

Computer Graphics
Prof. Sukhendu Das
Dept. of Computer Science and Engineering
Indian Institute of Technology, Madras
Lecture - 42
Digital Image Processing, Image Compression, Jpeg Enhancements

Welcome back to the lectures on computer graphics. We are in the very last phase of the lecture series. So far we have discussed various aspects of computer graphics starting from devices to transformations to clipping, line drawing, rendering, hidden surface removal and in the last class we have seen a few special miscellaneous topics dealing with aliasing, animation, soft object modeling and so on and so forth. No computer graphics application or tool is complete without handling pictures or digital images. Digital images or digital image processing is a subject by itself that forms in most graphical applications or tools an important part of the applications.

So one of the applications of computer graphics is also digital image processing where you should know how digital image pictures are stored, retrieved, how they are manipulated if necessary. And as I was saying you have to go through a very exhaustive full-fledged course on digital image processing. But from the point of view of computer graphics I will give brief highlights of a few aspects of digital image processing which is necessary from the computer graphics angle.

You will need to store pictures in certain formats, you need to retrieve them as well and well you also need to manipulate them, you need to alter certain pictures, of course you know to probably rotate pictures in 2D but we will talk about scenarios where the image is of degraded quality and you will see how in the time to come how to enhance the pictures, how to remove noise from the pictures. The image could be noisy due to various effects due to sampling itself or due to the electronic circuitries which create noise in the picture. There could be noise coming out when the image is transmitted over the net.

So we will see topics about image enhancement also. But first come first serve, we will see how the digital image is stored in a computer. Of course there are various formats, various file formats based on which a digital image is stored. And we will probably be able to discuss only one of them in some detail and I will probably name a few other formats which do exist.

Of course there are numerous proprietary digital image formats based on which the picture or a digital image is stored in a computer. But we will talk about a compression standard which is very commonly used in different websites, web pages, sites which hosts digital image galleries, it will typically store image in a certain formats which is easy to transmit without much of loss of information.

Without much of degradation and quality you need to store a picture in as small amount of spaces possible not only to save storage space but also you should need less time to transmit over the internet. These days we have websites which not only hosts different type of graphical pages and contents but also pictures and movies of course. But if you take just pictures by itself there are whole lot of image galleries and other websites which store pictures which you can retrieve. But when you need to retrieve the picture the contents of the graphics page takes a long time when you need to download a digital picture. If it is very large in size the amount of time required to transmit it over the net could be very large. So you need to compress it and store it.

So we will talk about the digital image compression format called the JPEG and these are the two aspects which we will discuss today about digital image processing. One is about compression and the second part will be on image enhancement which we heard, we talked about trying to remove the noise in the picture and improve the quality of a picture. So before going into JPEG let us also look at the different digital image file formats. I was just mentioning a little while earlier that there are various types of file formats which are used to store digital pictures.

(Refer Slide Time: 07:43)



So just to name a few of the most commonly used or a popular image file formats which are used by most graphical systems or even image processing toolkits or webs, web pages or web sites which host pictures are the following. Let us see a few of them. We see here a list of few commonly used and popular digital image file formats.

Well, the first of this is the JPEG format which we will discuss in detail today. It is developed by the joint photographic experts group. The society of that developed this digital image file format which will be based on discrete cosine transform which we will discuss in detail later on. Then of course the same group came up with a JPEG 2000 which is based on discrete wavelength transform.

Among the other popular file formats are the BMP which is known as the Bitmap file format. Then we have the TIFF, TIFF which is when you expand you will see that it is Tagged Image File Format. That is a very common format used for scanning documents the TIFF or the tagged image file format. Then you have a class of digital file formats called the PGM, PBM and PPM together. Almost quite a few things are common in terms of the file headers here.

The PGM is called the Portable Grey Map. PGM is Portable Grey Map and PBM is Portable Bitmap. I repeat Portable Bitmap is PBM and PPM is Portable Pixel Map. So these fall under one class of portable map file formats grey bit and the pixel respectively. Then at the bottom left of your screen you have the GIF which is the Graphic Interchange Format, I repeat Graphic Interchange Format. You also have the RAS which is the sun raster file format typically used for storing pictures in sun OS systems. It is called the sun raster image file format. Then you have the RAW file format which could be used in any system. In fact it is a file of the digital image which does not contain any overhead in terms of the pixel resolution, the number of bytes per pixel and the format etc.

The RAW will not contain any information about the format of the file as it is format free. Typically you may have one byte per pixel for monochrome or three bytes per pixel for color. Then of course you again have the SGI. The SGI format for storing graphic images in SGI which is in SGI silicon graphics proprietary format like we had the RAWs which was some raster preparatory format.

SGI is the SGI proprietary format. Then you have here XBM. Well XBM is a monochrome bitmap format in which the data is stored in c language data array. So monochrome bitmap is almost similar like a RAW where the data is stored as c language data array.

The CDR is again a proprietary format of coral draw. When you use the coral draw software for drawing images and pictures you basically save it in coral draw format basically in vectored format for storing pictures, drawings whatever you have. Then the last of this which you see on the right bottom of your screen is the PNG format or the Portable Network Graphics format. PNG is Portable Network Graphics format. These are a few examples of the commonly used popular digital file formats.

So let us go back and look at JPEG compression which we will discuss. We will discuss about the JPEG file format which is used because that is probably one of the most commonly used. And why it is probably most commonly used is that it can store pictures in a compressed format and when you have a huge database of image galleries to store this space becomes a criteria for you to store a large database of image galleries.

Then a compression standard using a mechanism to compress the image data and store it will be very useful and handy because you will require less space to store each image file. And you can store more images or more image files within a certain disc space, limited amount of disc space although whatever large it may be.

If each individual image files take some or require some small amount of disc space than the raw image format where each pixel requires one or three bytes to store. So let us look in to the image compression standard called the JPEG. We will see the expansion of the word JPEG very soon. So we will discuss today the image compression standard JPEG or JPEG as it is popularly known as

Before going into JPEG format we will understand what image compression is. The basic requirement of image compression is to reduce the amount of data required to represent a digital image. So we discussed about this that the amount of storage space in your secondary storage or even primary secondary storage mostly to store the image formats. If that can be minimized some how then you can accommodate more number of image files within a certain limited space.

(Refer Slide Time: 12:46)



So, the basic purpose of image compression is to reduce the amount of data required to represent a digital image. How to reduce? We will see soon that an image has lot of redundant data. So we will try to remove the redundancy in the data or remove the redundant data and store it in a compressed form. The question comes where the redundancy is and why at all do we have redundancy in the image format.

That is a big question but we will answer that with a few points in the next slide about where is this redundancy. That is the extra data which you do not need to probably store to display the image properly in some form without much of degradation in quality if there is a redundancy. That means some extra amount of bits or bytes are used to represent each individual pixels or a group of pixels let us say can we throw of a few bytes.

Can we throw off a few bits per pixel or for a group of pixels and that means the entire image for that matter and then use less amount of storage space to store the data. So where is the redundancy? Well, the first type of redundancy which exists is the coding redundancy. This concept is typically similar to data compression. Those who have background and data compression can easily follow that. Otherwise I believe that you have to read concepts of at least say the Huffman tree or the Huffman coding format to understand data compression.

A very typical question which could be asked to you or by any image processing expert is what is the difference between a data compression and an image compression? But there is another question between data processing and image processing but that is not a question which should be attended in this lecture. But at least since we are discussing compression as a part of computer graphics here we will see that whatever concepts of data compression are applicable for any data, we talk of any data I am talking of bytes of records of student roll numbers, CGPS, addresses, data in the form of text files, data in the form of executable files, various other types of files object files, adobe, PDF files etc we need certain number of bytes to store any type of file.

There is redundancy in any sort of the data and you can use a zip tool any sort of a zip tool to compress the data and that is done based on the concept of data compression or data redundancy. So similar concept is also of course definitely applicable in the case of a digital image where you have a coding redundancy as you can see here where we will say that the grey levels of the image uses more code symbols than what is really needed or what is actually needed to represent it.

That means the pixels of an image sometimes use more number of bits than what is actually necessary where if you say each pixel if you remember the display devices and the video RAM which we discussed and the refresh buffer we were talking of 3 bits, 8 bits or even 24 bits to represent a pixel. It is 8 bits for a grey level, 16 bits also possible or 24 bits for a full color, the question is do you need all those bits to represent a full image. If you take individual bytes out of those pixels it will be a combination of 0s and 1s. There will be a few 0s and few 1s depending upon the grey level value or the color value. And you can probably throw off a few of those very efficiently of course cannot throw it down in a very random manner. And use the non redundant bits or bytes to store that information. And that is done precisely the same way as is done in any coding concept.

You should be able to get this concept of Huffman coding based on any data compression course or even under algorithms course where Huffman tree construction is explained. **So I leave that as an exercise** that you assume for the time being. You know Huffman coding or you will go back and study it because that is an integrated part of JPEG compression as well. So, data compression is done by that particular concept of coding redundancy

As we move to the next part inter pixel redundancy which is an important part of image compression not data compression here because data may not have pixels but the image do have pixels. So this inter pixel redundancy results from structural or geometric relationship between objects in an image. That means if you look at a set of pixels,

neighborhood pixels in an image, let us say if you are looking at this picture towards me and you take a set of pixels around this particular corner which could be bright and then you can have greyer shade here.

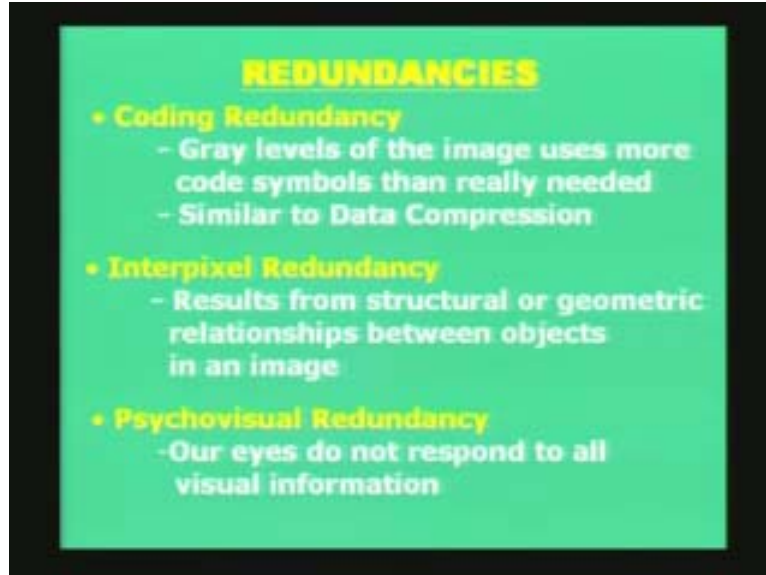
Let us say if you are looking into a set of pixels through this particular region you can take a small rectangular strip the pixels have the same value. All the pixels in the background let us say for the image we look at a static picture even for that you have the same value. And for most parts of the image we will have such continuous set of uniform grey level values, uniform color of pixel values. Then why do we store all those color information for all those pixels, we do not need. That means the concept is that between adjacent pixels in the neighborhood, if you take a group of pixels around in an image except at certain very few small parts of the image in general the adjacent pixels or nearby pixels have same or if not same very similar color or grey level shading. And this inter pixel difference is so small compared to the actual values that if you store the inter pixel difference which could be 0 or a very small number you will need few number of bits to store that than the actual value.

The actual value could be in the range of 100 to 200. But if you take adjacent pixels the gap could be as low as about 1, 2 or even 10 or 20 or even in some cases 0. And if you need to store values in the range of 1 to 10 or 20 the number of bits you know you can do with only four or five bits. Whereas to store values in the range of 100 or 200 you will need seven or eight bits. So there comes the inter pixel redundancy where you can throw of those extra bits somehow by storing only the differences and you require less number of bits to store the difference.

We will see that with an example later on where adjacent pixels have similar values and then you can throw of the redundant bits and store only the differences. What are those differences? They are not the redundancies but. The inter pixel redundancies in the value which is something like all the values are high. They are similar and high or low whatever the case may be you just store the differences and by storing the differences the redundancy is thrown out.

Therefore, results from structure or geometric relationships in objects in an image basically means that nearby pixels belong to the same structure or the same geometrical object and hence you will have the same pixel values and you need not store all those pixels explicitly an image to display it. That is where the inter pixel redundancy lies. The third redundancy comes from the fact that psychovisual redundancy. Psychovisual redundancy says that our eyes do not respond to all visual information.

(Refer Slide Time: 21:02)



Now this concept is related to the perception of the eye, visual perception of the eye. That means although typically our eye is probably the best visual sensor available in the world very naturally built by nature.

There could be other types of sensors which could be efficient in a certain environment or an application visual sensor but compared to the normal charge couple device cameras our eye impact can adjust to very large range of variations of intensity or color. That means with our eye we can see almost in a very dark room to a moderate dark to a moderately bright to a very brightly illuminated room or in a outdoor environment where we have bright sunshine. In a bright sunny day to a gloomy day our eye more or less reacts or works in a similar manner.

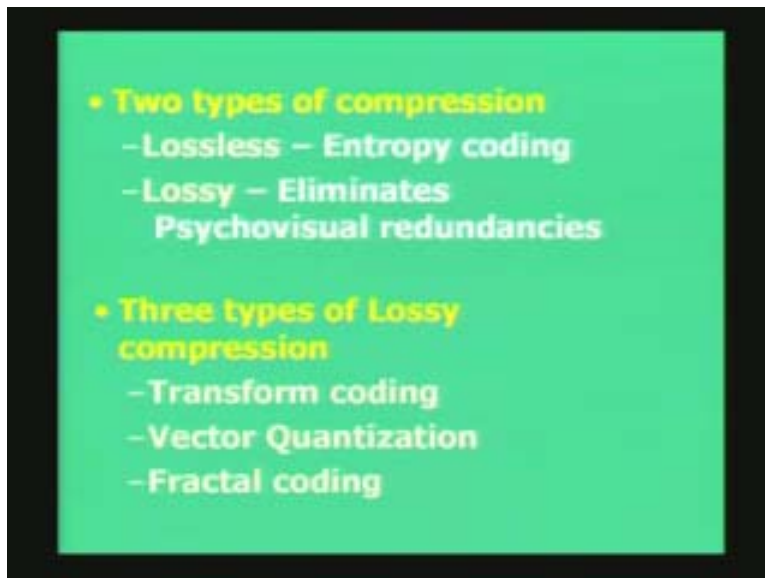
Why? It is because the eye has a set of sensors called the rods and cones in the retina which can definitely adjust itself to changes in brightness. When you move from a dark room to an outdoor bright sunny day in the outside environment or come from outside to inside of a room you will need a little bit of time but your eyes do adjust. So the eyes do adjust to a large scale of range of variations of light in db units if you say. It can adjust illumination variations. But on the other hand there it has one more drawback. It neglects small variations in intensities.

Perceptual psychologists or biologists or numerologists will be able to tell us that the eye is not very sensitive to small variations. That means you change the illumination little bit in the room by artificial mechanisms switching on switching off one light more than that changing the illumination by controlling the voltage of the lamps which are illuminating the room the eye does not respond to small very small variations. Well, if this variation is significant it does but otherwise it cannot.

In fact when you are seeing a picture and if there are small variations in grey levels of pixels the eye integrates, the eye sees an integral effect of that and almost throws of those small variations. Sometimes it is not able to see. Of course even if it sees the brain will also discard it. That means in an image we do not need to show or store small variations in grey levels of pixels to display the information content of the image. That is the psychovisual redundancy where small variations in grey levels also can be thrown off and we can only maintain large variations in grey level values.

What I mean by this is, if you talking of an eight bit pixel format to store eight bits or one byte per pixel to store a digital image that means values from 0 to 255. You can probably visualize changes of values from 50 to 100 to 150 to 200 and so on. But small changes say 150 to 155 or even 160 the eye cannot locate such a small difference. So eyes do not respond to all of the visual information present in the image in terms of small changes in color or grey level and hence those also could be eliminated. That is why you have three types of redundancies now and we will see how it is handled by the digital image compression format called the JPEG which is one of the standards of compression.

(Refer Slide Time: 26:21)

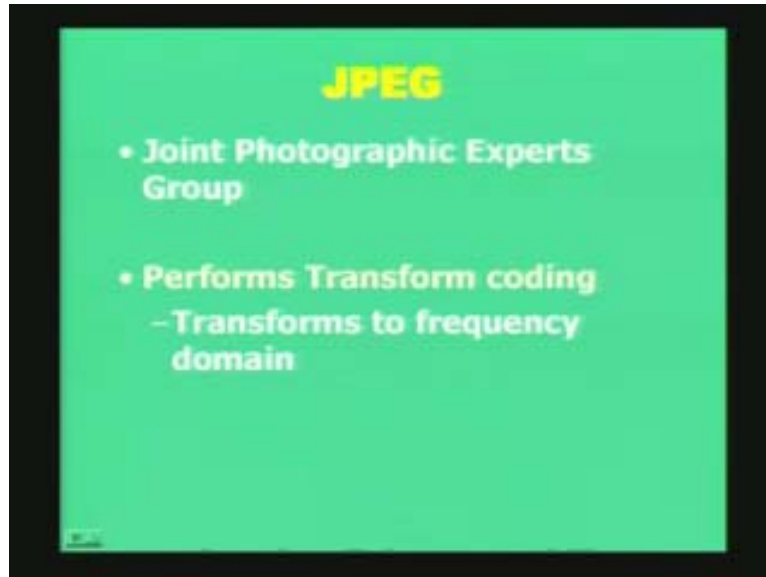


There are two types of compressions; one is lossless; another is lossy, entropy coding. An example of entropy coding which I gave you is the Huffman coding, which is lossless for data compression and we can have lossy compression which eliminates psychovisual redundancies because if you throw of small variations in grey levels you cannot recover them back.

The difference between lossy and lossless is, if you store an image in a lossless compression format then you will not able to recover the entire image under any circumstances some information will already be lost. But the image will apparently look very similar to the original image. We will see that with an example as how the overall image property does not change small variations. You may be able to locate but not

much. But in the case of lossless compression you will be able to recover the whole image. In the case of lossy compression you will not be able to recover the entire image.

(Refer Slide Time: 26:55)

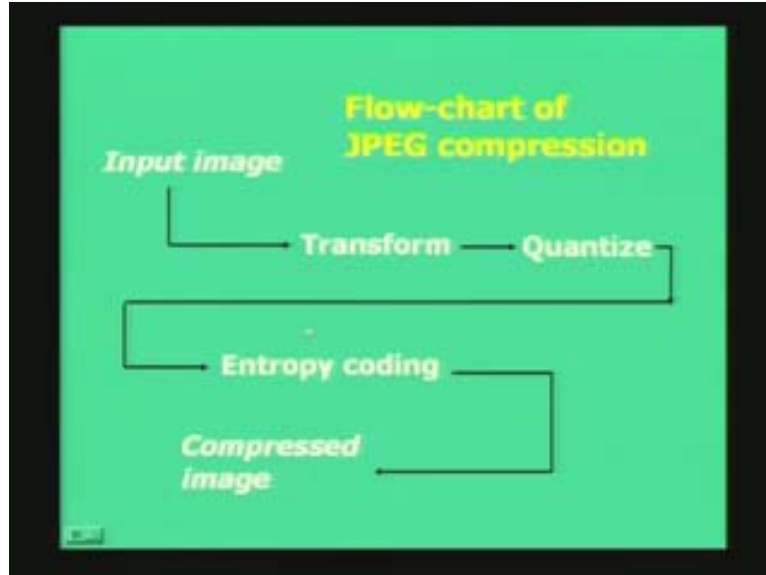


So these are the two types of compressions which are available. There are three types of lossy compression; transform coding, vector quantization and fractal coding. And we will see that our concept of JPEG compression deals with transform coding mostly and we also have a little bit of quantization step available. This is the expansion of JPEG. JPEG or JPEG as it is called, it is a joint photographic experts group that is the name which develops the standard for compression of an image and basically performs transform coding that means it transforms an image to a frequency domain.

Those who have some idea about signal processing or image processing little bit will be able to follow what I mean by transforming an image or pixel data to a frequency domain those who have not can just go through the equations and then later on go back and look into a book on signal or image processing and look into concepts such as Fourier Transform or Discrete Cosine Transform.

Discrete Fourier transform or Discrete Cosine Transform is typically used for transform coding in JPEG compression. I will show you the equation and try to describe what it means within the limited time available to me. But we have to go back and look into concepts of Discrete Fourier Transform or Discrete Cosine Transform to have a good idea. So let us look at the flowchart of JPEG compression. The flowchart of JPEG compression says that you have an input image here which will be transformed by a Discrete Cosine Transform.

(Refer Slide Time: 27:52)

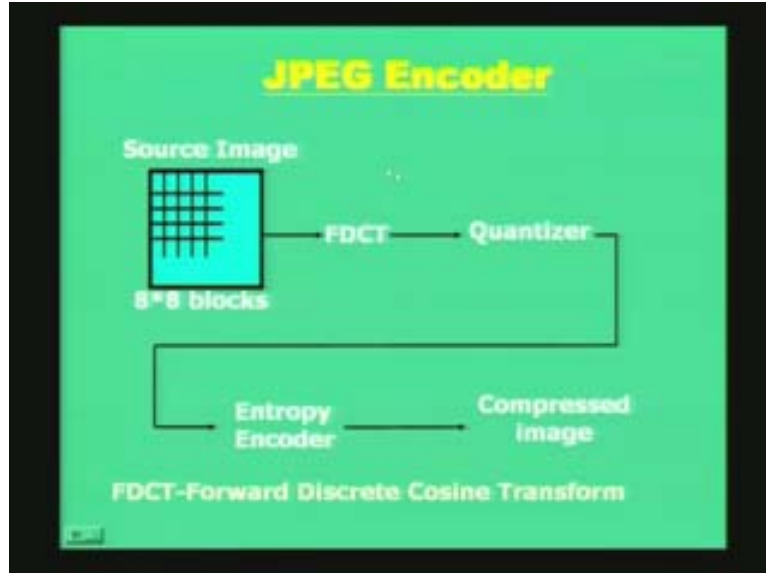


That is the transform coding and then it will be quantized. So we do have a quantization table which will do that for you and then of course it passes through an entropy coding. **I will not have much time to discuss** but I will say that this is basically based on Huffman coding. And then of course that gives you the compressed image. This entropy coding does the lossless data compression and transform coding does the psychovisual and inter pixel redundancies are thrown off in the transform and the quantization steps and the entropy coding is lossless where it does the typical data compression. There are basically broadly about three steps in the JPEG compression transform, quantize and entropy coding.

Let us look at the transform. That is a JPEG encoder where an image is passed through a FDCT. FDCT means Forward Discrete Cosine Transform. I repeat Forward Discrete Cosine Transform. So a source image is split up into 8 into 8 blocks. So it is divided into sub images or windows. Each sub image or window is of size 8 into 8. That means if you have an image 80 into 80 then how many blocks will you have if each block is size 8 by 8? You will have 100 blocks.

How? We have ten rows and ten columns of blocks. So in array of blocks each and that array of blocks will have ten rows and ten columns because 80 by 8 so 10. If the image size is 256 into 256 can you guess what will be the array of blocks? How many rows and columns you will have of the array of blocks? It is very simple, 256 by 8. So you will have 32 into 32 array of blocks. So each of these blocks after the source image is split into 8 into 8 blocks.

(Refer Slide Time: 30:05)



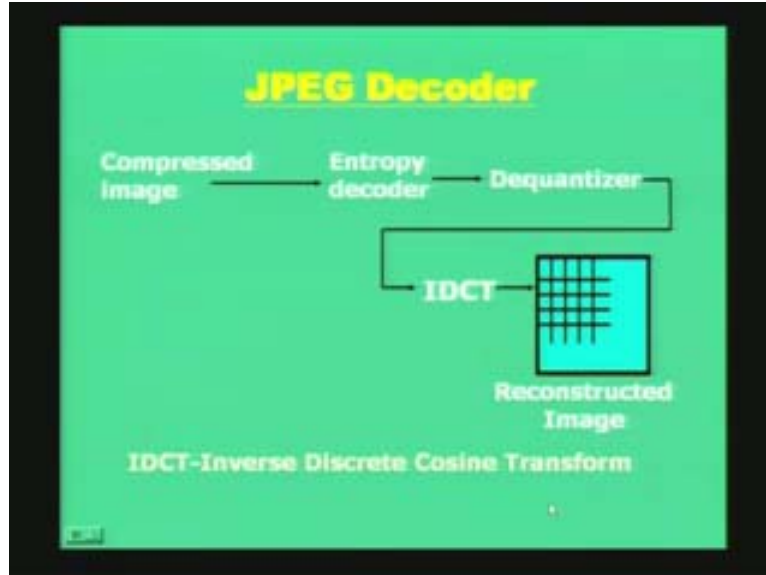
Each of these blocks separately pass through a Forward Discrete Cosine Transform. We will see the equation here then it passes through the quantizer and of course entropy encoder and get compressed image. As I was saying FDCT is the Forward Discrete Cosine Transform.

Let us look forward, JPEG decoder we had the encoder which produced you the compressed image from the source image or the input image. That was the job of the encoder which will give you the compressed image much much less in size and compact compared to the number of bytes required to store the original source image or input image. Compressed image requires less size.

What is the job of the decoder? Just the reverse, take the compressed image and then reconstruct back the original or input image. That is what the decoder does here, it takes a compressed image as you can see here. It takes the compressed image there is entropy decoder as we had an entropy encoder it is the reverse task of the Huffman encoder.

We have a Huffman decoder here, the quantizer instead of a FDCT which you had for an encoder the decoder will have an Inverse Discrete Cosine Transform IDCT is Inverse Discrete Cosine Transform. This is done for each block separately and the blocks will be stacked up in an array and that is what helps you to provide the reconstructed image.

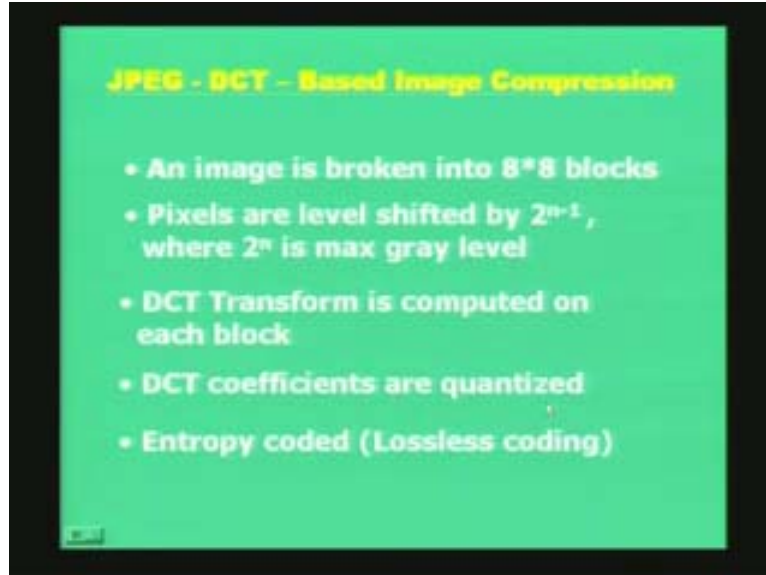
(Refer Slide Time: 30:21)



So the role of the encoder is just the reverse of the decoder or the job of the decoder is the reverse of the encoder system. Let us get into the concept of what happens in the transform coding in frequency domain. So if you look back the JPEG Discrete Cosine Transform based image compression you know that the image is broken into 8 into 8 blocks and then pixels are level shifted by 2 to the power of n minus 1.

So if there are smaller number of bits to store the pixel values or represent the pixel value then the maximum number of the value will be 2 to the power n minus 1 will be the maximum grey level value or 2 to the power n values will be available so half of that 2 to the power n minus 1. So if n is 8 it will be divided by 2 to the power 7 that is 2 to the power 8 minus 1, 2 to the power 7 which is 128, it will be shifted by 128 that is 256 by 2. So pixels are level shifted by 2 the power n minus 1 and then DCT transform is computed in each block.

(Refer Slide Time: 33:14)



We will see the equation now and then the DCT coefficients which are obtained by the DCT transform are then quantized and then they are entropy encoded by the Huffman tree and that produce that part is lossless. The lossy part comes here in terms of the DCT transform and the quantized coefficients. This is an example of Discrete Cosine Transform. Those who are familiar with Discrete Fourier Transform will understand that this expression looks very similar to that $f(x, y)$ is an input two dimensional array or a function T u v are the transform coefficients.

u and v represent the variables for the frequency. In analog case the integral becomes sigma, in a discrete case 0 to n minus 1 the size here n is 8 so it will run from 0 to 7 both x and y and g x u v is basically the basis in the Fourier Transform it is an exponential imaginary part and in the case of Discrete Cosine Transform the expression of g x y u v is given by this particular expression.

Product of two cosines one along u one along v and α u and α v are the normalized coefficients as given. So that is all you can just note down this expression here and these are just the normalizing factors. Inverse Cosine Transform will also have a similar expression only you will have a negative sign. That is what is called the Discrete Cosine Transform.

(Refer Slide Time: 33:49)

Discrete Cosine Transform

Forward Discrete Transform

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

$f(x, y)$ represents pixel value

where

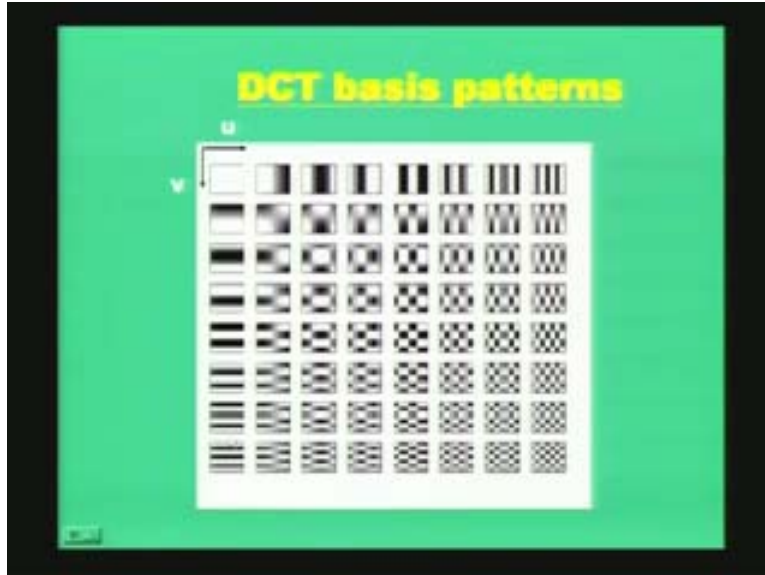
$$g(x, y, u, v) = \alpha(u)\alpha(v) \cos\left[\frac{(2x+1)u\pi}{2N}\right] \cos\left[\frac{(2y+1)v\pi}{2N}\right]$$
$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{N}} & \text{for } u = 0 \\ \sqrt{\frac{2}{N}} & \text{for } u = 1, 2, \dots, N-1 \end{cases}$$

So, given an $f(x, y)$ the $g(x, y, u, v)$ is given as given here, $g(x, y, u, v)$ is given here and given a certain $f(x, y)$ of a certain dimension or size on 0 to $N-1$ you will get the corresponding $T(u, v)$ which are the Discrete Cosine Transform or in short henceforth we will say DCT coefficients that is what is typically used. Generally DCT coefficients are the Discrete Cosine Transform coefficients $T(u, v)$.

These are the basis patterns, if you look at eight different basis you can see there are horizontal rows of eight basis and vertical columns of eight blocks and each represents a certain $g(x, y, u, v)$, $g(x, y, u, v)$ as given by this expression here. This particular expression let us forget the alpha part which is just a normalizing factor. It is basically a cosine along horizontal and vertical directions given by x and y . u and v are the frequencies along x and y respectively. As you can see here if you move along the horizontal directions you will see increasing frequencies of a cosine. It is like a sinusoidal curve shown in terms of frequency.

Thus you can see as you move to the right or from top to bottom u increasing from left to right v increasing from top to bottom the frequency of the pattern is increasing.

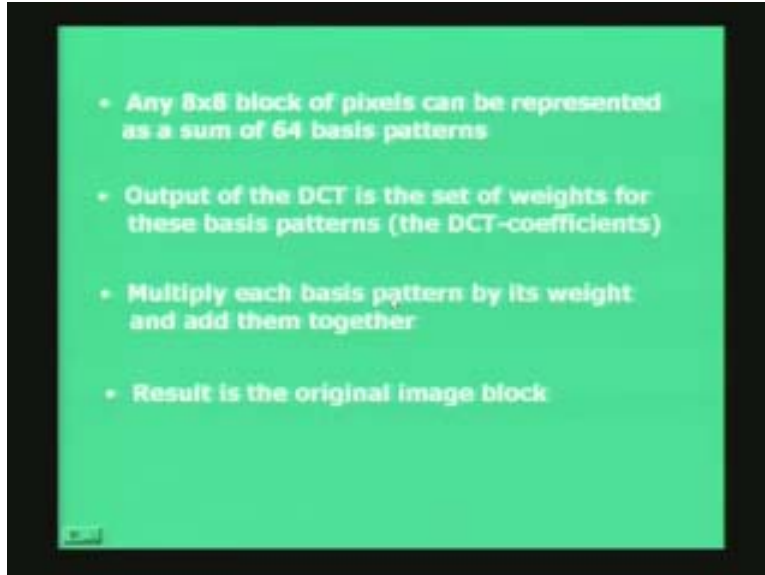
(Refer Slide Time: 34:50)



So these are the DCT basis patterns and any 8 into 8 block of pixels which are obtained from the images can be represented in fact are represented as a sum of these 64 basis patterns. So you have 64 pixels in a block which is represented as a sum of weights of this basis patterns. Fourier Transform also does in terms of sinusoidal frequency cosine and sine e to the power g ω this is in terms of the cosine function so that is the difference. That is what is done here and then output of the DCT is the set of weights for these basis patterns called the DCT coefficients.

The DCT coefficients are nothing but the set weights obtained by Discrete Cosine Transform and multiply each basis pattern by its weight and add them together. If you do that you will be able to get the original image block from the image block based on which you got the DCT coefficients, **mind you.**

(Refer Slide Time: 37:01)



So you got this DCT coefficients from the image block but if you use the set of weights or DCT coefficients and multiply the basis pattern with these weights which you got or the DCT coefficients which you got multiply the basis pattern and you should get the original image block back.

So, that is part of the reconstruction process the inverse DCT which we will see, that is given DCT coefficients given in the basis pattern if you go back to the equations it means basically you are given $t u v$ and the $g x u v$ where $g x u v$ is the functional basis and $t u v$ are the DCT coefficients given those two you can reconstruct $f x y$ also based on certain orthogonal relationship. We will skip over those mathematics with the time constraint in mind because this is not a full-fledged compression lecture on image compression but just giving highlights about how to compress the data.

Let us take an example of compression. Just look at the sample 8 into 8 block the image could be any size but I have just taken an arbitrary 8 into 8 block from the image and these are the pixel values. As you can see here the values range the highest from 154 down to a lower level of about 52 that seems to be the lowest possible value and this seems to be the highest possible value.

(Refer Slide Time: 37:49)

Sample 8*8 block

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	114	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

So you can see yourself that you need eight bits to store these pixel values. They are first level shifted by 2 to the power n minus 1 so minus 128 subtract or add minus 128 or subtract 128 from the previous sample which is this sample 8 into 8 block minus 1.

(Refer Slide Time: 38:30)

Level Shifted by -128

-76	-73	-67	-62	-58	-67	-64	-55
-65	-69	-62	-38	-19	-43	-59	-56
-66	-69	-60	-15	16	-24	-62	-55
-65	-70	-57	-6	26	-22	-58	-59
-61	-67	-60	-24	-2	-40	-60	-58
-49	-63	-68	-58	-51	-65	-70	-53
-43	-57	-64	-69	-73	-67	-63	-45
-41	-49	-59	-60	-63	-52	-50	-34

So let us take 52 minus 128 what will you get? You will get minus 76. Let us take another value the maximum value 154 minus 128 what will you get? It is 26.

(Refer Slide Time: 39:08)



The slide displays a grid of DCT coefficients. The top-left cell, containing the value -415, is highlighted in red and labeled as the DC coefficient. The values decrease in magnitude as they move away from the top-left corner.

DC co-efficient	-29	-62	25	55	-20	-1	3
7	-21	-62	9	11	-7	-6	6
-46	8	77	-25	-30	10	7	-5
-50	13	35	-15	-9	6	0	3
11	-8	-13	-2	-1	1	-4	1
-10	1	3	-3	-1	0	2	-1
-4	-1	2	-1	2	-3	1	-2
-1	-1	-1	-2	-1	-1	0	-1

This is how the level shifting is done. Each individual pixel is subtracted with the value 128 and on this what we do is we take the DCT coefficients. After transform we get the set of coefficients. As you can see here this is called the DC coefficient. That means it resembles some sort of an average energy of the 8 into 8 block and you can see these coefficients are those weights. And as you move towards the right or down and more diagonally across you find lesser and lesser weight values so that is the interesting step.

If you look back again I mention here this is the DC coefficient here and move to the right move to the bottom or move diagonally across the 8 into 8 you will find that the numerical value of the DCT coefficients keep going down. That means most of the information is in here. Actually if you suppress some of these values you are not losing much of information in the image because there is where we will say the redundancy in the inter pixel or psychovisual part lies in the high frequency contents they are important in some cases. But if you are not interested in the actual values of the pixels for certain applications for certain image processing applications you will need the exact pixel values in which you may not be able to throw out the redundancies.


But in terms of simple viewing transmission, fast transmission and viewing when you can afford to lose data or have lossy representation or lossy storage you do not require lossless that is what I mean. Then you can throw of this redundant information not required before the coding starts in terms of the Huffman coding or entropy coding. The inter pixel redundancies, the pshycovisual redundancies information that can be thrown out of the image and that is done by, we will see how, if you look at this coefficient the small coefficient values contribute too less to the overall information of the image.

The DC coefficient at all values nearby and around here is what the important information is. The information here is very minimal and of course you can see already some of the coefficients are started to become 0.

(Refer Slide Time: 41:00)

Quantization table

Divided
by

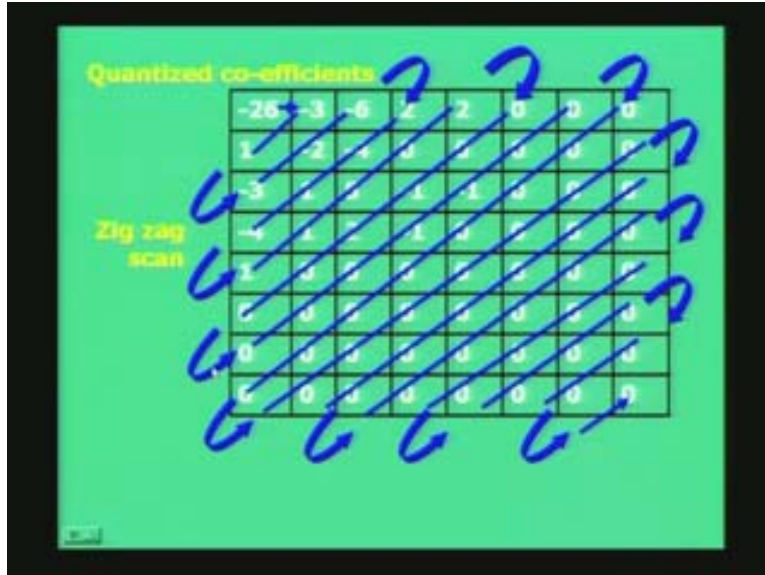


16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

So how do you do this? You use a quantization table which is also suggested by JPEG and you say that you basically divide the Dc coefficients by the quantization table.

Let us go back, this was the DCT coefficients, so pixel by pixel here we should not use the word pixel so the coefficients each array say if this is a $i j$ index arbitrary index if you take a $i j$ say 4 3 or 3 4 this particular DCT coefficient will be divided by corresponding value in the quantization table. So take this mask of the quantization table put it on top of these DCT coefficients and divide and you will get a resultant information like this.

(Refer Slide Time: 41:48)



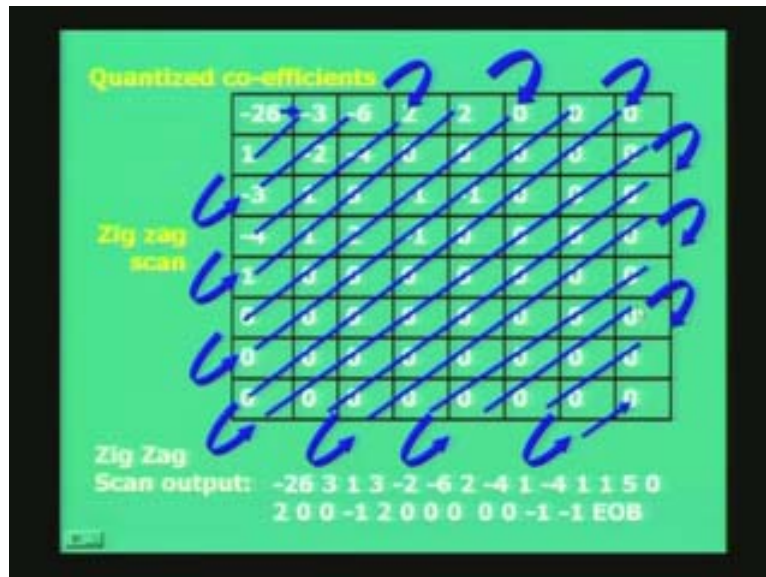
The quantize coefficients which you will have will have these values. And it is scanned in a zig zag pattern. This is an important step in JPEG compression these are scanned in what is called as a zig zag pattern.

Do a zig zag scan if you start from this and try to move in this fashion as pointed out by this blue arrow here then you will find you will get the zig zag scan order in which the coefficients automatically will be sorted in more or less descending order.

Let us look at the zig zag scan order. You see that the first value will be minus 26, minus 3, 1, minus 3, minus 2, minus 6 then 2, minus 4, 1, minus 4, 1, 1, 5 and so on. Let us look at the first few values 26, minus 3, 1, minus 3, minus 2, minus 6 so first value will be minus 26. Look at the zig zag scan output. I said earlier minus 26 3 and 1 minus 26 minus 3 there should be a minus here, minus 26, minus 3, 1, minus 3 again minus 2, minus 6, 2, minus 4, 1, minus 4 and so on. So this should be minus 3 and this should be minus 3 so that is the scan order. And of course you do not need to go all the way around up to all the 64 values to 0 because you see all the values are already 0 here.

How come all these values are 0? It is because the quantization table which had the values to be divided, the values in the DCT coefficients are divided by the quantization table they are adjusted in such a manner that unless a significant DCT coefficient of a higher frequency is prominently available that value will automatically become 0.

(Refer Slide Time: 43:08)



So you do not need to store all the 0 values. In fact you see that the last non zero values we stored here minus 1 and then you can put the end of buffer. It means that beyond these all the values here will be 0 EOB all the values here will be 0.

(Refer Slide Time: 44:28)

- Zig Zag scan is done so that the coefficients are in order of increasing frequency.
- The higher frequency coefficients are more likely to be 0 after quantization.
- This improves the compression of run-length encoding.

So you can see that the sequences are already sorted from about 8 into 8 equals 64 values. You will probably have about twenty to twenty five values at the most thirty values let us say in this case. You can count it yourself and find out how many values you have to store. And this zig zag scan output is taken to the entropy encoder.

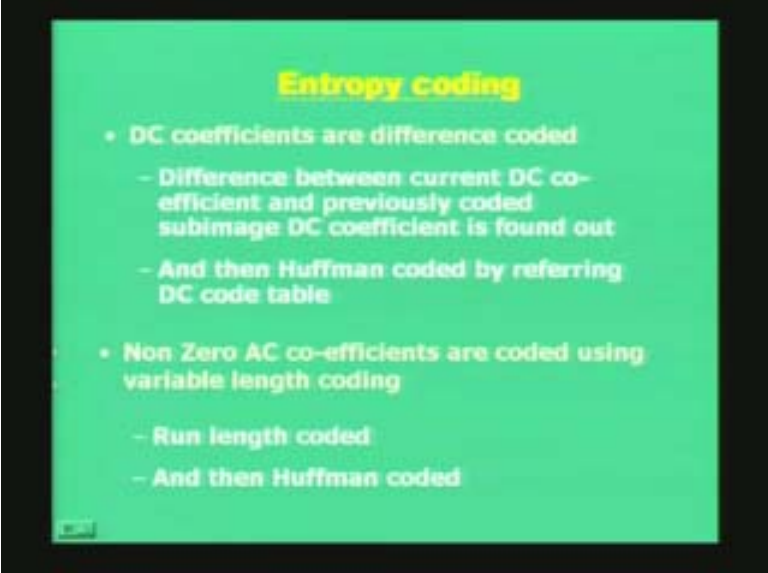
The zig zag scan is done so that the coefficients are in order of increasing frequency then the highest frequency coefficients are more likely to be 0 after quantization as we have seen that already. Then this improves the compression of run length encoding. It means when you have more 0s towards the end and you do not need to even store them then the run length coding also could be used when if you need to store all of them then of course you can use run length coding otherwise do not use it.

Entropy coding, in entropy coding the DC coefficients are difference coded. This means the difference between the current DC coefficient block and the previously coded sub image DC coefficient is found.

We said that the image is divided into small 8 into 8 blocks. So you do not need to store the first DC coefficient quantized value itself. You can look at the difference between the first DC coefficient or DC coefficient of the first block and the DC coefficient of the second block and the DC coefficient of the third block and so on and there also you provide inter pixel redundancy. That means you throw off the actual value and store only the difference. That means if you have the first value and the difference they are on then you can reconstruct the actual values also. So the DC coefficients are difference coded. That means again I repeat difference between the current DC coefficient and the previously coded sub image DC coefficient is found out and then Huffman coded by referring the DC code table. Huffman coding can be used here which is the entropy coding.

The non zero AC coefficients because there could be some 0 AC coefficients the non zero AC coefficients are coded using what is called a variable length coding. First they are run length coded and then they are Huffman coded. So this is the part of the entropy coding and similarly you can have an entropy decoding in order to the time limit available I may not be able to cover the great details about this but you will be able to learn yourself if you do a course on coding theory, data compression itself or based on simple data structures and algorithms where Huffman tree is covered where the Huffman coding and decoding also will be covered.

(Refer Slide Time: 45:48)




Entropy coding

- DC coefficients are difference coded
 - Difference between current DC coefficient and previously coded subimage DC coefficient is found out
 - And then Huffman coded by referring DC code table
- Non Zero AC co-efficients are coded using variable length coding
 - Run length coded
 - And then Huffman coded

So assuming that you know or you will come to know about the entropy coding if you have done entropy coded in a certain sequence in this case the sequence is disquantize zig zag sequence. I repeat; not disi but quantize sequence quantize zig zag scan sequence that is entropy coded. You can use a decoder to recover the same coefficients in the same zig zag scan pattern and the data coded is entropy decoder in the decoder.

(Refer Slide Time: 46:49)



The coded data is entropy decoded in the decoder

-26	-3	-6	2	2	0	0	0
1	-2	-4	0	0	0	0	0
-3	1	5	-1	-1	0	0	0
-4	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

So we look at the reconstruction back and this is what you will get back. This is an example of what you will get back as you can see there are probably more 0s than what we saw. The first value was still minus 26, minus 3, 1 and so on but there are lot more 0s

because that has come due to the coding and the decoding part. Some values have been thrown of already and then they are multiplied by the same quantization table, this is the quantization table which we used to quantize the original disi coefficient so you must multiply it back to give the denormalized coefficients.

(Refer Slide Time: 47:16)

-416	-33	-60	32	48	0	0	0
12	-24	-56	0	0	0	0	0
-42	13	80	-24	-40	0	0	0
-56	17	44	-29	0	0	0	0
18	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Of course we must keep in mind that the DC coefficient you might have to take the difference from the previous block to add this to get the DC coefficient back but if it is the first block then of course you have to get the actual one. Otherwise for each successive blocks you only store the difference and the difference is coded so you take the difference and look into the previous disi coefficients and then add it. That is what you need to do for successive blocks. So you get the denormalized coefficients either first block or any block and you can see here lots of 0s.

This was not present earlier. We had some zeros but not so many in the coefficients DC coefficients derived after from the 8 into 8 block of pixels. It went through quantization step, it went through entropy coding that part was of course lossless on length and of course there we threw of only the 0 coefficients we were not coding anyway but non zero AC coefficients were coded and that was the quantization has thrown off certain redundancy information and that is like chopping of high frequency or small variations in the image.

So this is the denormalized coefficients, what has to be done now with the denormalized coefficients? You need to apply an inverse DCT inverse Discrete Cosine Transform will give you these values from the **previous.....** this was the denormalized coefficients, apply inverse DCT, get this, this was already simulated and given to you, you do not have to copy these values but just to know after inverse DCT you need to level shift the values. Remember, you had shifted the values by subtracting $128 \cdot 2^{n-1}$.

(Refer Slide Time: 48:47)

After Inverse DCT

-70	-64	-61	-64	-69	-66	-58	-50
-72	-73	-61	-39	-30	-40	-54	-59
-68	-78	-58	-9	13	-12	-48	-64
-59	-77	-57	0	22	-13	-51	-60
-54	-75	-64	-23	-13	-44	-63	-56
-52	-71	-72	-54	-54	-71	-71	-54
-45	-59	-70	-68	-67	-67	-61	-50
-35	-47	-61	-66	-60	-48	-44	-44

So you have to add that value here. So after inverse DCT if you look back to this table you need to add 128 which will give you the reconstructed sub image after level shifting. This is what you get back after JPEG compression of the 8 into 8 block. All 8 into 8 blocks will go through this sequence. Now, if you remember those values I will show them to you in the next slide that the values have changed. This is what we get after encoding and decoding. That is why I was saying this is the reconstructed sub image of 8 into 8 the original values of the image were something like this.

(Refer Slide Time: 49:16)

**After level shifting
(reconstructed subimage)**

58	64	67	59	62	68	70	78
56	55	67	89	98	88	74	69
60	50	70	119	141	116	80	64
69	51	71	128	149	115	77	68
74	53	64	105	115	84	65	72
76	57	56	74	75	57	57	74
83	69	59	60	61	61	67	78
93	81	67	62	69	80	84	84

This was the original sub image.

(Refer Slide Time: 49:50)

52	55	61	66	70	61	64	73
63	59	66	90	109	85	69	72
62	59	68	113	114	104	66	73
63	58	71	122	154	106	70	69
67	61	68	104	126	88	68	70
79	65	60	70	77	68	58	75
85	71	64	59	55	61	65	83
87	79	69	68	65	76	78	94

Let us look at a few values, the first row 52, 55, 61. What have these values become? It has become 58, 64, 67. Let us look at the column 52, 63, 62, 63 has 58, 55, 69. This is the result of coding and decoding from the starting point which was 63, 62. Let us look at the maximum value 154. What has this value become now after reconstruction? You see it is still the largest one but it is 149. That means at each point there is an error which has script in. Now strictly it is not an error because that is what we want to do. We wanted to throw some redundant information and store only those information which is essential and that is what it is. And so each pixel will have certain variations but it does not cause much in terms of when you look at the picture. We will see with an example with a picture also but let us look at the difference between the original and reconstructed sub image pixel wise.

(Refer Slide Time: 50:51)

Difference between original and reconstructed subimage

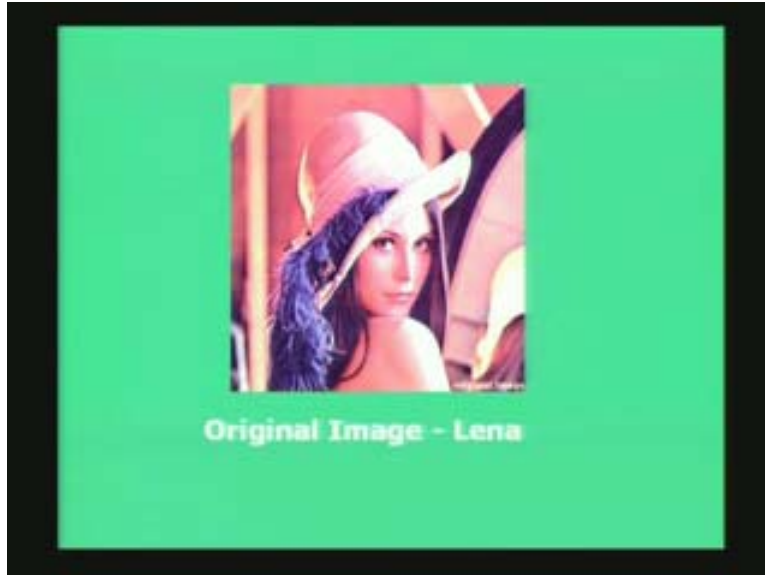
-6	-9	-6	2	11	-1	-6	-5
7	4	-1	1	11	-3	-5	3
2	9	-2	-6	-3	-12	-14	9
-6	7	0	-4	-5	-9	-7	1
-7	8	4	-1	11	4	3	-2
3	8	4	-4	2	11	1	1
2	2	5	-1	-6	0	-2	5
-6	-2	2	6	-4	-4	-6	10

In some cases of course you do not have a difference but the difference is 0. But you can have a maximum difference of as large as about 12 or even 14 that is the maximum error which you will have. This pixel has the maximum error.

What is this? Let us look at this value. This value was 66, it was actually 80. We got reconstructed value 80 but the original value was 66. That is why you got the maximum value of 14. Now if you have followed this pixel by pixel 8 into 8 equals 64 values you have subtracted and this is the error which you have got. So let us look at the image as if this is done in an overall image what happens? Does the image look really bad if you have these errors after reconstructing back.

Once it is reconstructed back and is stored in this JPEG format you will always reconstruct back the same image which you will be seeing now. Once the redundant function is thrown out there will not be any further loss unless you force some more redundant information to be thrown out because in JPEG 2000 in JPEG you can actually select the quality of the picture and highest quality will need larger space and lesser quality will require lesser space. So let us look at an example. That is the standard example which is used in image compression literature by all researchers world wide the classical image of the Lena. Of course it is a colored image do not worry about the color factor.

(Refer Slide Time: 52:27)



So if this is the original image and if you carefully look at it this was the original image. I will show an example of an image compression using JPEG where the compression ratio is 4.2:1. That means I am able to use JPEG compression and compress it to almost 1 by 4th the size, 4.2:1 means the ratio of the uncompressed image divided by the compressed image that is the ratio, the compression ratio we talked about 4.2:1. The ratio of the bits or bytes required to store the image in compressed format and uncompressed format. So, if the original image was something as seen here as you can see even with the compression ratio of 4.2:1 that means you almost require about one fourth this size to save this with respect to the original image. You will barely see any difference between this image and the original image.

Do not worry about the size of the enlargement in the image but it is only a matter of zooming here but you will not be able to see the difference. There is some differences but not. But let us look at a larger compression ratio. Now if you look here I have increased the compression ratio 7.3:1. That means the image requires about one seventh the size to store.

(Refer Slide Time: 54:03)



Earlier it was one fourth and it is one seventh and if you look back into the picture the left hand side image which had the compression ratio of 4.2:1 was almost close to the original image but there are lots of facets here. If you carefully come and have a look the background information wherever there are regions of continuous information either in the background or in the hat or on her face or on the shoulder or even here there are patches of what is called as the blocking effect blocking artifacts in JPEG.

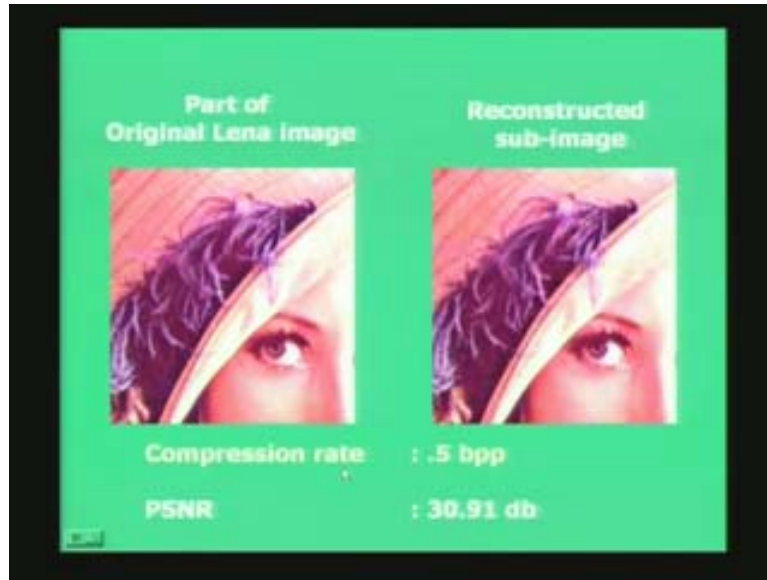
You can see it is clear here you might have to probably change the brightness or contrast of the image you have a closer look. You can see certain patches here certain patch here certain patches here as well and there are the effect of course similar to aliasing of a line but it is on grey shades and this is what is called the blocking artifacts. It is marginally present here also but not really visible. This is so negligible that is close to the original image but it is apparent here mostly as well as in some parts in this location.

I think this part is very clear. And of course again in some parts in the bottom of the shoulder here and of course on her face and on the hat as well on the background portions here as well as here in this part where there are significantly strong blocking artifacts of JPEG. This blocking comes mainly due to this 8 into 8 blocks which I talked about but that is one of the reasons. The other reason is some information is thrown out. As you can see if I had not pointed you out these differences of the blocking artifacts these two images would have been almost visually similar.

If you look back at the original image and then I show you from this from a distance and even seen this all the images look apparently similar. You will be able to identify that this is the image of the Lena. So in terms of psychovisual redundancies thrown out the right hand side image requires one seventh or lesser space than the original image. Hence it is better to save this image if you want to only show the image. Not to do some processing

image requires original grey level pixel values which is very rarely required. So that is the purpose of image compression.

(Refer Slide Time: 56:04)



Let us look at the couple of last examples here. This is also part of the Lena image where the compression rate, there are various ways by which this compression metrics which are used to define compression. In the previous one we used a compression ratio then we have the compression rate and the PSNR Peak Sinaled Noise Ratio. These are signal processing terms.

How many bits are required per pixel? That means in this case I require half bit per pixel with respect to one bit per pixel here. That means it requires half the storage space of this. And you can see some small artifacts coming out in the reconstructed image not much visible. This requires half the storage space and PSNR is about 30.91 db.

Look at this I increased the compression, 0.22 bits per pixel original image here part of the mandrill image which is also very commonly used, these images are very popular because they have areas which have both texture and flat region so that is why they are used. You can see the very strong blocking effect here all around almost.

(Refer Slide Time: 56:47)

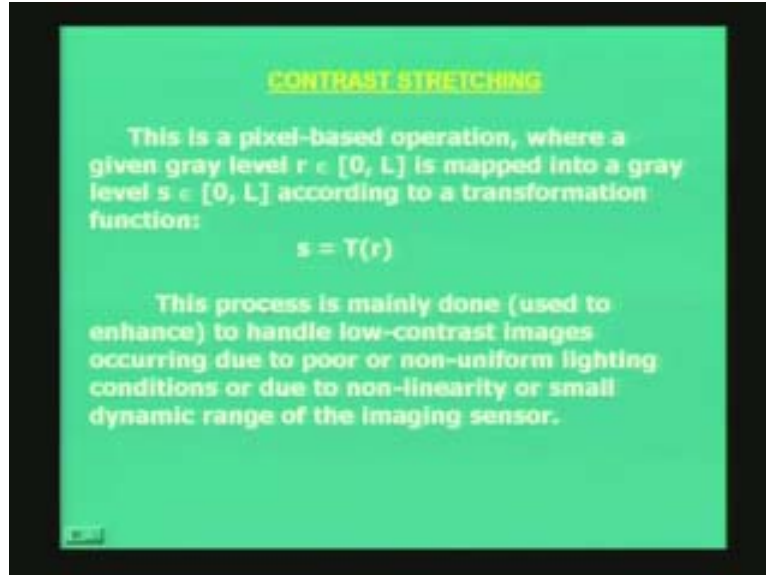


Everywhere you can see the blocking effect which is not available here, it is very small. Probably you have to come close to the screen to have a look at it. You can see these regions compare it with the blocking artifacts here you can see effects of blocking here as well which is very small. Some blocking effects here as well as well here, all over the blocking artifacts because the compression ratio is so large, 1:36 0.22 bits per pixel PSNR about 20 db.

These are the examples of JPEG image compression. In the time remaining we will start the next section and then move on to the next lecture. Well I will introduce the concept of other part of digital image processing which is also important from computer graphics people is to manipulate image so that you can improve the quality which is called digital image enhancement. So look into the slide, we will introduce the concept of digital image enhancement and stop where we are talking of what is called a contrast stretching. this is a pixel based operation where the image intensity or the grey level are lying within the range 0 to 1 is mapped in to a grey level s 0 to 1 according to a transformation function. So r is the input grey level s is the output and you require a transformation function t . This process is mainly done or it is used to enhance the images where you need to handle low contrast images occurring due to poor or non uniform lighting conditions or due to non linearity or small dynamic range of the imaging sensor.

In many cases you require enhancement or contrast stretching where the image could be degraded to low lighting conditions there could be noise in the image. We will talk of noise removal later on but the image could be degraded due to long storage, transmissions, it could be noisy, it could be **the minuet as acquired** the sensor was not working well, the lighting environment is very poor or may be some sort of camouflage or smoke where the picture quality is not **good**. So you need to probably do some stretching of the contrast. But you typically do these in TVs and you all observed any show, entertainment or education or whatever is the purpose.

(Refer Slide Time: 59:39)



We will see here in the next class examples of contrast stretching and enhancement based on pixel values but just to go back the basic example any enhancement based operation which does contrast stretching transforms a pixel value r of the original image to a new pixel value s lying between range of grey level value 0 to 1 and this process is mainly done to handle low contrast images occurring due to poor or non-uniform lighting conditions or due to nonlinearity or small dynamic range of the imaging sensor.

So these are the reasons why we need to enhance an image, improve the quality of the image and that is done by various techniques. We will discuss in the time remaining to us a few mechanisms of how to handle the image quality, how to improve the quality of the image if it is degraded due to lighting conditions, sensor conditions, sensor parameters or even due to degraded due to noise. We will discuss these aspects in the next class when we meet, thank you very much.