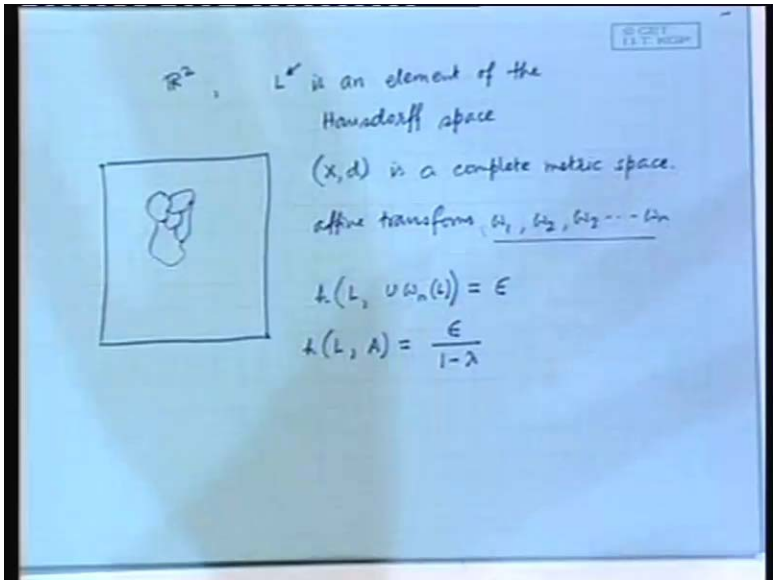
Chaos Fractals and Dynamical System
Prof. S. Banerjee
Department of Electrical Engineering
Indian Institute of Technology, Kharagpur
Lecture No # 19
Fractal Image Compression

We have seen that the way to generate any general fractal image is to use the collage theorem and to recapitulate what the collage theorem essentially says suppose there is a underline space $_{R2}$space of two real numbers and there is an object L in that which means that in this space, I have some kind of a object which is nothing but a compact subset of the $_{R2}$space. So this is my starting point L. so what is L? L is an element of the Hausdorff space. So there is a underline space R_2 in which there is an element and we will assume that there is a (x,d) which is the complete metric space.

(Refer Slide Time: 00:01:07 min)



Then somehow we'll construct some affine transforms w_1, w_2, w_3 so on and so forth. So that when w_1 is applied not only on a point but all the points on the set, it will produce a subset of the set. So the whole set when operated on by the metrics or the affine transformation w_1 then it gives a subset of the original set, w_2 gives another subset of the original set, w_3 gives another subset so on and so forth. so that their union gives me back the whole. Suppose, somehow we have been able to construct such affine transforms then the collage theorem says that the Hausdorff distance between the original set L and union of the… (Refer Slide Time: 03:50). Here we have the original set L and L transform by w_1, L transform by w_2, L transform by w_3 all that taken a union of which means the transform thing, first one iterate gives name, Suppose this distance is very small which means that we have started from the set L and we have applied certain affine transformations on the set L thereby we have got this union, another object so we can find the distance between original one and the transformed one. We find that is small number

epsilon. If that is so then the theorem guarantees that if you calculate the distance between L and the attractor of that IFS, IFS is iterated functions system which means w_1, w_2, w_3, w_n iterated a large number of times on any initial set will always converge on something the attractor so that the distance between the original set and the attractor will be epsilon by one minus lambda, where lambda is a contractivity factors of this system. what do you physically, what will you do?

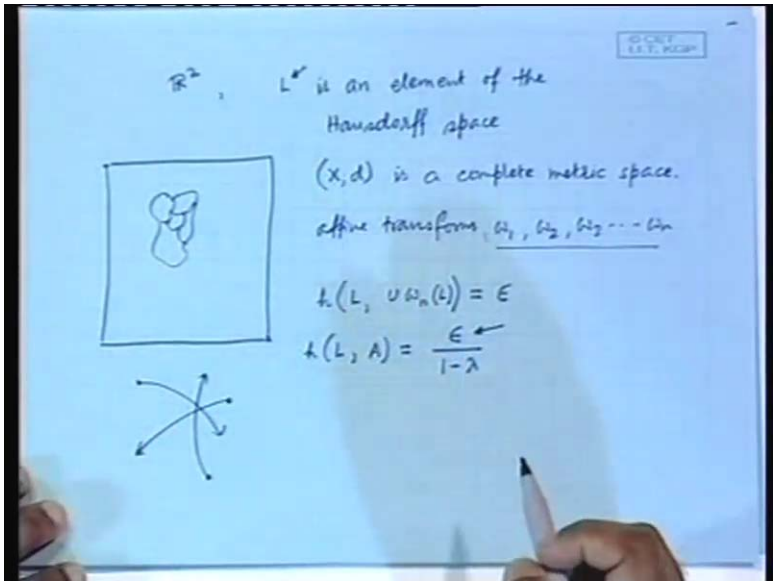
We have start up with some image, we will somehow find the affine transformation so that you get parts of that image. So that the union becomes as close as a original image is possible that distance is our epsilon. if you do so then we can forget about that set, forget about that initial object and then we can simply keep on iterating this iterated function system w_1, w_2, w_3, w_4 and all that on any initial image and that will ultimately converge on to the set viewer this only says that when it converges, it cannot be said that it converge exactly on the set L, it can be say that it converges on some attractor A and then the distance between the L and the attractor A will be small, that's all. If this epsilon is non zero then it will only be small but you cannot say that it will be zero. If it is zero then obviously this is zero. In that case A becomes nothing but the L. So with this if you want to obtain the set of affine transformations what we will need to do is to find out how can we transform the whole image to obtain the part of the image.

(Refer Slide Time: 00:06:52 min)

For example here on the computer screen there is an image. In this image, by looking at it is a coral image, can you find out which part you would like to obtain from the whole by looking at this image? See the whole, the problem is how to transform the whole so that you get a part of it. You see the part on the upper right side is similar to the whole so that can be obtained by means of a affine transformation of the whole. The one that is at the middle that's also a affine transformation of whole and the one that is at the lower left corner this side is also a affine transformation. So I can simply by looking at it I know that you need three of the w's in order to produce this image.

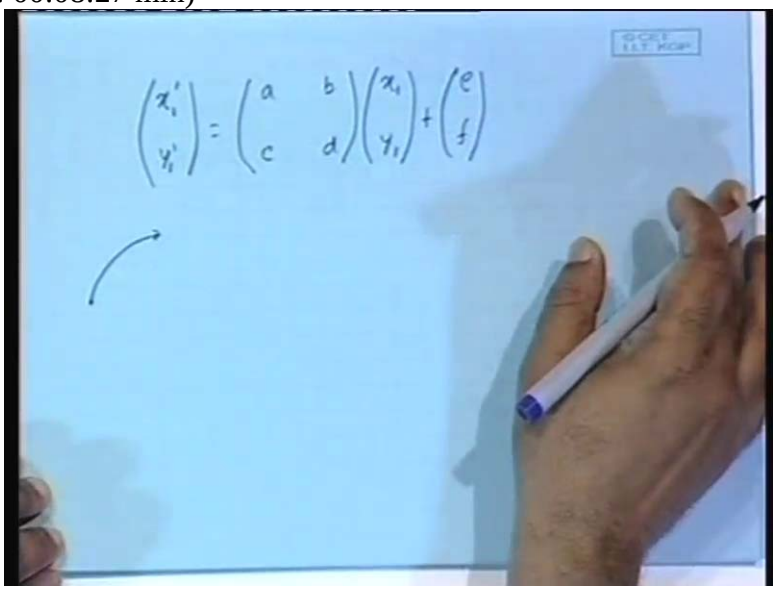
The next question is how to obtain the values in the affine transformation? That can be obtained by simple 6 by 6 metrics solution.

(Refer Slide Time: 00:08:06 min)



What you will do? You will find out that a particular point maps to a particular point, another particular point maps to another particular point and the third point maps to another point. The moment we have identified three points that map to other points, you can easily write down the equations like so x_1 dash y_1 dash is obtain as a b c d, x_1 y_1 + e f. For each of this transformations if you can find out this point maps to this point, we have x_1 y_1 mapping to x_1 dash y_1 dash therefore if you substitute here you get 2 equations, 6 unknown's.

(Refer Slide Time: 00:08:27 min)

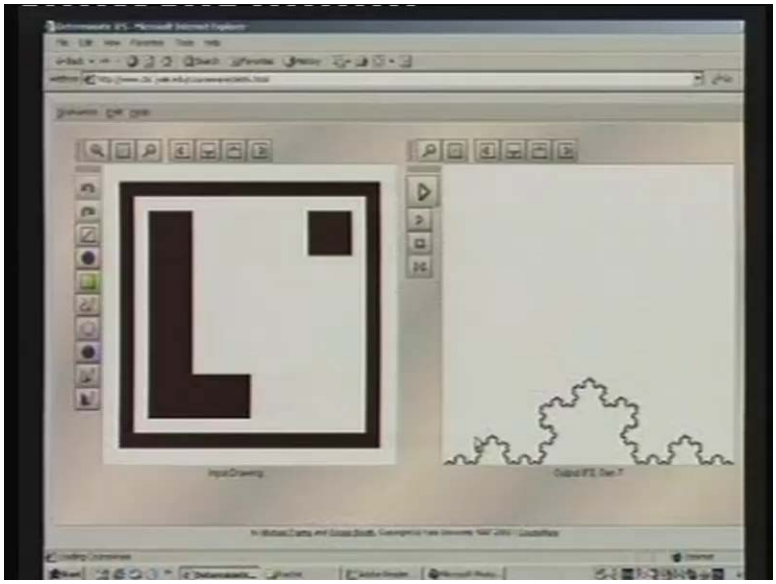


If you can identify three of this points you get six equations six unknown's and that's enough, you can always find the numbers. So in this case I will give you the task of identifying these numbers for this particular fractal image.

(Refer Slide Time: 00:09:44)

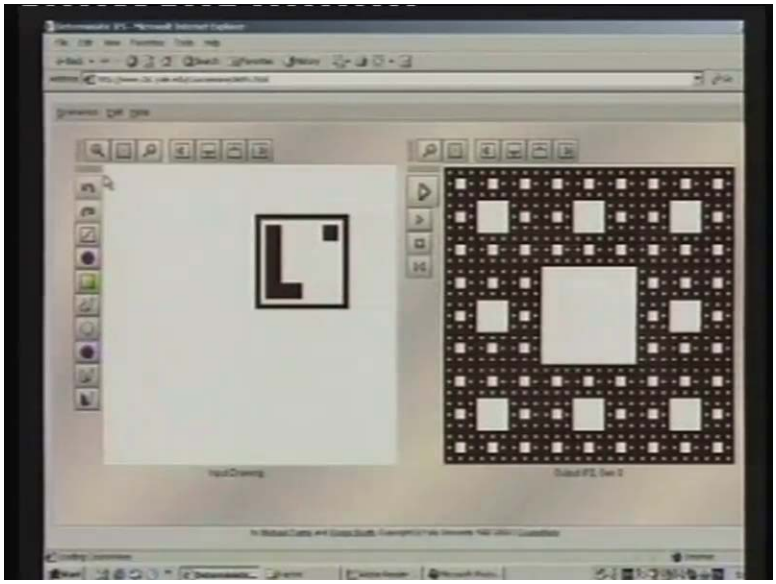
We have already treated this in the last class. Now I wanted to bring back this because i wanted to show that the way the probabilities are assigned, I have already talked about the probabilities in the random iteration algorithm you need the probability in order to generate this fractal. In what way? In the random iteration algorithm you start from any point and then keep on applying the w_1, w_2, w_3, w_4 in sequence to those points and the probability of having w_1, the probability having w_2 and all that is assigned depending on some pre assigned probability. So you need the probabilities. So here you can see that the four transformations land the image into four parts which are color differently in this case and the region marked by red and the region marked by blue are almost the same size, value; region marked by green is much larger. So in order to make sure that the similar number of points land there so that the density of points are similar, you need to put a different probability in that part. The point that I would make next is that such a immensely complicated image required only 24 numbers, to codify this image which means that this allows a very large compression of the information contained in the image. What exactly did we use? We use the concept of self similarity, the fact that part is similar to the whole that is what we have used. If the part is not similar to the whole then we not be successfully applying the same kind of algorithm but if it is so like here you can apply this method and obtain a very large compression. So here we are talking about image compression really and we are able achieve a very large compression ratio and the success obviously depends on the kind of safe similarity that we have in a particular image. Let me go back to another image.

(Refer Slide Time: 00:12:21 min)



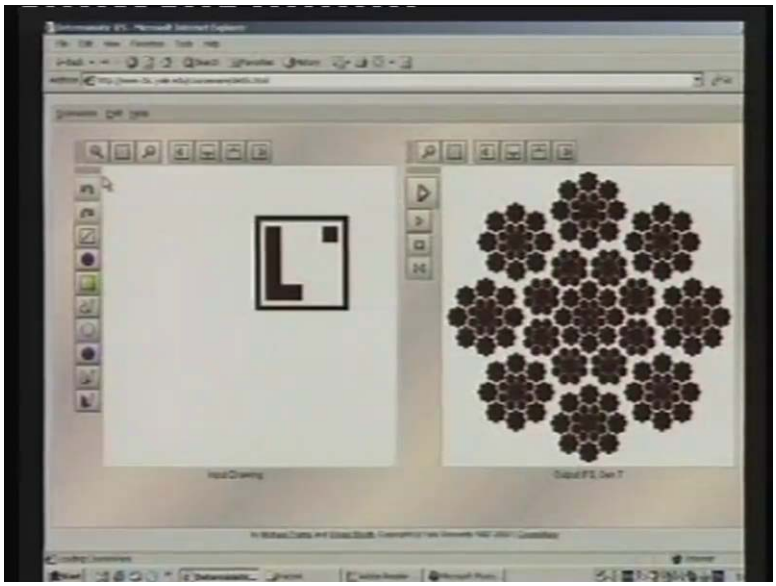
Here we have the core curve as the final image L. Can you identify what are the affine transformation necessary in order to obtain this or how many would you need for that, how many of the w's will be needed? 3? No. See can you see the image? Here can you see the curser? So the whole range is from here to here and out of that the whole thing when affine transformation can be made into this small part. So that can be one affine transformation. the same thing you know affinely transform that means same thing when you rotate bit you get this part, same thing when you rotate bit you get this part, same thing when you transfer it bit you will get this part, union of them this mean by the whole set. So you need four affine transformations to generate this image and you can easily write that. For writing the IFS for this particular image you don't need you know six by six matrix calculation. Why don't you need? Yes, because initially i have found that this a b c d e f can be represented as R_1 cos theta $_{R2}$sin theta that kind of representation that means the kind of shrinking and kind of rotation can be obtained right from there. So you can compose these in terms of R and theta thereby you can directly write down the numbers depending on how much you shrinking and how much you rotated. So you can write down the IFS of such an image just by looking at it with hand, simply use a protractor to find out by how much you rotated that's enough. Let's go to another image.

(Refer Slide Time: 00:14:40 min)



You know where it's converging to? So in this particular image, how many will you need? Right in the first iteration you knew that it will required eight, such common sense is always handy. How will you make it? Give me just one of the affine transforms, you will need to shrink the square into one third of it in each side. So that will be the elementary affine transformation and then you to translate them into 8 different places. So that will not be very difficult for you to do.

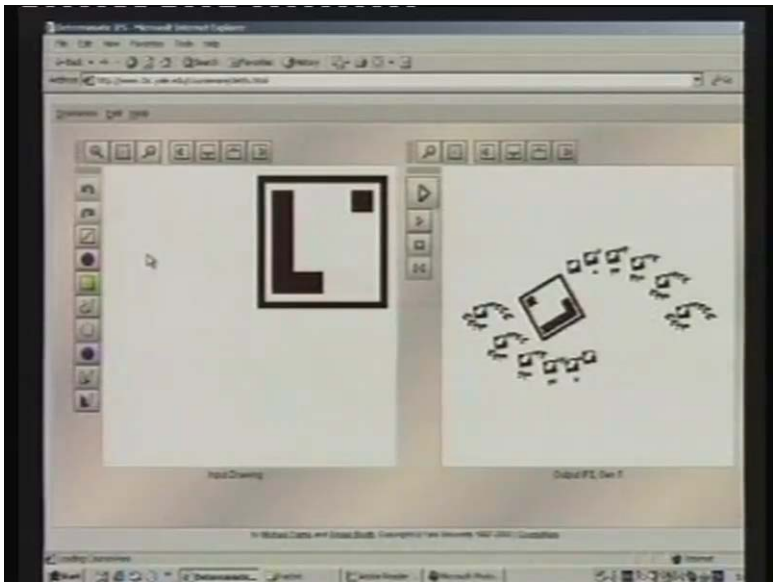
(Refer Slide Time: 00:15:47 min)



Just look at this. What will you need in this case? As it is happening you know how many you need because 1, 2, 3, 4, 5, 6, 7, 8, 9, you need 9 of that. You immediately know but how will you

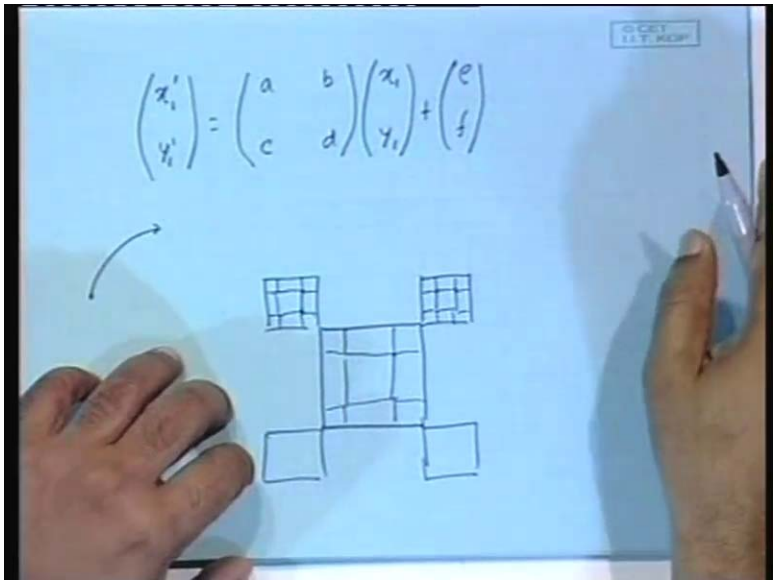
transform them, how do you obtain the transforms? Can you just look and tell me.You will keep rotating, initially for the outers ones you need to shrink it, translate it and then keep rotating by an angle. That can be obtained from that R theta kind of structure of this invisible matrix. Middle one is another that means you need an affine transform for the middle one with a different contractivity factor. This once will have to be contracted by a factor which are the same but the middle one requires different contractivity factor, you can see that. So you will need 9 of them.

(Refer Slide Time: 00:18:29 min)



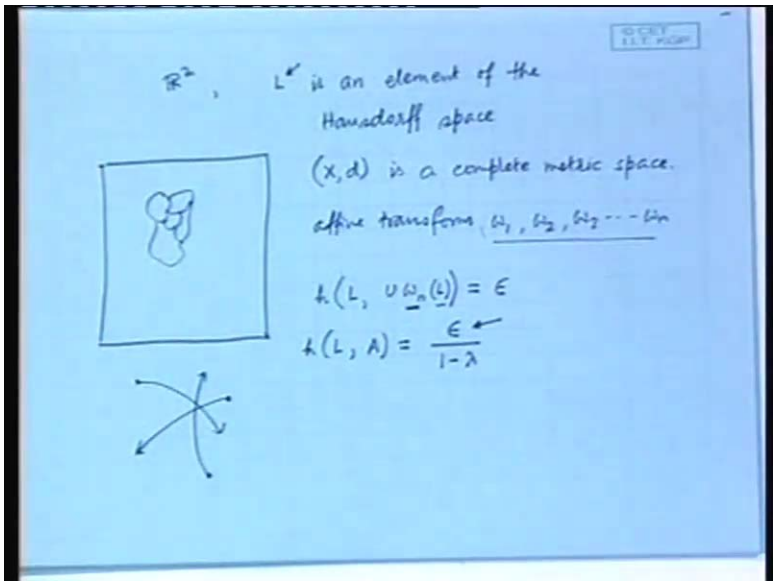
The way it is going, you will immediately know. Now you need more (Not Understandable) (00:18:10 min). You can see that it's converging on to something. How many you would need? You need three really but this is sometimes difficult if you look at the final image only. Now you are seeing it happening, so you know that in each iteration it is really doing a thrice. So you know that it requires three but you should develop the ability to look at an image and tell no this requires really three. So that requires some practice. So what I'll asking to do is to get this back okay.

(Refer Slide Time: 00:19:12 min)



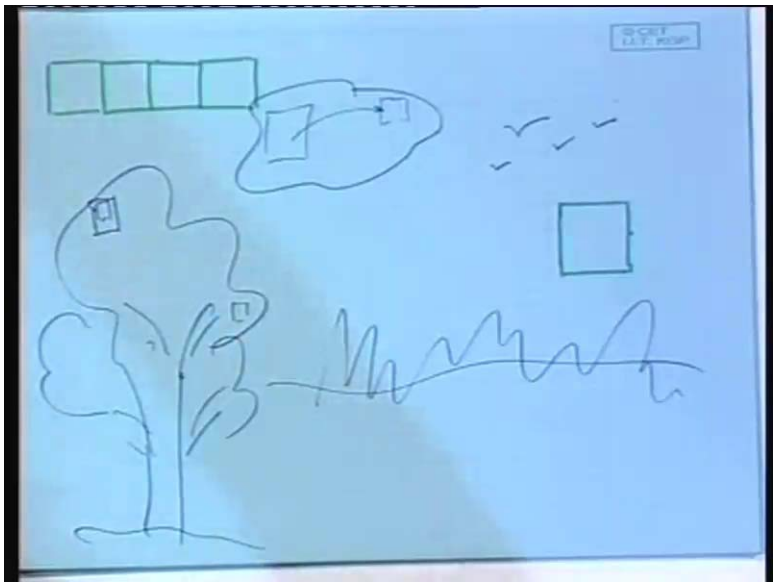
Suppose there is an image where you have, look at the drawing here. Imagine that this kind of structure is there in each of them. So how to do that? This is there and this is there, suppose you have this as the image that you have to converge on, what would be the structure? The whole thing you will make one affine transform to create the middle one, one here, here, you need five of them. The point is that the moment you have created all the 5 then that information, those numbers are sufficient to create this object out of anything. So after having this much practice, we are in a position to take the next step. Look at the structure of the collage theorem.

(Refer Slide Time: 00:20:37 min)



We say it that there is an element of the Hausdorff L, x d is a complete matrix space so that as we take the steps we know that we always land in an element of the Hausdorff space. Now we had defined the transforms or we wanted to make the transforms in such a way that the transforms things, their unions creates a very close resemblance to the original image and at that stage we had sort of assumed tacitly that we will apply the w's to the whole image to get parts of it. Is it really required by the theorem? The way we prove the theorem, essentially all you need is to apply some transformation to the image L. But at this stage, it doesn't say anything as to whether we have to apply to the whole image or the part of the image. This algorithm was working fine for this kind of images that we have seen because in these cases the hole is really transforming to the part but if there are many cases where the hole is really not transforming to the part, whether you can get a part from a part. There are same similarity between parts but not that the hole is transforming into a part. In those cases you can apply the same theorem without any difficultly. So in this case when you apply the omega n to L, we will only say they will apply the w_n to not L not the whole set L but the part of the set L then also it is true. There was nothing in the step that we followed in order to derive the collage theorem that excluded their possibility. So that brings a very interesting possibility that suppose you have some general image.

(Refer Slide Time: 00:22:55 min)



Right now I do not have an image on this but imagine that on this you have got some kind of image. Say some kind of a cloud here, some birds flying here, a tree here and whatever it is, you can imagine any kind of scenery. Now here we have the clear sky, here we have the clouds, here we have these things. Now you see the sky there is a great deal of what is known as affined redundancy in these parts of the image. It is not necessary to store all this information related to all the parts of the sky because in the parts of the sky, a part can be obtained by an affined transforms of the another part. if that be so, then obviously we can store the image as which part transforms to which part, there by finally creating this image. So how will we go ahead with this kind of algorithm? What we'll do is we need to define a few affined transformation w's.

As you said that in this case w's will be transforming parts to parts. Say we find that this part of the cloud is quite similar to this part of the cloud, only if you do some affined transformation you get this part. If that be so then the larger square can be made to transform into the smaller square not the other way round because you need contraction mappings. So you always have to find some larger area in the image which when affinely transformed in this smaller area within the same image. So in that case say there is a tree. The tree has some part here, another part there, a larger part here and then this part can affined. If you find that this part can be affined the transform to get a very close resemblance of these part then the information contained in this part is really redundant.

Similarly you will find that in most images there is a very large extent of affined redundancy. So the algorithm that the fractal image compression relies on essentially takes care of this affined redundancy and tries to find out the minimum information necessary in order to store that image. So here the idea is what we are dealing with is fractal image compression. The standard way of image compression is discreet cosine transformation, while in this case we go by completely different logic. There it is the higher order terms that are eliminated but in this case we essentially talk about the affined redundancy. What we will do is we will divide the images into boxes, say a box like this, a box like that, a box like this and so on and so forth. Then suppose I am trying to find out where the first box comes from that means we are trying to find out if the information contained in the first box is really necessary or redundant. If it is redundant then its information must be available by some affined transforms from somewhere else. So suppose it is this part has to be a larger square from which it comes that means the original image is divided into smaller boxes which we try to find from where does this information come from and then we scan the whole image and then we try to locate. Here is a place which when affinley transforms give this image. So, this image this part, this subset maps to this subset. These blocks from where this information comes to here are called the domain blocks and the ones where it maps to are called the range block. So the whole image is divided into the domain blocks and range blocks and finally we try to find out for each range block where does the information come.

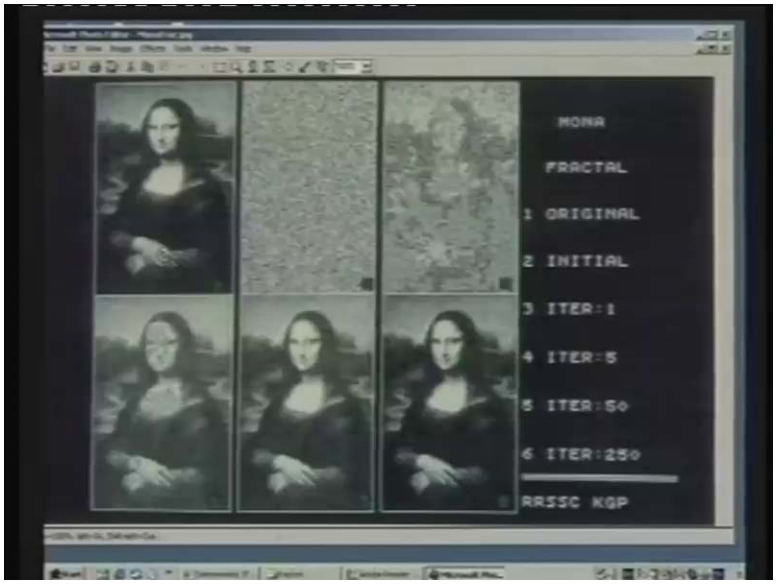
Now you would notice that form lesser number of choices because normally you have seen that affinely transform in any possible way but here since we will be dealing with squares, squares can be only a few ways transformed in order to give another square. So this square can be rotated in all possible angles, can be flipped or can be shrunk. So these are the possible things but it cannot be rotated in forty five degree's in order to get this image. So there is only a few number of affined transformations possible that will get from this image, it will get this image. So the number from which you choose is rather limited in this case.

In the earlier cases we had considered an infinite array of possibilities but here the array of possibilities will be limited. So you can only check from a array of possibilities which one fits is the best and find out of the whole image which particular domain block gives me the list, out this squares. Then we'll go to the next one, do the same procedure, go to the next one, do the same procedure and finally when you have scanned the whole image then what you need to store? You need to store that for this particular range block, the source was here another particular address of the domain block plus this specific affined transformation that was necessary in order to bring it here. So you do not need to store the image pixel by pixel, you don't need that.

You are dealing with relatively larger say 8 pixel by 8 pixel kind of blocks and we have found that here is a 16 by 16 pixel block from where it comes. So you need this address and this address and you need the transformation that's enough. So at the end of the day what you storing is nothing about birds, nothing about sky, nothing about the tree, nothing about grass, nothing. You are only storing some addresses and the transformations necessary to bring this domain block to this end. Then after having done so, you can start from any initial image and you will always converge in to the final image always. Some questions are obvious. Firstly, so far we are dealing with this binary images that means we are talking only about zeros and ones either a particular dot is there are not there but in actual images, if you are talking in terms of say a gray scale images a particular dot is either there or not there. That's not the situation, there can be grayscale value assigned to each fix cell. How to take care of that? In real images there are colors also, how to take care of that? The way to take care of that is there are few ways but one you can imagine this way. That for each one so far we have been talking about R_2 space in which we imagine this.

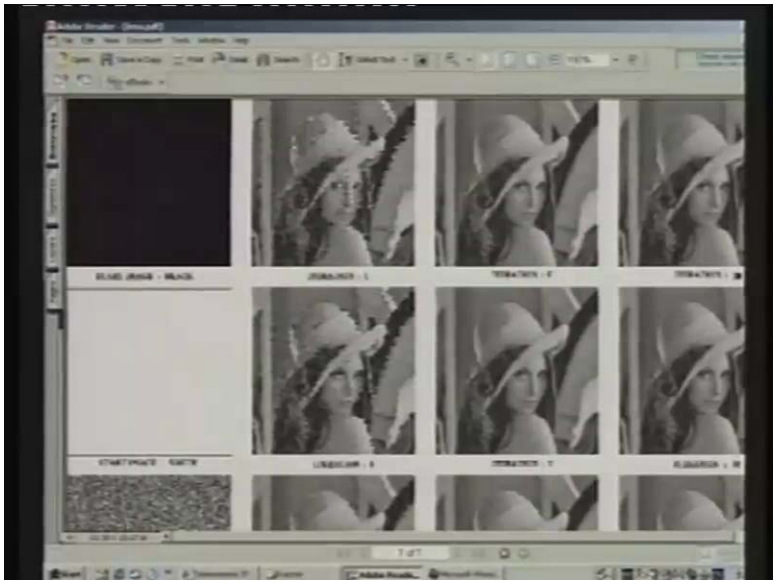
Now for each one for each pixel there is an additional element to it that is a grayscale value and the grayscale value varies from zero to two hundred fifty six. So you might imagine that as a depth. So apart from a value, apart from whether it is there or not there. There is also a depth element to it so that we are essential dealing now with R_3 space. We can do the same thing. There was nothing very special to $_{R2}$space that we talked about. So we can think in terms of R_3 space that means there is the x coordinate, y coordinate and a z coordinate is nothing but the gray scale value. So this way it is possible to obtain the image and you can do the same procedure. In terms of colors what we normally do is if you can obtain the grayscale image then the color image is nothing but four grayscale images in four of the elementary colors. So essentially if you can store the image in four grayscale images, you have got the color image. So the generation of color image is not difficult. Remember this particular algorithm is successful only in cases where you have great deal of affined redundancy. Otherwise it does compare with other available image compression algorithms like in JPG and NPG'S also, those use the discrete cosine transforms. There in particular types of images they perform better and in some types of particular images the fractal images compression performs better.

(Refer Slide Time: 00:33:54 min)



So at this stage let me show you a few things. For example here we have started with the L, the image of Monalisa that is our subset L and we have obtained the set of affined transformation that we will get it there. So we can start from any arbitrary image which is middle one, the second one. Can you see the curser moving? Yes, so this is the starting image for the decompression algorithm and as we go on applying that this is the first iterate, this is the fifth iterate, this is the fiftieth iterate, this is 250^{th} iterate then you see it is almost same as the original image. So that has been done in between is for this particular image, this algorithm has been applied that means the whole image as been divided into parts, the range blocks and for each range block an algorithm has been run on it to find out from which particular domain block it brings it there to some affined transformation. From an array of possible affined transformation a $particula_{r1}$has been chosen which gives the best fit and thereby we have obtaining whole image. We have codified the whole image and that code when ran on the actual image we get same image back. This images is may not be all that clear because this is actually a screen shot that means a actual photograph taken from a computer screen.

(Refer Slide Time: 00:35:43 min)



So let me show this one which is the Lena image. It is a standard bench mark used in image compression. So for this also the image compression code was generated and if you start from a completely black image you can see this iteration 1, iteration 5, iteration 30, you are almost there. Completely white image, again same process you get the random image. There also you get, can you see the iteration going? This one and this one would be different because the initial conditions were different but ultimately converges on to the same conditions and it started from the parakit and that is also converges on to the same image which essentially shows that it is independent of your initial conditions. I had a plan once to start from my image to converge on to some good actress but it dint work out. So is it convincing now? That it is possible that it's durable, if it is doable then these are all black and white images though but easily understand but any colored image can be broken up into four black and white images and you can do the same thing.

So in the study of fractals what actually we have done? We have first seen that the natural objects are fractals. Natural objects are fractals in the sense they have complexities within complexities and at no level of magnification does the complexity seems to exist. So these kinds of structures, these kinds of objects, these kinds of geometrical objects need a different kind of description and so we found that description, that quantification can be done through the idea of dimension of the object. We have obtained the quantification of the dimension, how to obtain the dimension by what is known as the box counting algorithm. You have done the box counting algorithm where you have a grid of like a graph paper, put on the particular object and you count the number of boxes and you put on that algorithm and you get the value. We have also seen that this particular number that we get as the dimension of that object that not only satisfies your curiosity but it as physical significance in the sense that in situation where the crookedness, the curvedness, the non-smoothness these things matter. There that number is a real quantification of the phenomenon under consideration.

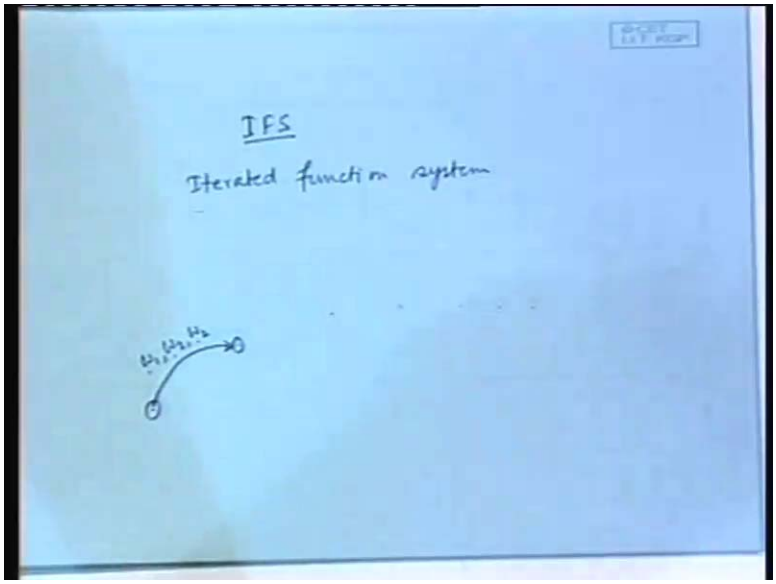
A very glaring example would be the human lungs. So its fractal dimension actually tells the amount of surface that is exposed to the air and as the quality of lungs degrades, it has been found that the fractal dimension changes. The solid catalysts for example, the amount of surface that is exposed to the reactants that depends on the fractal dimension. So it is not only the material, not only the quantity, not only the surface area but also the fractal dimension that is also important in these cases.

We have seen that fractals are important in the study of dynamical systems in the sense that in a dynamical system, we have a state space consisting of a state variable and in a state space we have initial condition and then if you have this set of differential equations then then you can evolve the initial condition to get the values at certain points. For certain initial conditions, the state may remain bounded and for some other initial condition it may go to infinity. So if you plot those initial conditions in this state space then you get a geometrical object whatever it is. In some cases that geometrical object is a fractal object, in some cases it may not be fractal object. So that it is always a fractal object. In dynamical systems the fractal objects also appear in parameter spaces that means every system has some parameter and the parameter if you look two parameters as X coordinate and Y coordinate and then any point in that parameter space will be a specific choice of parameters.

You will find that for certain choices of parameters, the system remains bounded or the system exhibits some kind of behavior. For another choice of parameter it may exhibit either a deferent kind of behavior or completely unbounded behavior. So if you now map the parameter space for the particular sets of parameters for the same kind of behavior then you get a set. So these are the issues that are often misunderstood in studying the fractals because fractals are beautiful, fractals are nice, looking fractals are good mathematics, all these are fine but actually they are representative of physical phenomenon. So in physical phenomenon you have the state space as well as the parameter space and the way we have seen that the state space its representation, a very toy model was given in terms of that Z_N is equal to Z_N square plus C and then we found that you have fractals both in the parameter space as well as the state space. What is it called, where it is the fractal in parameter space? It is a Mandelbrot set. What is called and where it is the fractals in the state space, it is a Julia set. So when you are studying any dynamical system, you do expect such things to occur but they will look different, they will not look like the Mandelbrot set but always you should remember that there are fractal objects in the state space as well as the parameter space.

Then we went on to the generalize method of understanding fractals that means fractals in general are compacts subsets of the underline space. We have considered that underlined spaces of $_{R2}$space, compacts subsets of that underlined space is basically images. So we have images and we said that all possible images, let there be a space where the elements are all possible images and then after a few very natural steps that are done in any study in functional analysis, we arrived at the confusion that the space is complete, the space is a metric space and we succeeded in deriving some transformation that will take a point of the space to another point of the space, meaning an image to an image. A succession of these, a repeated application of these iteration of these will lead to some kind of image that let us to the idea of the iterated function system IFS.

(Refer Slide Time: 00:43:47 min)

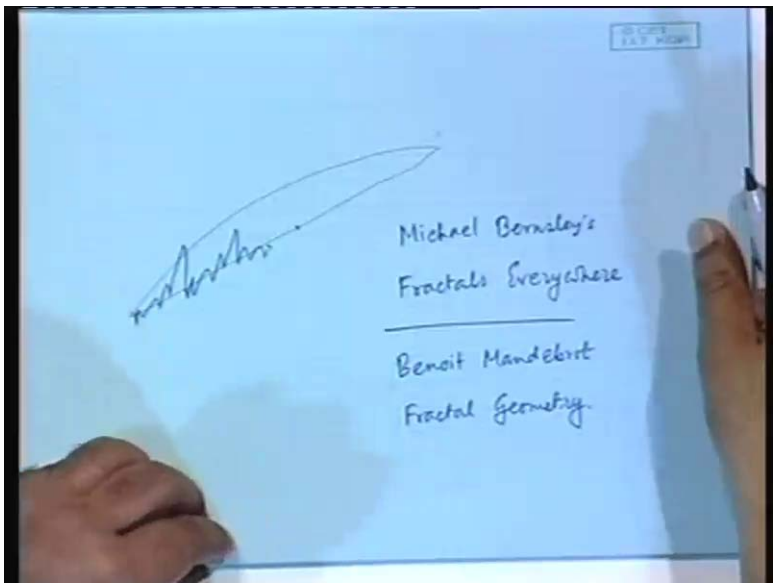


So what is an IFS? IFS, iterated function system. It is basically taking steps in the Hausdroff space. So one point to another in the Hausdroff space is the collection of w's so that from here to here you apply each of these affined transforms to the original set, you get the transformed ones and take the union. So all that taken together is the iterated function system. iterate that repeatedly, you always get to some kind of image. So long as the contractivity factor of the iterated function system is less than and that is guaranteed so long as since these are affined transformation, there is the a b c d matrix. If the determinant of that is a b c d matrix is less than one, very simple. So you can always set the values so that they are less than unity and you can always be guaranteed that it will converge on to something. So it's nice to play around with it, it is nice to simply say okay let me choose a particular three affined transforms $_{w1}$, $_{w2}$, $_{w3}$ and let set the values. I know that it is contractive and let see where it converges.

So I argue to play around with these things then that will give you some idea of the relationship between the ultimate image that you get and the kind of affined transforms you started with. Then we have done the idea of the generation of the iterated function system from any object that can be done for the complete same similar object in a deterministic manner, complete same similar means where the complete thing, full image maps to a parts. As it happens in the fern image, as it happen in the three image, as it happens in the spiral image so in those images it is possible to obtain the part from the whole so you have a particular time. There are also situation where the parts are obtained from the parts of the image. In those cases also you can write down the iterated function system but if then have to say which part mapping to which part. For the full one you did not need to store the coordinates of the location of the source because the hole maps to a part but if you map a part to a part then obviously you would need to store which part comes, which parts means it has some coordinate, some address. So you will need to store that address.

Obviously when you have a part mapping to a part that kind of a situation, the contractivity factor or the compression ratio would be smaller because you are additionally have to store the information about the place from where it comes. So this can be applied to binary images, you can easily do that by simply binarize any of these images. so you have the Lena image, you have the Monalisa image, you have any of the images that we can down load from the net and you binarize it, using any of the software that you have. Most of the window software that are able to do that. So you get binary image out of it and you can apply the same algorithm. It will be very simple to apply the same algorithm on binary images. So provided you have time and provided you have the playfulness, you write the code, at least do it for binary images that means to search the image for proper domain blocks which will map into proper range blocks. I am not giving you as an assignment to be done by everybody but those of you who are enterprising enough to try it out, do try it out. The one where you are really using gray scale images that is relatively difficult to code because of the third dimension, the depth. Relatively difficult, so i do not think you will be able to do it within the space available for this course but the binary images do it and of course you have the vacation now. If you have a computer you can always write the code for even the gray scale images. So that is almost all about fractals, almost means there are also issues like these days generation of fractals are used even in interpolation. For example if you have some data from here to here and if you are asked to interpolate what will you do?

(Refer Slide Time: 00:49:00 min)



You either interpolate by a straight line or by some kind of a flying kind of flit, you don't have any other way but if you know that from this point to this point, the data is like the share market index. Obviously you know that in between it will bend like this, it cannot go like this. So if you now have two points and you want to interpolate, how do interpolate? Obviously you cannot interpolate by a straight line or a flying kind of it. This square kind of it, you can do that. So in those cases you need a fractal to interpolate this to this image. How to generate those fractals? Yes, there are methodologist for that.

There are methodologist to generate surfaces not only curves but surface. For example if you see a rock surface, if you see a mountain surface it has texture. How to generate that kind of a texture? So there are issues involved with generation of that kind of surfaces which i will not go into but those of you who are enterprising enough, you can read the books. There are many books that are now available; the elementary book is Michel Bernsley's Fractals Everywhere. There is also a book by fractal geometrical or something like that. So now you may go ahead with reading these books and further information can be gathered but for the purpose of this course that is all. So this is the extent to which we have covered on fractals in this course. Thank you very much.