well, we are considering the special domain image so its indices are n 1 and n 2 so the variables in the summation process are going to be n 1 and n 2 and its limits of summation has to be considering the complete image size. So if the image size; if we consider that the image is of size capital n 1 by capital n 2 in that case our n 1 is going to be 0 to capital n 1 minus 1 n 2 is going to be 0 to capital n 2 minus 1.

In a special case where we have a square image n 1 and n 2 equal to capital N in that case it should be 0 to N minus 1, 0 to N minus 1 for n square number of pixels. this is the This is what we will be calling as the forward transformation process. So this should be the general equation for the forward transformation and we should be able to design an inverse transformation.

So what the inverse transformation is going to do is to get back s (n 1, n 2) from the transformed coefficient S (k 1, k 2) and what we are going to have is that s (n 1, n 2) that should be equal to capital that should be equal to the summation process double summation process S (k 1, k 2) into the inverse transformation kernel let us call as the h so h is the inverse transformation function h (n 1, n 2) (k 1, k 2). Here (Refer Slide Time: 10:12) the summation has to be performed over k 1 and k 2 so that it should be k 1 equal to 0 to N 1 minus 1 and k 2 equal to 0 to N 1 minus 1.

Now there is one thing which we have not mentioned yet that is to say that here this is the summation process, mind you that how many..................... so what is the computation complexity of this (Refer Slide Time: 10:39) it is going to involve n 1 times n 2 number of multiplications. So, if we are considering the order, the order of computation is going to be order of n square in the sense that if you say that N 1 is equal to N 2 is equal to N.

(Refer Slide Time: 10:53)



Lossy Image Compression

Technique : DCT

$N_1 \times N_2$

$N_1 = N_2 = N$

$S(k_1,k_2) \longleftrightarrow s(n_1,n_2)$

Forward Transform

$$\boxed{S(k_1,k_2) = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} s(n_1,n_2)\, g(n_1,n_2,k_1,k_2)}$$

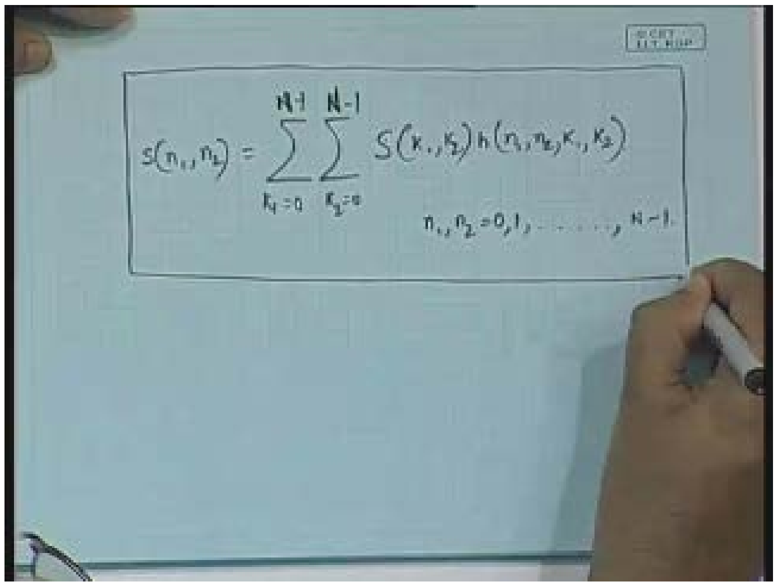
Hence, considering that, the order of computation is going to be order of N square. So at the end of the n square computation what we get we get one (k 1, k 2) value. But mind you, in the transform domain we are going to have n square number of such S (k 1, k 2) coefficients because this equation one can write for (K 1, K 2) equal to 0 to N minus 1. Or rather to say, if you write it in a general way K 1 can go from 0 to N 1 minus 1 and K 2 can go from 0 to N 2 minus 2. Or let us write it this way. In fact let us write it as 0 to N minus 1, here also 0 to N minus 1 (Refer Slide Time: 11:45) considering as a square image case; of course we are not losing the generality in any sense so S (k 1, k 2) we are having and N square number of such equations are possible, that is correct.

So, when n square number of such equations are possible; that means to say that, for each of this n square number of equations we are going to have n square number of multiplications so the order of computation that we clearly see here is going to be order of n to the power 4. This is a two dimensional so that is how it has increased. We know that, for a discrete Fourier transform one dimensional discrete Fourier transform we all had seen in our Digital Signal Processing courses that it is order of n square that is what we get but in this case it is order of n to the power 4.

Of course there are many fast algorithms for computation of these transformations. So those who have gone through the Digital Signal Processing course are familiar with tools like the FFT the first Fourier transform which significantly reduces, that instead of the n square order of n square for the one dimensional signal case you can make it into order of n log n for using the FFT algorithm.

Hence, now this S (k 1, k 2) means to say that k 1 and k 2 are going from 0 to n minus 1. So, in this case also, in the summation process we are writing (k 1, k 2) going from 0 to N minus 1 and we are getting s (n 1, n 2). So, as we did in the previous equation, even here also it should be possible for us to write down N such equations for each of these variables n 1 and n 2 which means to say that there are n square number of such equations which are possible. So this equation that we wrote just now this is going to be our inverse transform.

(Refer Slide Time: 13:54)



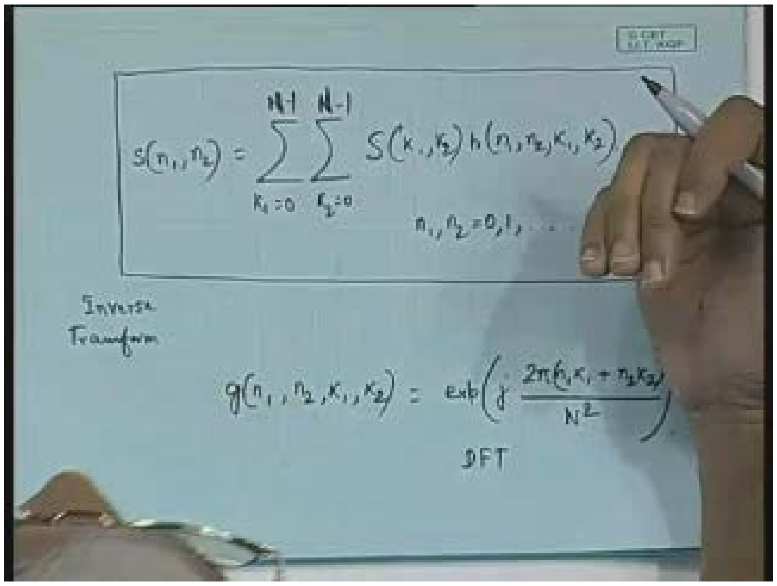
$$s(n_1,n_2) = \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1} S(k_1,k_2)\,h(n_1,n_2,k_1,k_2)$$

$$n_1, n_2 = 0, 1, \ldots\ldots, N-1.$$

We have the forward transform and we have the inverse transform and we did not have to write S cap (n 1, n 2) over here because the transformation that we are using is exactly reversible which means to say that we should get back s (n 1, n 2) exactly. It is not the transformation process which is lossy, but please remember always that it is the quantization process which is the only lossy element in the system. And we are doing all this why we are going through all this transformation process is simply because that in order to have the decorrelation of the coefficients. So S (k 1, k 2) should be decorrelated and again as we say that few of the coefficients of this S (k 1, k 2) would carry most of the energy that means to say that having the energy compaction property fulfilled. So now, so given that so this is the generalized form.

Now as a special case one can consider any type of kernel (Refer Slide time: 15:01). So let us say, for example, if this g (n 1, n 2) (k 1, k 2) we have the function let us say exponential to the power, say if we choose g (n 1, n 2) (k 1, k 2) to be the exponential of j and then we write it as 2pi n 1 k 1 2pi (n 1 k 1 plus n 2 k 2) divided by N square; so if we write that in that case what is the transformation that results so that goes so that becomes the discrete Fourier transform because it is a complex exponential; complex exponential kernel basically leads to the discrete Fourier transform or the DFT.

(Refer Slide Time: 15:59)



$$\boxed{s(n_1,n_2) = \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1} S(k_1,k_2)\,h(n_1,n_2,k_1,k_2) \qquad n_1,n_2=0,1,\ldots}$$

Inverse Transform

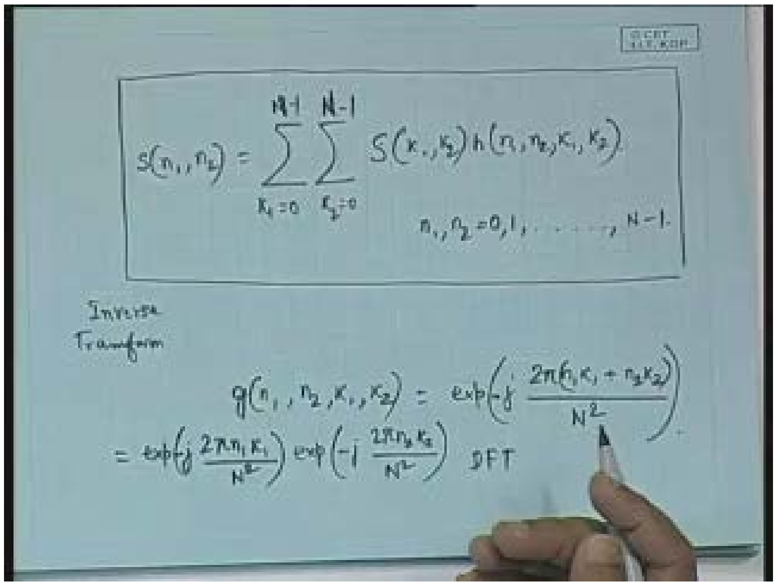
$$g(n_1,n_2,k_1,k_2) = \exp\left(j\,\frac{2\pi(n_1k_1+n_2k_2)}{N^2}\right)$$

DFT

If instead of exponential we take a cosine kernel in that case we get the discrete cosine transform. There are other transforms also which should be possible; we did not use anything specific over here but let us study some of the very important properties of this g (n 1, n 2) (k 1, k 2) which we very often come across.

The first of this property which we quite often make use of and come across is that, just say for example, look at this kernel that is to say the discrete Fourier transform kernel (Refer Slide Time: 16:38). One can write this kernel as a product of this n 1 term and the n 2 term which means to say that it is possible for us to write it down as exponential............ actually it is customary to write the minus in the forward transformation process and plus in the inverse transformation process. So it is minus j 2pi (n 1, k 1) divided by N square divided by N square into exponential minus j 2pi (n 2, k 2) divided by N square; just express this as a product of these two, realize it is the same thing. But in the process of realization we get an advantage in the sense that we can implement first a row-wise transformation and then we can use a column-wise transformation. So such kind of properties which I just now described is going to be a is going to be mentioned as the separable transform. So this is called as the separable transform.

(Refer Slide Time: 17:22)



$$\boxed{s(n_1, n_2) = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} S(k_1, k_2)\, h(n_1, n_2, k_1, k_2) \qquad n_1, n_2 = 0, 1, \ldots\ldots, N-1.}$$

Inverse Transform

$$g(n_1, n_2, k_1, k_2) = \exp\left(-j\, \frac{2\pi(n_1 k_1 + n_2 k_2)}{N^2}\right)$$

$$= \exp\left(-j\, \frac{2\pi n_1 k_1}{N^2}\right) \exp\left(-j\, \frac{2\pi n_2 k_2}{N^2}\right) \text{ DFT}$$

Hence, in mathematical terms the separability inevitably means that for the case of the separable transform it is g (n 1, n 2) (k 1, k 2) we should be able to write this as a product of two kernels g

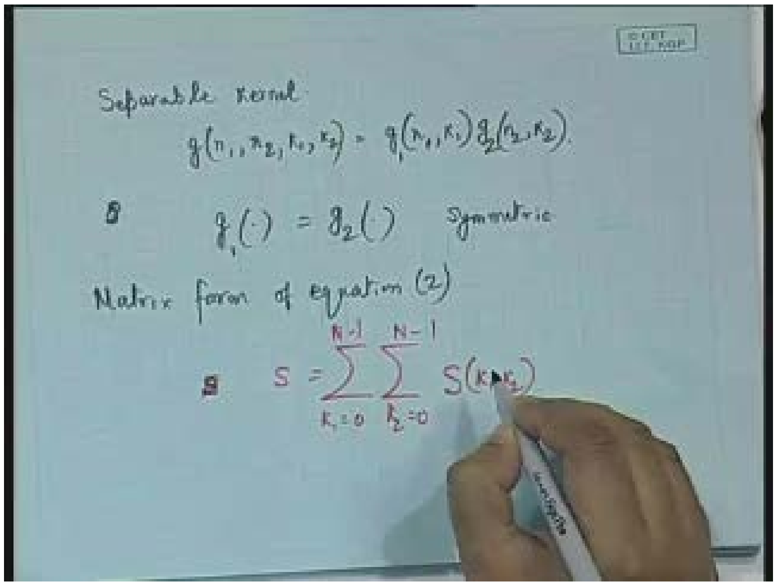
1 we can say (g 1, n 1, k 1) into (g 2, n 2, k 2) where in general g 1 and g 2 could be different functions also. So if this is fulfilled that means to say that we can express as a product like this we will call it as the separable kernel. This is a separable kernel and we can have a and in a specific case when we have g 1 and g 2 to be the same function; if we have g 1 g 2 to be the same function then we call this as symmetric.

You see that clearly in the DFT (Refer Slide Time: 19:05) you can very clearly see that this 2pi (n 1, k 1) and 2pi (n 2, k 2) the function is basically of the form of 2pi n k by N square so that is the general form of the function which is same as that for the g 1 which is this one and for the g 2 which is this one. So this is a typical case an example. The DFT is a typical example of a symmetric and separable kernel. Likewise the DCT the discrete cosine transform as we will clearly see, I mean, just after sometime when we take up the DCT we are going to see that it is going to fulfill the very similar properties: the separablity and the symmetric; it is a symmetric and separable kernel.

Now let us have a look at this equation that is to say the inverse transformation equation. Let us again number these equations. The forward transformation let us call as the equation number 1 and the inverse transform let us call this as equation number 2. Now let us have a look at the equation number 2.

What we are basically having is that there are n square number of such equations. So it is possible for us to combine this n square number of equations and write it in a compact matrix form. because after all, if we compute s (n 1, n 2) for (n 1, n 2) varying from 0 to N minus 1 for both the variables n 1 and n 2 what we are going to achieve is that it is going to have all the..................it is going to compose an N square matrix which means say that n by n matrix will be composed after that. So combining all these different N square number of equations that we get from equation number 2, one can write.................. a matrix form of equation number 2, one can write as; we are going to write it as S and this S is going to be the S matrix so that is so we can write for the matrix equation a different color so this S is the matrix S which is going to be summation, here the double summation is for k 1 is equal to 0 to N minus 1 and for k 2 is equal to 0 to N minus 1 and then this is S (k 1, k 2) as it is, this S (k 1, k 2); well, this S (k 1, k 2) is after all a scalar quantity but it is to be multiplied by what; h (n 1, n 2) (k 1, k 2).

(Refer Slide Time: 22:10)



Separable kernel

$$g(n_1, n_2, k_1, k_2) = g_1(n_1, k_1)\, g_2(n_2, k_2)$$

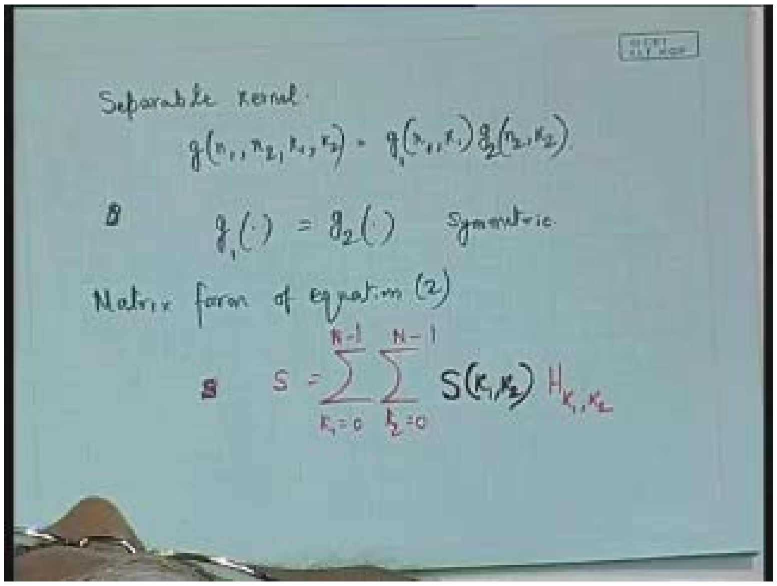
$$g_1(\cdot) = g_2(\cdot) \quad \text{Symmetric}$$

Matrix form of equation (2)

$$S = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} S(k_1, k_2)$$

Now, for a particular k 1 and k 2; okay, this (k 1, k 2) (Refer Slide Time: 22:19) is a variable index that goes within the summation process but n 1 and n 2 this n 1 and n 2 comes from this. That means to say that depending upon the elements of this matrix; because now this n 1 and n 2 that composes the index of the matrix elements. So the index of the matrix elements will dictate h (n 1, n 2) component of h which means to say that, for every value of (k 1, k 2) for each k 1 and k 2 we are effectively going to have a matrix, we are effectively going to have a matrix whose elements would be given by h (n 1, n 2). So, for a particular value of k 1 and k 2 keeping k 1 and k 2 the same we are going to have h (n 1, n 2) where n 1 and n 2 will decide the matrix elements. Thus, using that, one can write this H k 1, k 2 also as matrix. So this is going to be a matrix and this is H k 1, k 2. So, just to avoid the confusion let us write this as a scalar quantity S (k 1, k 2) this is a scalar quantity. So whatever I am writing with red is the vector of the matrix quantity. So this is an n by n matrix S and this H k 1, k 2 this is also a matrix of dimension n by n.

(Refer Slide Time: 24:10)



Separable kernel

$$g(n_1, n_2, k_1, k_2) = g_1(n_1, k_1)\, g_2(n_2, k_2)$$

$$g_1(\cdot) = g_2(\cdot) \quad \text{Symmetric}$$

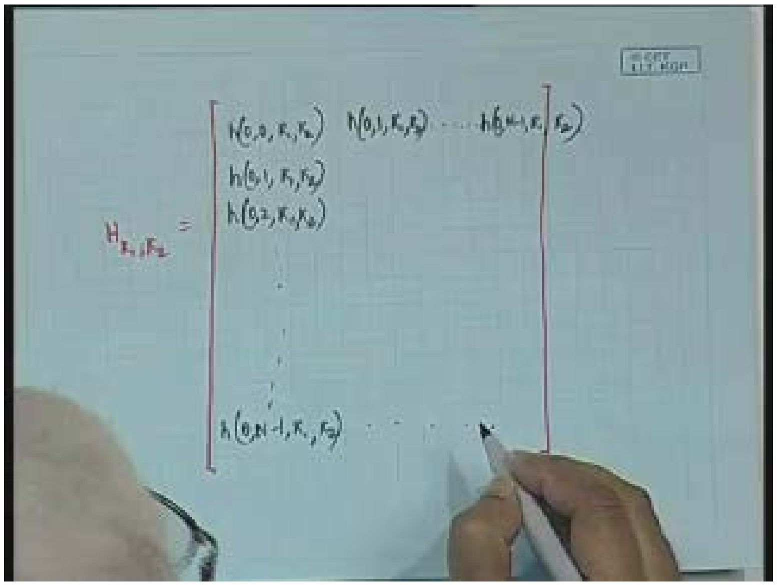
Matrix form of equation (2)

$$S = \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} S(k_1,k_2)\, H_{k_1,k_2}$$

Now, what is this matrix?

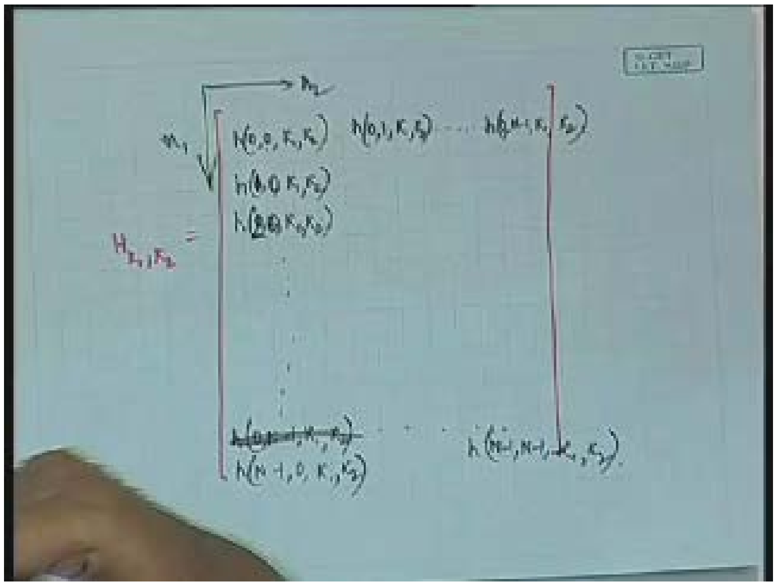
We can realize this matrix using the transformation kernel; in this case the inverse transformation kernel, using the inverse transformation kernel we get this. But after all this is also an n by n image. In fact if we write the elements of this matrix this H k 1, k 2, we are going to write, I mean, this is this is the way the matrix will be composed off that is to say this H k 1, k 2 what we are writing essentially this is a matrix whose elements would be given by H (0, 0) (k 1, k 2), this k 1 and k 2 is going to be the same for all the elements. So what we are going to have is that there will be N square number of such matrices. This is one of those matrix for a particular value of (k 1, k 2). This is h (0, 0), the next element should be h (0, 1, k 1, k 2) and like this it goes for h (0, n minus 1, k 1, k 2) and on this side we are going to have h (0, 1, k 1, k 2), here it will be h (0, 2, k 1, k 2) and the last one is going to be h (0, n minus 1, k 1, k 2); did I make any mistake?

(Refer Slide Time: 25:36)



[Conversation between Student and Professor – Not audible ((00:25:38 min))] h) (1 {sa} (00:25:44) sorry sorry yes yes yes yes I mean, I just............. yeah, it is 1, 0 2, 0 yes, (Refer Slide time: 25:49) because it is the row index so in this in this direction I assume that n 1 is changing that is correct and thank you for pointing out this mistake; this is n 2 direction so n 2 is this one this element is changing and in this case it is 1, 0 2, 0 and this is going to be h (N minus 1, 0, k 1, k 2) thank you for the correction.

(Refer Slide Time: 26:15)



And in this case the last element is going to be h (N minus 1, N minus 1, k 1, k 2). Now this is a matrix. And when we change k 1 and k 2, the first matrix is going to be for k 1 is equal to 0, k 2 is equal to 0; the next matrix is going to be for the other values of k 1 and k 2 which will be decided by the summation process. So effectively what we are doing is that we are taking this h (k 1 and k 2) matrix, so, for a particular value of (k 1, k 2) whatever we get we are multiplying that with a scalar coefficient S (k 1, k 2).

After all, what this matrix h (k 1 and k 2)?
It is an image because S is an image and h (k 1, k 2) is also an image. But this image is actually a basis image because this is not an image which is obtained from any camera or which is not obtained from any natural image and no such source is there for this image but this is an image which could be realized by using the transformation kernel itself. So this is a kind of a basis image. So H (k 1, k 2) is a basis image and what we are doing is that this basis image we are multiplying by the particular coefficient S (k 1 and k 2).

So, for every value of k 1 and k 2 we are having a basis image. So how many such basis images we are having?

We are having existence of n square number of such basis images. So, essentially one way of looking at the whole transformation process is that the transformation process basically tells you or gives you the clue to realize an image through a composition of basis image multiplied by its equivalent coefficient. So it is an image which you are multiplying by S (k 1 and k 2) a scalar quantity and then adding it up. And by adding that up, by adding such n square number of weighted images you are getting the image with which you are working with. So this is the basic behind the image transformation. And essentially the discrete cosine transform which we are going to discuss now follows this same basic principle which means to say that if we know the kernel we should be able to compose our basis images. But before we go over to the discrete cosine transform or any other or any specific transformation process in general let us see that in what manner are we going to apply the transformation.

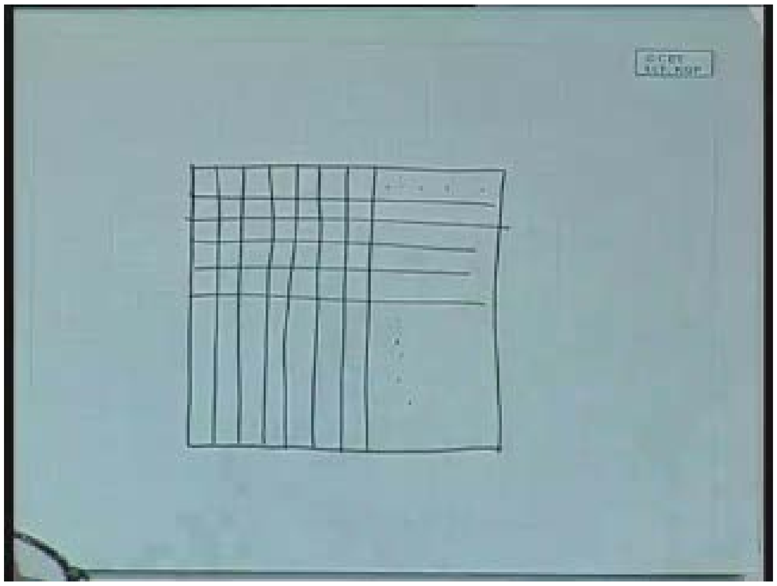
Now, in this case both for equation number 1 as well as for equation number 2 we had seen that this n basically this n 1 and n 2 we consider to be the same. So this n is nothing but the size of the image so it is pertaining to the dimension of the image so that n by n is the size of the image.

Now what we are saying is that this transformation is applied over the entire image if the size of the image is say 1024 by 1024. In that case for the entire 1024 by 1024 we are going to apply this.

So what for we are applying?
We are applying it in order to get the correlation properties that exist in image. Now it is a big image so the property of the image is not going to be the same everywhere; there will be some regions of the image where we will be having too much of details and there will be some regions of the image where the details will be much less. In that what we are going to get is that, I mean, over such a kind of......... it is looking at an image with its globality, globality in the sense that we are considering the entire image and exploiting its correlation properties. But actually speaking, if you consider the image in parts if you divide the image into small small blocks and try to study the property of the pixels that exist within the block within those individual blocks you will be able see that within those blocks the pixel values would tend to have a more homogeneity in the sense that there is more of a correlation between the pixels which are within the block rather than

the kind of global correlation which we will be realizing using this apart from the fact that when we think of any hardware implementation for a 1024 by 1024 size realizing a hardware transformation block for this it is going to be herculean task. So, before performing the transformation we should be able to apply what is called as the we should be able to subdivide the image into such small small parts or blocks which means to say that conceptually we should go ahead like this that we should have an image. Therefore, let us say that this is our image (Refer Slide Time: 32:42) and this image is subdivided into these many blocks and so on. Anyway this is enough. This way we will be further subdividing, this way also will be subdividing.

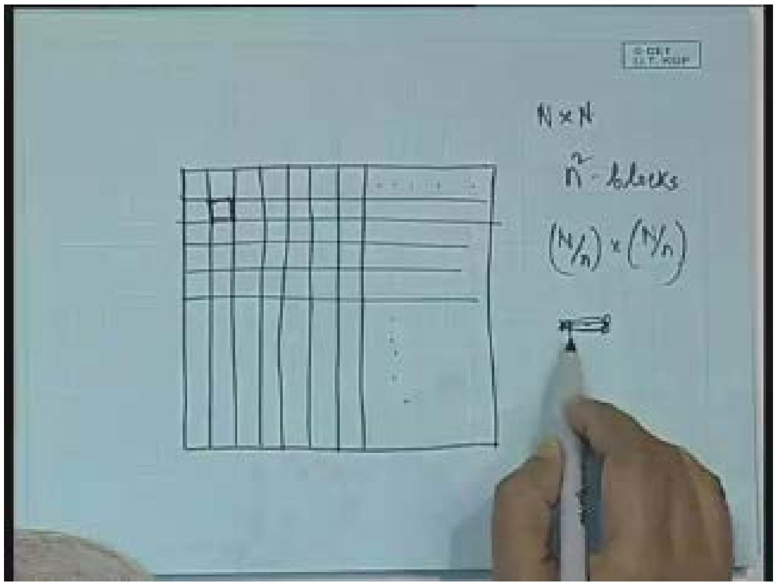
(Refer Slide Time: 33:06)



So, to take each of this, so what is this one; this one is a block of image. So if we have the total image size to be N by N and if we make it into total; if we are going to have n square number of such blocks, if we subdivide, if we partition the image into n square number of such blocks then each of the blocks is going have a size of what; capital N by small n into capital N by small n, so we are going to partition the image into these many things. In fact a typical number that the standard makers had thought of is to have n is equal to 8 or in some of the recent transforms they are also using n is equal to 4. So why such small numbers of 4s and 8s? It helps us in two ways.

First is that as I was mentioning about the hardware realization that hardware realization is manageable with a smaller size transformation. So definitely whenever we are having n equal to 4 or n equal to 8 we can make a hardware block of that and there is no difficulty in replicating that hardware block; we can do a kind of parallel processing. Because if the image is subdivided into such kind of non-overlapping blocks; in fact the manner in which we have subdivided is by having non-overlapping blocks. So, by having such kind of non-overlapping blocks we are going to........ I mean, we can have a parallel processing like say, for example; if we are going to have n square number of blocks we can assign n square number of processers, each processer can do a computation of performing a transformation, a GCT or any other transform for a block which is just the size small n capital N by n capital N by n.

(Refer Slide Time: 35:30)

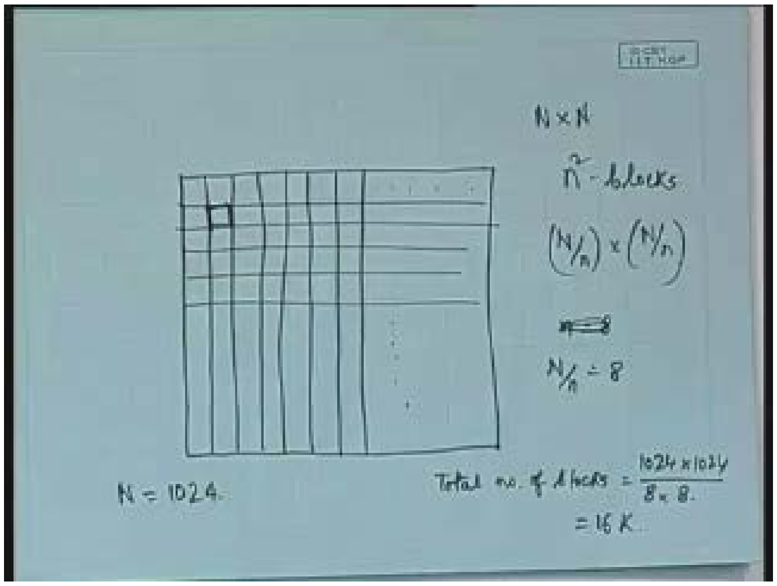


It is not small n equal to 8 sorry I made a mistake; it is capital N by small n equal to 8which means to say the block size block size is going to be 8 which means to say that as a very typical example if we have let us say n equal to 1024 and if we have the block size to be 8 by 8 in that case the total number of blocks total number of blocks that we are going to have is going to be 1024 by 1024 divided by 8 by 8 so it is a 128 by 128 what is................. the number is that 128 by 128 realizes 16000 16000 times so 16 K 16 K number of blocks are existing. Still it is a large amount of computation but remember that in this particular example we have got 1 megapixel so that itself is a very large number.

Now what we are going to consider is that how to take up the transformation for such small blocks. Yes as I was mentioning that computational advantages is one point and the second is that when by subdividing it into smaller blocks we are able to use the correlation that is existing within the block in a much better way. So there is more exploitation of redundancy......... the individual blocks of smaller size is going to exploit the redundancy more as compared to a single large mega block type of a thing; I mean, we can take the entire image to be a mega block, we are not doing that but we are having computational advantage, we are also having the exploitation of correlation in a better manner.

Now, in the case of the discrete cosine transform we are going to use the same principle; subdividing into blocks. Before we come to the discrete cosine transform just a few words about the other type of transforms and in this connection I would like to make a mention of the KL transform Karhunen–Loeve transform KL transform.

(Refer Slide Time: 41:08)

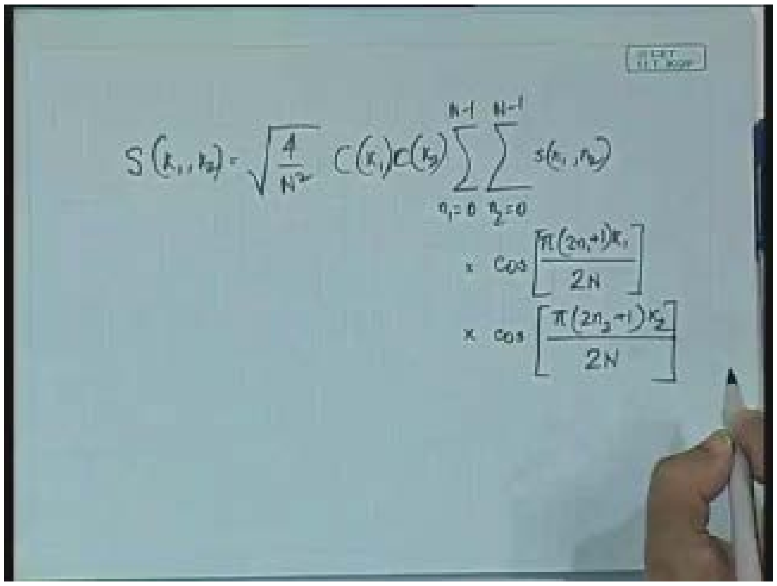


Now incidentally KL transform is said to be an optimal transform and it is optimal in the mean square error sense. What it means to say is that, basically in a computational manner........... I am not going into the details of this because there is a good reason because of which KL transform despite we mentioned as optimal transform is not used in practical circuitry that is because of its bulk computational disadvantage is extremely poor in terms of its computational efficiency that is one of the reasons and because it is image dependent you cannot have a readily available kernel prepared; cosine values you can readily make it prepared, exponential values you can make it readily prepared but you cannot keep prepared something which is image dependent where the kernel is to be realized from the image. So, as a result of that the KL transform is not very popular but essentially what it does is that this also will subdivide the image into blocks and for individual blocks of size n by n.

We are going to compute the covariance matrix; yes, we are going to compute the mean and subtracting the mean from the original block and then having a covariance. We are going to have that as the covariance matrix and then we are going to compute the Eigenvalues of the covariance matrix; Eigenvalues or the Eigenvectors of the covariance matrix and then we are using........................

So, by ordering those Eigenvalues and considering only the first few Eigenvalues and regarding the other Eigenvalues which are not corresponding to the bulk of the energy term one can have a very good efficiency. But it is seen that by having a transformation like the DCT that is the discrete cosine transform we approach the performance almost as close as that of the KLT in most of the practical images. Since that is the situation, in that case why do not we go in for a transformation like the DCT because DCT is after all realizable using the trigonometric directly and in fact for DCT the kind of transformation that we are going to use is like this that in this case S (k 1, k 2) could be written as.............. that is straightaway so root over 4 by N square times C (k 1, k 2) sorry C (k 1) C (k2) into a double summation n 1 is equal to 0 to capital N minus 1 again yet another summation n 2 is equal to 0 to capital N minus 1 small s (n 1, n 2) into cosine so this multiplied by........... so cosine pi(2 n 1 plus 1) k 1 divided by 2N. And there is yet another cosine term because it is separable. This cosine term involves only n 1 and k 1 and its product cosine term involves n 2 and k 2. So it is pi (2 n 2 plus 1) k 2 divided by 2N.

(Refer Slide Time: 42:32)

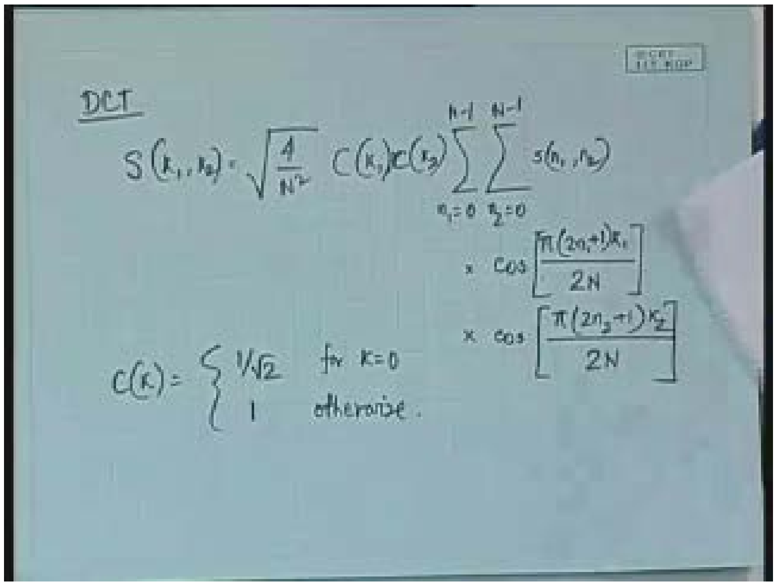


$$S(k_1,k_2)=\sqrt{\frac{4}{N^2}}\,C(k_1)C(k_2)\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}s(n_1,n_2)$$
$$\times\cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right]$$
$$\times\cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right]$$

And in this case C (k) is defined as 1 by root over 2 for k is equal 90 and this is equal to 1 otherwise. That means to say that only the border the coefficients S (k 1, k 2) which are existing in the borders have a different multiplication coefficient as compared to those terms which are there in the middle. For terms in the middle; that means to say that where k 1 is greater than 0 and k 2 is greater than 0 we are going to have C (k 1) C (k 2) both as 1 which means to say that this is a straight multiplication by 1 for this term.

Now this is the DCT, this is what we are going use off and look at this kernels (Refer Slide Time: 43:39) cos and here also cos which means to say that if you take this S (k 1, k 2) and take yet another (k 1, k 2) let us say you take k 1 prime, you take k 2 prime and you get another transformation like this and you try to correlate S (k 1, k 2) with S (k 1 prime, k 2 prime) and if you are going to add it up you are going to have a value equal to 0 because it is going to be an orthogonal transform. This trigonometric is this trigonometric term is really the tool that gives you the orthogonality in this case. So this is the DCT, the discrete cosine transform.

(Refer Slide Time: 44:49)



DCT

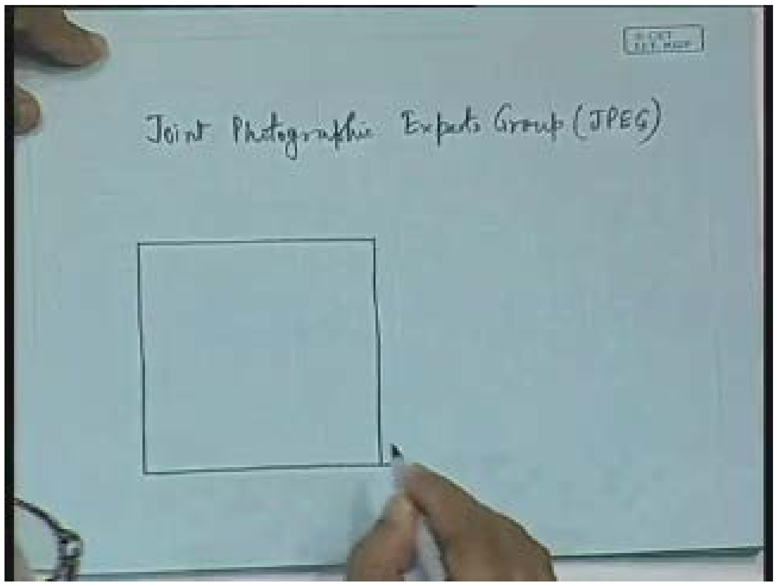
$$S(k_1, k_2) = \sqrt{\frac{4}{N^2}}\, C(k_1)C(k_2) \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} s(n_1, n_2) \times \cos\left[\frac{\pi(2n_1+1)k_1}{2N}\right] \times \cos\left[\frac{\pi(2n_2+1)k_2}{2N}\right]$$

$$C(k) = \begin{cases} 1/\sqrt{2} & \text{for } k=0 \\ 1 & \text{otherwise.} \end{cases}$$

We start with n by n number of coefficients and we end up with n by with n by n number of coefficients and we start with n by n number of image or pixels, we end with up n by n coefficients. Hence, this is the discrete cosine transform and it is seen that the DCT is giving us a performance which is as close to that of the KL transform for most of the images, very close to that and it has got very good energy compaction capability. So that makes it a real good candidate to be used for the effective compression of the images. In fact because of this reason this DCT was adopted in the very first image processing image compression standard that was designed way back in 1988 or 89 by the first joint photographic experts group joint photographic experts group or what is called as the JPEG. So JPEG, the first still image compression standard had made use of this DCT as the transform.

Now, essentially what we are going to have in the process of DCT is that the first step is to subdivide the image into such non-overlapping blocks of size 8 by 8 or 4 by 4 whatever. I mean, the essential idea is to divide it into non-overlapping blocks and then for each of these non-overlapping blocks we are going to apply this transformation and after applying this transformation what results?

(Refer Slide Time: 46:55)



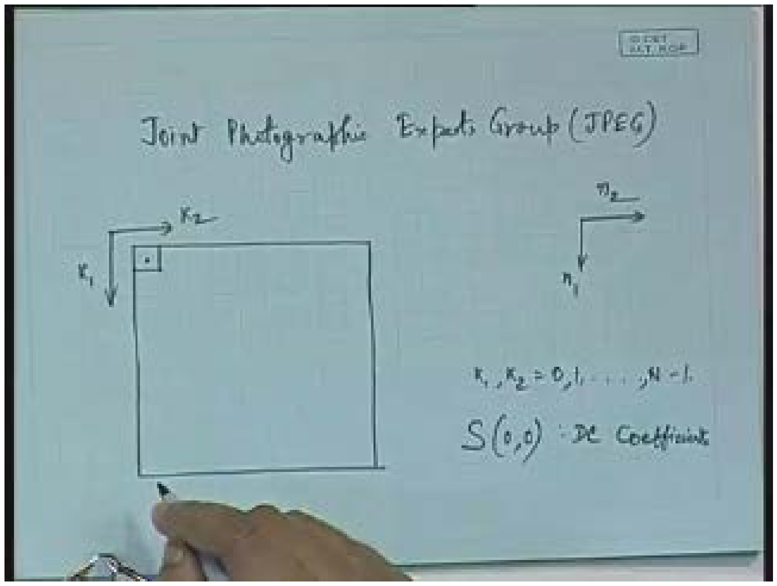
If this is the transformed array, so this is the transformed array, the transformed array also we are going to represent like this that if we took this to be our n 1 direction and this to be our n 2 direction (Refer Slide Time: 47:10) in that case in the transform domain also we are going to this to be our k 1 direction and this to be our k 2 direction.

So what is this term; what is k 1 equal to 0, k 2 is equal to 0?
Now this particular element in this matrix; so k 1 and k 2 is also the matrix that is having n by n number of elements. So with k 1 equal to 0 and k 2 is equal to 0 we refer to this particular element in the matrix.

So this is S (0, 0) so what is the special significance of this term? You see, when we put (0, 0) (Refer Slide Time: 47:59) when we put k 1 as 0 k 2 as 0 what do we get?
First is that we get C of 0, C of 0 so 1 by root 2 multiplied by 1 by root 2 which means makes it a half so half of a quantity and in this case root over 4 by n square leads to 2 by n so it is 2 by n into half which means to say that overall 1 upon n and what are we getting in that process; we are getting S (n 1, n 2) and here this cosine term with k 1 is equal to 0 makes it 1, this with k 2 is equal to 0 makes it 1 so it is like summing up all these S (n 1, n 2) values just a simple

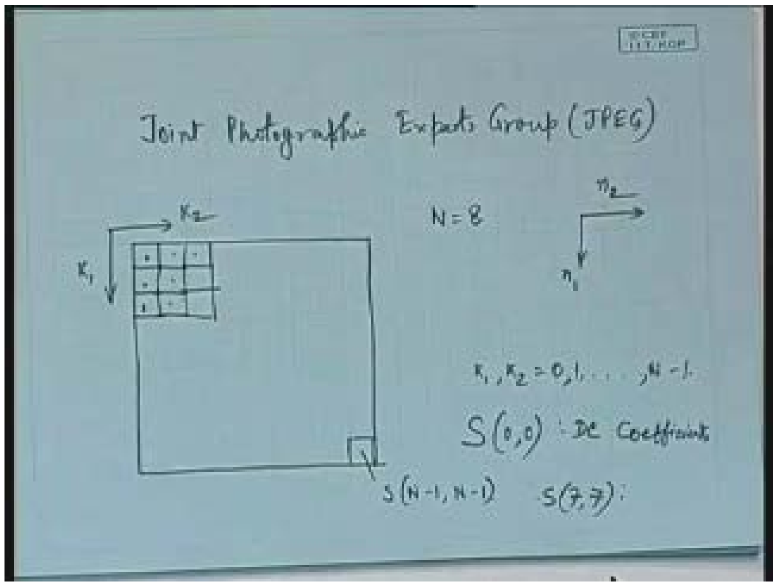
summation, summing up which means to say that this leads to some kind of an average value so this leads to the average value or it is the DC value.

(Refer Slide Time: 49:05)


Joint Photographic Experts Group (JPEG)
k_2
k_1
n_2
n_1
$k_1, k_2 = 0, 1, \ldots, N-1$
$S(0,0)$: DC Coefficient

So we call this (k 1, k 2) to be 0 as the DC coefficient. So S (0, 0) is the DC coefficient and all other coefficients will be referred to as the AC coefficient. So this coefficient is going to be S (1, 0), this will be (1, 0), this will be (0, 1), this will be (1, 1), this will be (0, 2), this will be (1, 2), this one will be (0, 2), this is (2, 0); this is (1, 0), this is (2, 0) (Refer Slide Time: 49:38). This is the way we will be representing and this last one that is to say the bottom right, so the bottom right will be S (N minus 1, N minus 1). If it is 8 by 8, if N is equal to 8 in that case this is S (7, 7); S (7, 7) is the last AC coefficient.

(Refer Slide Time: 50:15)



Joint Photographic Experts Group (JPEG)

k_2
k_1
n_2
n_1
$S(N-1, N-1)$

$N = 8$

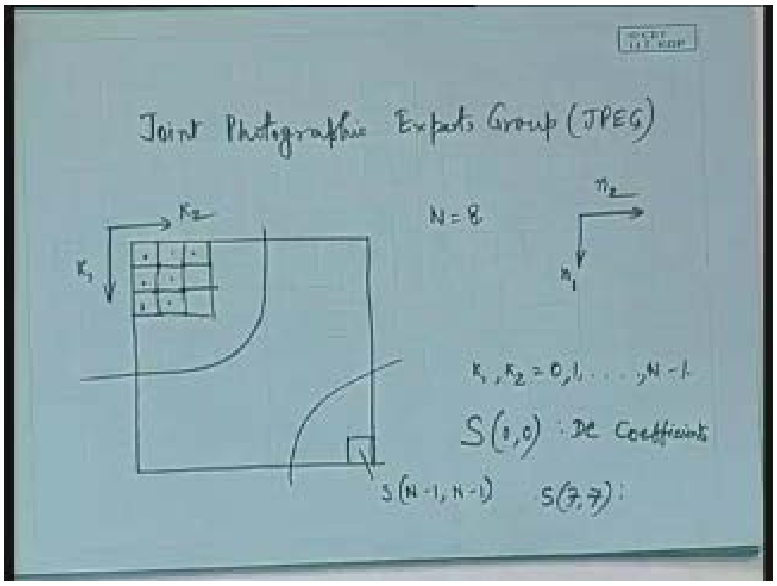
$k_1, k_2 = 0, 1, \ldots, N-1.$

$S(0,0)$: DC Coefficient

$S(7,7)$:

Now, in terms of the frequency, frequency meaning the spatial frequency here of course this is the highest frequency component; it is highest frequency both in the k 1 direction and in the k 2 direction; it is in the k 1 and the k 2 direction that is what we are going to have so all the low frequency components are centered around this region and all the high frequency regions are in this region. So we will be expecting that the maximum energy will be in this zone and this will be the AC which will carry much less of energy.

(Refer Slide Time: 51:09)


Joint Photographic Experts Group (JPEG)
k_2
k_1
N = 8
n_2
n_1
$k_1, k_2 = 0, 1, \ldots, N-1$
$S(0,0)$ · DC Coefficients
$S(N-1, N-1)$
$S(7,7)$

Now how do we pick up the coefficients?

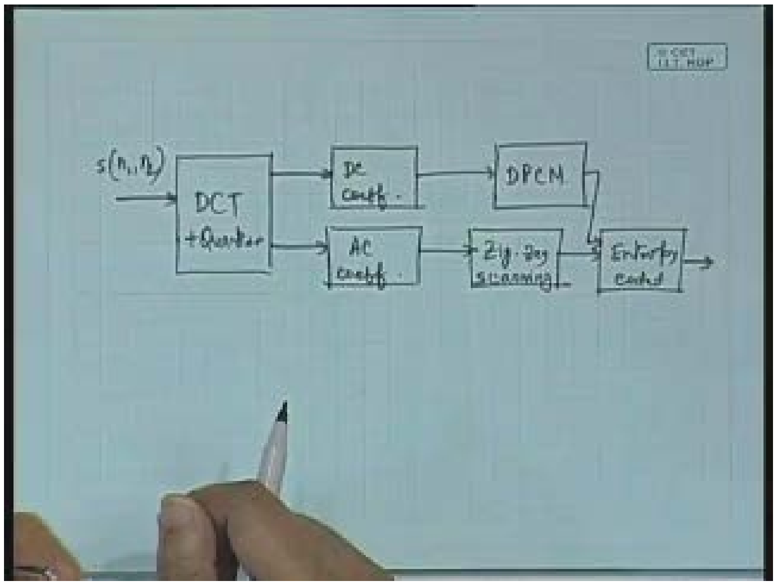
To pick up these coefficients we are going to traverse in this direction. this is the DC quantity (Refer Slide Time: 51:24); next we go......... over to from (0, 0) we are going over to the next pixel which means to say from that from (0, 0) we go to (0, 1) and then we go over to (1, 0), then we go over to (2, 0), then we go over here, then we go over here, then we go here so this is the manner in which we should travel in order to pick up the coefficients.

So what results is that this process of scanning what are we picking up?

We are picking up the coefficients in a zigzag manner. This DCT array what we get should be scanned in a zigzag way. So this zigzag manner of picking up the coefficients ensures what; that we start by picking up the low frequency components and we end up with the high frequency components. So as a result of that so and Then after picking up these coefficients what we do is that whatever correlation exists between this way of travelling even that correlation also will be exploited by going through an entropy coding of the scanned zigzag scanned coefficient. Thus, the overall block diagram that we going to have out of this is something of this nature so let us have a look at the overall block diagram.

We can just say that, the first step is that the input is s (n 1, n 2) and then we are going to have the DCT and the DCT is going to give us the DC coefficient one DC coefficient and the rest as the AC coefficient so let us separate them out so let us call this to be the DC coefficient and this to be the AC coefficients. Now the AC coefficients are actually scanned in the zigzag way. So, after AC coefficients there is a zigzag scanning and after zigzag scanning these are entropy coded. The DC coefficients also could have been included in this scanning process but the DC coefficients are treated in a bit of a different way, why you know?
It is because the DC coefficient is remarkably different from that of the AC coefficient. But DC coefficient is after all going to pick up a global property of the two dimensional signal which means to say that it is signifying the average intensity and average intensity is not going to change from one block to its neighboring block so there is a good degree of correlation between the DC of this block and the DC of the next block so one can make use of that redundancy factor by encoding the DC coefficients through a differential pulse with modulation process.

(Refer Slide Time: 56:01)



It is a DPCM which we can use and all this DPCM also the DPCM coefficients also could be entropy coded. Finally the entropy coder output is going to yield us the bit stream which can go into the channel. This is the overall compression system. Of course I missed one block so let us say that this is DCT plus quantizer because, if we are going to achieve compression then obviously we are going to quantize the coefficients and then use it.

So next class I will tell you little bit more about this quantizer that what type of quantizer would you use in the DCT and then we will go over to the other transforms. Thank you for today.