

# Digital Image Processing

Prof.P.K.Biswas

Department of Electronics & Electrical Communication Engineering

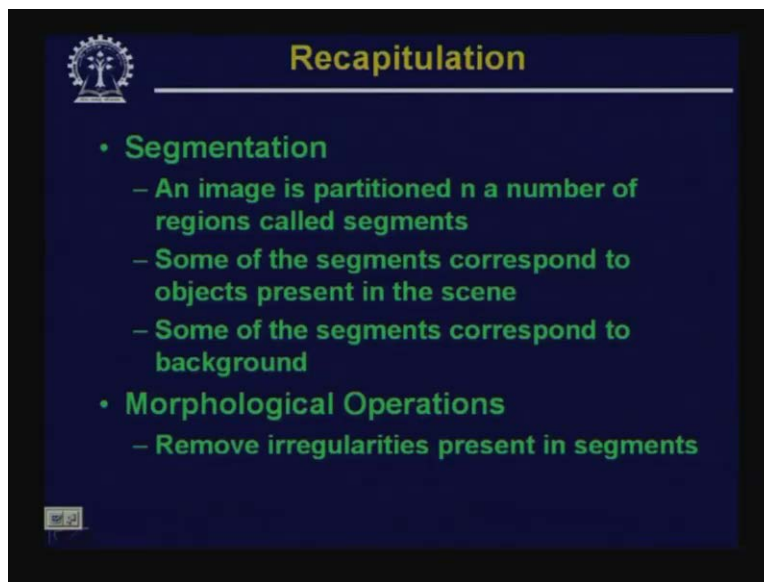
Indian Institute of Technology, Kharagpur

Lecture - 37

## Object Representation and Description-1

Hello, welcome to the video lecture series on digital image processing. Now, till our last class, we have discussed or we have completed one aspect of image processing. That is we have done the morphological image processing techniques on the segments where we have seen that if **the segments** after segmentation operation, the segments that we obtain those are not regular; then the segments can be regularized by using the morphological image processing operations.

(Refer Slide Time: 1:31)



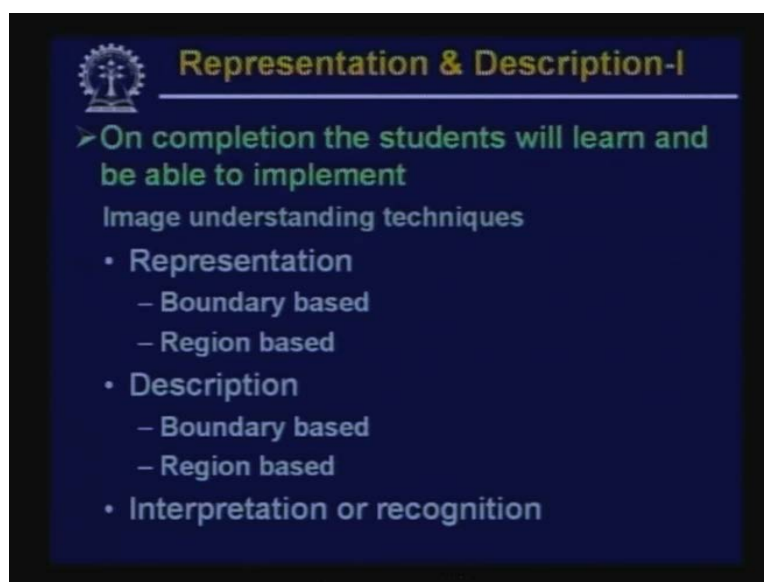
So, till last class, what we have done is once an image is segmented, we have said that the purpose of image segmentation is that an image is partitioned into a number of regions which we call as segments. Some of the segments that we obtain correspond to the objects present in the scene and some of the objects or some of the segments correspond to the background region in the image and as we have said that because of the presence of noise, the segments that we get, those segments may not be regular segments. There may be some noises present in the object region or even some noises present in the background region.

If there are more than one objects present in the image, then it may appear that because of the presence of noise, those different object segments may be joined. One may be joined to the other by some spurious boundaries and these boundaries are also due to the noise. So, what we have seen for during our last few lectures is that we can employ morphological techniques to remove such kind of noises and after doing the morphological transformations, the segments that we get are quite regular. So, for that we have talked about various morphological operations like dilation, erosion and other morphological transforms which are generated which are developed using the basic morphological transformations like dilation and erosion.

Now, from today's lecture, we will gradually move to some other aspect of image processing which we will call as image understanding. So, what we want to do in image understanding topic is that given the segments or the segmented output of an image or regularized segmented output of an image whether it is possible to understand or make the computer understand what is the object present in the image or what is the object which corresponds to a given segment present in the image.

So, in order to do this, in order to make the computer understand the objects present in the image, we must have some proper description of the segments or some proper description of the objects present in the image so that these descriptions may be matched against some knowledge present or stored in the computer beforehand. So, once you have such a description which can be generated from the segments present in the image, these descriptions may be matched against the description which pre-stored in the computer memory in the form of model base or in the form of knowledge base and once we find that a description generated from a particular segment matches against the description of a particular object; we can immediately infer, the computer can immediately infer that this is the objects of object X or object Y which is present in the image or a segment say **l** is a segment 1 corresponds to say object X which is already there in the knowledge base.

(Refer Slide Time: 5:15)



**Representation & Description-I**

➤ On completion the students will learn and be able to implement

Image understanding techniques

- Representation
  - Boundary based
  - Region based
- Description
  - Boundary based
  - Region based
- Interpretation or recognition

So, in order to do this, what we have said is the first operation that we have to do is we have to find out a proper representation mechanism. That is given a segment, how do I represent that particular segment. So here, 2 possible representations; the segments can be represented in 1 of the 2 possible ways. One of the possible ways is the boundary based segmentation and the other possible way is region based segmentation.

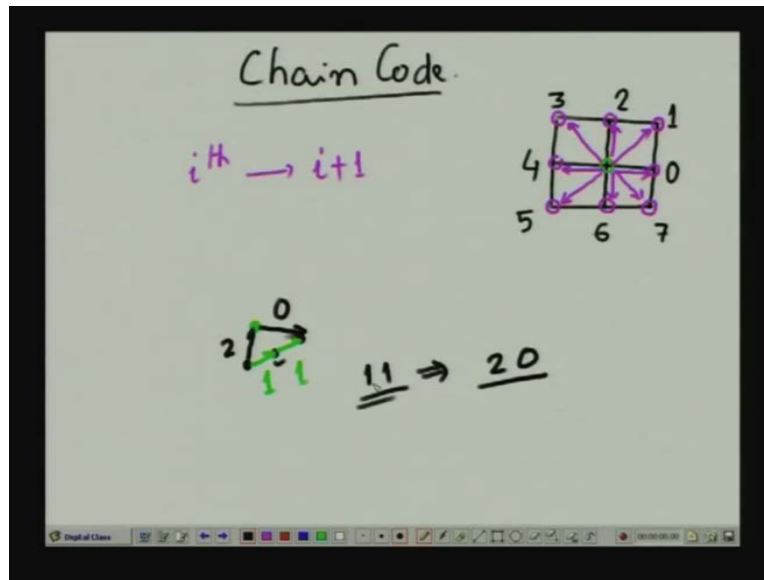
So, in case of boundary based segmentation, what we do is you take the boundary or contour of the segment and represent the boundary in some form so that it helps us to generate some description of that boundary and then finally, using this boundary based description, we can try to match this description against similar such description present in the knowledge base. Whereas, in case of region based description or region based representation, what we are interested in is not only in the boundary of the region or boundary of the segment but we are also interested in the surface property. So, we can have a region based representation, from that we can have a region based descriptor, description generated from the regions and these descriptors can again be matched against the similar descriptors present in the knowledge base to identify a particular object.

So, it is quite clear from this that if we are going for boundary based representation and boundary based description, then what we are interested in is mainly the shape of the object. Whereas if we are also interested in the surface reflectance property such as the color texture and so on; in that case, simple boundary based representation and boundary based description is not sufficient. What we have to go for is region based description and region based representation.

So, we can have one of these 2 types of representations as well as the descriptions; the boundary based representation and boundary based description. Similarly, we can have region based description and representation and region based description. And finally, once you obtain the descriptors whether the descriptors are boundary based descriptors or the descriptors are region based descriptors; finally what we have to go for is some matching mechanism which will match these descriptors against similar such descriptors present in the knowledge base to identify a particular object.

So, in today's lecture, we will mainly concentrate on the boundary based representation and what are the different types of the descriptors that we can obtain from boundary-based representation. So, the first such representation scheme that we shall consider is called chain code.

(Refer Slide Time: 8:41)



So, as we said that this chain code is a boundary based representation scheme and from this boundary based representation, we can find out what is the boundary descriptors that are possible to obtain. So, let us see what is this chain code. The chain code is something like this, say given a boundary in digital domain, as we are talking about the digital image processing techniques; this boundary is nothing but a set of points set of boundary points, discrete boundary points on the object contour.

Now, what we can do is once we move from one point on the object contour to the neighbouring point in the same object contour; then I can have **one possible** one of 8 possible moves if I move, go for 8 connectivity. So, it is simply like this; so suppose, **I have** let us consider a regular grid, so as we have said earlier that if I assume that I have a center pixel somewhere here and then this center pixel has 8 neighbouring pixels, 4 in the diagonal direction, 2 in the vertical directions and 2 in the horizontal directions.

So, since as we have said that the boundary is nothing but a set of points or set of pixels present in the image and if I assume that this set of pixels are connected and assuming 8 connectivity; then starting from any point on the boundary if I want to move to the next point on the boundary following some mechanism, either we can move in the clockwise direction or we can move in the anticlockwise direction.

Now, as we move say from  $i^{\text{th}}$  point on the boundary to say  $i+1^{\text{st}}$  point on the boundary; then I can have only 1 of 8 possible moves and these possible moves are I can move either in the right direction or I can move in the diagonal direction -right top or I can move in the vertically upward direction or I can move in the diagonal direction to the left top or I can move in the horizontal direction to the left or I can move in the diagonal direction, in the direction of left bottom or I can move in the vertically downward direction or I can move in the diagonal direction that is in the right bottom direction.

Now, you find that as you move from one boundary point to the next boundary point, the displacement or the length that you move is 1 because you are moving from 1 pixel to 1 of the neighbouring pixels. So, the length of this move is equal to 1 and at the same time, each of these moves has a specific direction. So, if you represent these moves by numerical numbers; say for example, you say that a move in the right direction is represented by a number say 0, a move in the top right direction is represented by a number 1, a move in the vertically upward direction is represented by number 2, a move in the top left direction is represented by a number 3, a move in the left direction is represented by number 4, a move in the left bottom direction is represented by a number 5, a move in the bottom direction downward direction is represented by a number 6 and a move in the right bottom direction is represented by a number 7.

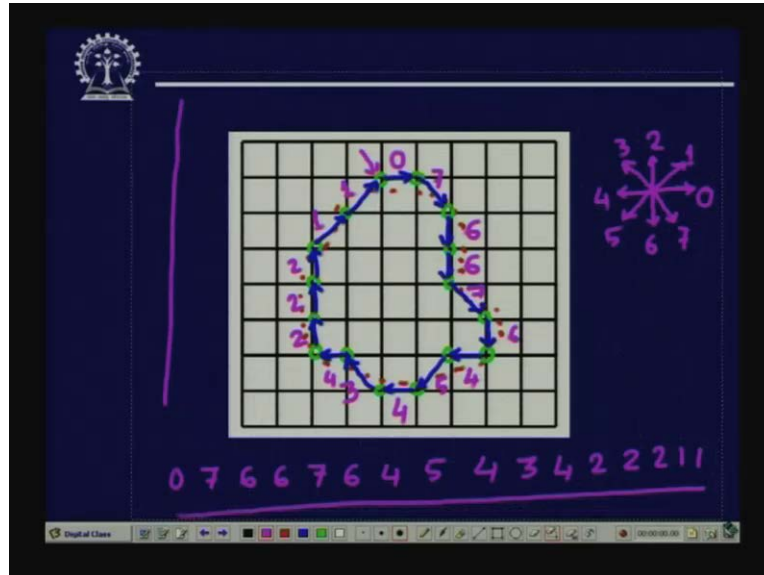
So, if you represent each of these moves by using **this 7** these 8 distinct numerals; then it is possible that as you move along the boundary, the inter move, when you complete the cycle that is when you come back to your starting point from where you have started your move either in the clockwise direction or in the anticlockwise direction, this entire boundary can be represented by a sequence of such moves and as each of the moves has got a specific number, so we can say that this entire boundary is represented by a sequence of such numerals where the numerals vary from 0 to 7.

So, what you get is a code, a numerical code for the boundary which we call a chain code. Now, let us see that for a given boundary; how we can represent how we can obtain a particular chain code? Now, as we have said just now that you move from 1 boundary point to the next boundary point. Now, if you simply apply this, the problem is because we are considering discrete points representing the object boundary; so what happens because of the presence of noise, say for example you have 3 consecutive pixels say something like this, let me draw it in a bigger way.

So, these are the 3 consecutive pixels that we have in an image. Now, it may so happen that because of the discretization process and also because of the presence of noise, the middle point that is this one is slightly shifted. So, instead of this middle point being present over here, the middle point may be shifted somewhere here. So, we find that because of the slight amount of noise, the type of chain code that we will get will be totally different. My ideal chain code should have been a move in this direction and a move in this direction. So, the chain code should have been ideally 11. But because this point is shifted due to noise or due to discretization whatever that is may be, what practically the chain code that you get is a chain code this that is move in the vertically upward direction which is 2 and then followed by a move in the right direction which is code 0.

So, instead of obtaining a chain code 11, what you are getting is a chain code 20 which is totally different from chain code 11. So, this is the major disadvantage of trying to find out the chain code from the original boundary points. So, instead of trying to find out the chain code from the original boundary points, what is usually done is we go for resembling of the boundary points or the resembling of the discrete points representing a boundary. So, let us see how that is done.

(Refer Slide Time: 16:31)



So suppose, I have a set of points representing a boundary say something like this. So, these are say set of points which represents a boundary. Now, the approach that I am taking is instead of trying to find out the chaincode from this original boundary points, I am trying to resample these boundary points by placing a grid. So, as shown in this figure that I have a grid which is superimposed on this set of boundary points; then what I do is I assign the boundary points to one of the grid locations based on its proximity or a boundary point will be assigned to one of the grid point, one of the grid location which is nearest to it. So, based on that, you find that I can identify a **set of boundary** set of grid points which can approximately represent the boundary.

So, what are the grid points that I can have? Now, I marked them in green. So, this is one of the grid points which can be used to represent the boundary, this is one of the grid points which can be used to represent the boundary, this is another grid point, this is another grid point, this is another grid point, this is another grid point, this is another grid point, this is another grid point, this is another grid point, this is another grid point, this is another grid point, similarly this one, this one, say this one. So, these are the different grid points which may be used to represent the boundary.

Now, once I do this, what I do is I represent a chain code, I find out a chain code for this grid points; not for the original boundary locations, not for the original boundary points. So given this, you find that this particular representation or grid points can be represented by a boundary by code like this. So, I can have this move here, I have this move, here I have this move; so this is what will be my chain code representation.

Now, if you remember the chain codes, the code for different reactions that we had is this direction was given a code 0, this direction was given a code 1, move in this direction was given a code 2, move in this direction was given a code 3, move in this direction was given a code 4, move in this direction was given a code 5, this was given a code 6 and this was given a code 7. So, by using these codes, you find that if I assume that this is my starting point; this move is represented by code 0, this move is represented by code 7, this move is represented code

6, similarly here it is code 6, here it is 7, here it is 6 again, here it is 4, here it is 5, 4, 3, 4, then 2, 2, 2, 1, 1.

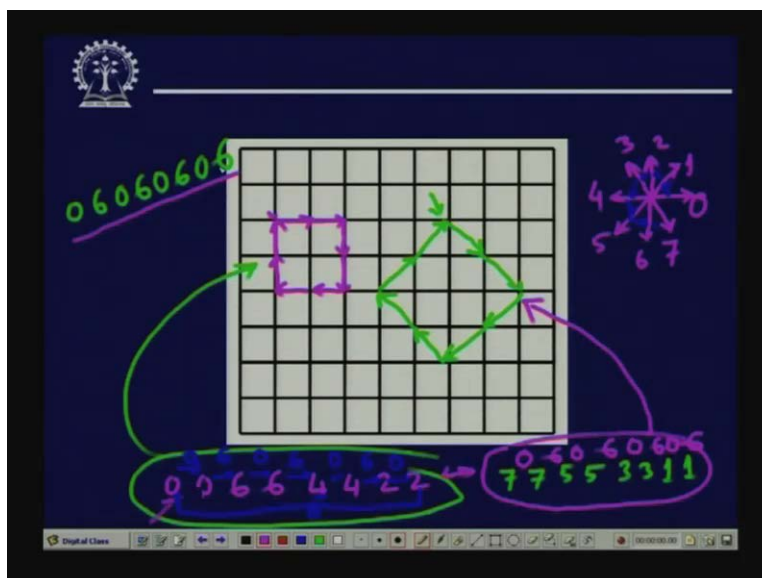
So finally, my chain code representation that I get for this shape starting from the indicated starting point will be like this; it will be 0, 7, 6, 6, 7, 6, 4, 5, 4, 3, 4, 2, 2, 2, 1 and 1. So, this will be the chain code representation of this particular boundary pixel. Again, you note that when we have obtained this chain code representation, we have not found out the chain code representation of the original boundary points but what we have done is we have resampled the boundary points by placing words and based on the grid locations; we have obtained this chain code representation.

Now, more advantages of this is, this can also take care of the scaling. That is by having a control, by varying the grid spacing, we can have the same boundary represented at different scales and accordingly I will have different chain code. Now, though the chain code representation appears to be very simple in this particular case, but it is not so simple because you find that the chain code that we have obtained here is not rotation invariant.

Though the chain code will be translation invariant, we can take care of the scaling by a proper grid size but the chain code in this case is not rotation invariant. But what we have said earlier is our purpose of representation is that **give** once we decide a particular representation, we want to generate some descriptor out of that representation and these descriptors will be used for final recognition of the object or identification of the object.

So naturally, the descriptors that we want to have should be ideally rotation, translation and scaling invariant. But as we have seen in this case that this particular representation, though it is translation invariant, I can make it scale invariant by having different grid sizes but it is not rotation invariant. To illustrate that this chain code representation is not rotation invariant, let us take a very simple figure. Say for example, I take a figure something like this.

(Refer Slide Time: 23:41)



Now, find that for this or let me consider even a simple figure this, again becomes a complicated one. So, let me consider even a simple figure. Suppose I have a figure, a simple boundary of a square and if you remember our different directions were something like this; so, this was 0,1,2,3,4,5,6 and 7.

So, here you find that this simple square, a square boundary; the chain code of this will be if I start from this starting point, will be 0,0,6,6,4,4, then I have 2,2. So, this is a chain code which corresponds to this square boundary. Now, what I do is I simply rotate this square figure by an angle 45 degree. So, what I get? If I rotate the same figure by an angle of 45 degree; the corresponding figure, the boundary will be like this. This is the boundary if I simply rotate this square by an angle 45 degree either in the clockwise direction or anticlockwise direction.

And, if I want to find out the chain code of this rotated boundary, then let us see what is the chain code that will have. Again I assume that suppose this is my starting point to find out the chain code; here, the chain code will be 7,7,5,5, then 3,3, then 1,1. So, you find that the chain code of this rotated figure comes out to be like this. So, it is the same figure and now if you compare this chain code with this chain code, you find that the chain codes are totally different.

So, it is the same figure which is rotated and after rotation, I get totally different chain code representation. So, this clearly illustrates that the kind of chain code that we get in this particular case is not rotation invariant, whereas rotation invariance is our requirement. So, how we can obtain a chain code which is rotation invariant? So, for rotation invariant chain code, we don't go for the simple way of chain code extraction or chain code representation but what we go for is what is called a differential chain code, the differential chain code is something like this.

Let us consider this simple example. What we do is once I get the first order chain code that is chain code as given state from this procedure, then I take subsequent codes, I take 2 subsequent codes and try to find out that if from the first code I have to move to the second code, then how many rotations let us say in the anticlockwise direction I have to perform; so by doing that you find that in this particular case and because this is differential chain code, I have to take the difference, so instead of considering chain code as a string, you consider the chain code as a cycle. That means the first code has to be obtained by rotation from the last code in our chain code string.

So, by applying that you find that if I want to move from 0 to 0 because the previous code was 0, the next code is also 0; so the number of rotations that I have to perform is 0 rotations. So, when I move from this 0 to 0, the number of rotations that we have to perform is 0 rotations. When I move from this 0 to 6, so you come to this particular figure, I have to move from 0 to 6; so the number of rotations that I have to perform is 1,2,3,4,5,6, I have to perform 6 rotations, I represent this by 6.

When I go from 6 to 6, the number of rotations that I have to perform is 0. As we move from 6 to 4, this is 6 and this is 4; how many rotations that I have to perform? 1,2,3,4,5,6,6 rotations again. When I go from 4 to 4, the number of rotations is 0. When I move from 4 to 2; so this is 4, this is 2, again you find that I have to perform 6 rotations. As we move from 2 to 2, I have to



perform 0 rotations. As I move from 2 to say 0, I have to perform; this is 2, this is 0, so you can compute that again I have to do 6 rotations.

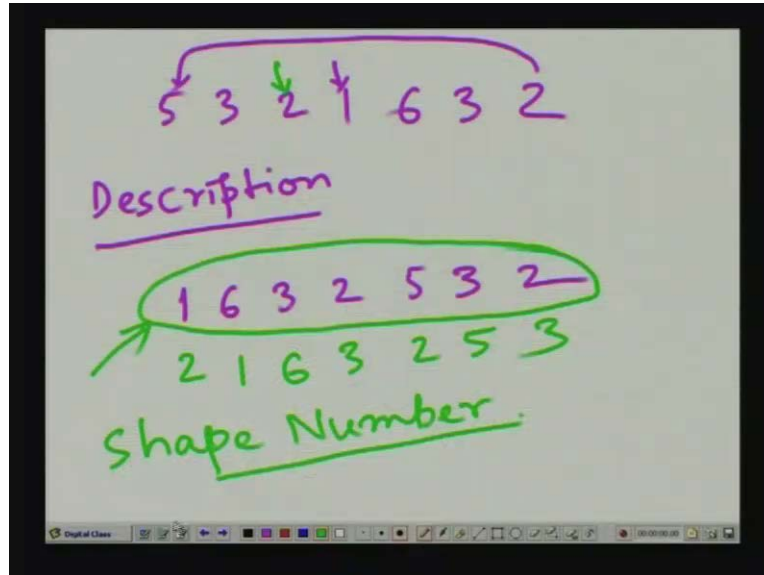
So, now this differential chain code, it becomes 0,0,6, then 0 again, then 6 again, then 0 again, then 6 again, then 0 again, then 6 again. Now, what happens in this particular case if the figure is rotated? Let us see that firstly I have to move from 7 to 7, so I have to perform 0 rotations. Let me use some other colour. So here, first I have to perform 0 rotations. Then I have to move from 7 to 5; so this is 7, this is 5. Again, in the anticlockwise direction you find that the number of rotations that I have to perform is 6 rotations. 5 to 5, again 0 rotations; 5 to 3 - this is 5, this is 3, again you can compute that you have to perform 6 rotations; 3 to 3 again 0; 3 to 1 - this is 3, this is 1, I have to calculate the number of rotations in the anticlockwise direction, so again you find that this will be 6 rotations; 1 to 1, 0 rotations; 1 to 7 - this is 1, this is 7, number of rotations in the anticlockwise directions again that will be 6.

So, the chain code in this particular case I get that is the differential chain code is 0,6,0,6,0,6,0,6. Here also, what I have obtained is 0,6,0,6,0,6,0,6. So, you find that instead of taking the direct chain code, if I go for the differential chain code, then whatever be the rotation, I get the same chain code representation and what I am doing is instead of considering this as a chain, I consider this as a cycle. Similarly, this has to be considered as a cycle.

So, this differential chain code, it will be translation invariant, it will be rotation invariant and the scaling, the scale invariance has to be done by choosing the grid spacing properly. So, I can have translation, rotation and scale invariant chain code representation, differential chain code representation of any given boundary.

So up to this, what we have done is a differential chain code representations of a boundary. But what we have said is we need a description so that the description which can be matched against the description present in the knowledge base. So, what is the kind of descriptor that we can obtain from this differential chain code representation?

(Refer Slide Time: 33:40)



So suppose, I have a chain code representation something like this; say 5,3,2,1,6,3,2 something like this and as I said that instead of considering this as a chain, I consider this as a cycle. So, a kind of descriptor that I can obtain from this particular differential chain code representation; so what I am saying is a differential chain code representation is that if I consider this as a cycle instead of a chain, then depending upon the starting point, this chain will be different. When I consider this as a chain, depending upon the starting point from where I start finding out the chain code, this chain will be different but the cycle will remain the same.

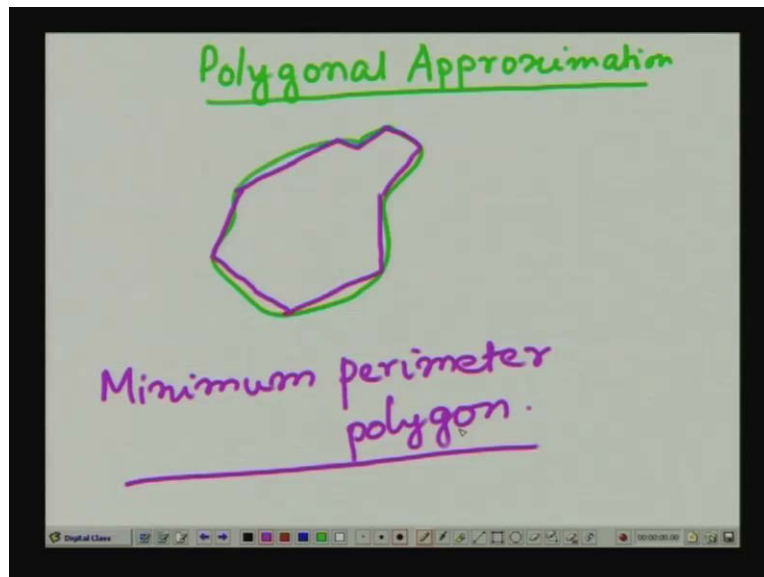
So, what I have to do is I have to redefine a starting point from this cycle so that from that the chain code that I get that will be say minimum, that will have say minimum numerical value. So, you find that from this particular code, if I take my starting point here; then what will be the numerical value of this? 1,6,3,2,5,3,2. For the original starting point from where this chain code, differential chain code has been obtained; the numerical value is 5,3,2,1,6,3,2. If I take the starting point here, say at 2, the corresponding numerical value will be 2,1,6,3,2,5,3.

So, you find that out of these 3 different options that we have considered, this is the one which gives you the minimum numerical value. So, what we have to do is once we get the differential chain code, considering the differential chain code to be a cycle; once it is a cycle that means the starting point is open, I can choose the starting point anywhere I like. So, what I do is I choose a starting point so that from that starting point if I open if I form the chain code, the differential chain code, the numerical value of the differential chain code will be minimum.

So, once I do that, in that case, I have a unique descriptor, unique description from the chain code representation or differential chain code representation which can be used for the recognition or understating purpose. So, this particular chain code that we have obtained having the minimum numerical value; this is what is called the shape number. So, this is what is called the shape number of the boundary of the object.

So, you can use this shape number to represent, to describe any given boundary and as we said earlier that these kinds of descriptors are used for shape understanding or shape recognition. If we want to find out or if we are interested in the reflectance property of the surface such as colour texture and so on, then this shape number is not helpful. So, along with the shape number, we have to use some of the descriptors that we will see in our subsequent lectures. So, this shape number is one of the boundary descriptors that we can use for object understanding or object recognition. There are many other boundary-based descriptors.

(Refer Slide Time: 37:35)



So, let us see one more boundary based descriptor which is by using polygonal approximation of the boundary. So, what is this polygonal approximation of the boundary? Suppose, we are given with an arbitrary boundary like this; what we want to do is this boundary has to be represented or approximated by a polygon following some criteria. So, there are numerous polygonal approximation techniques which have been reported in the literature. Obviously, we cannot discuss all of them but we will consider some simpler polygonal approximation techniques.

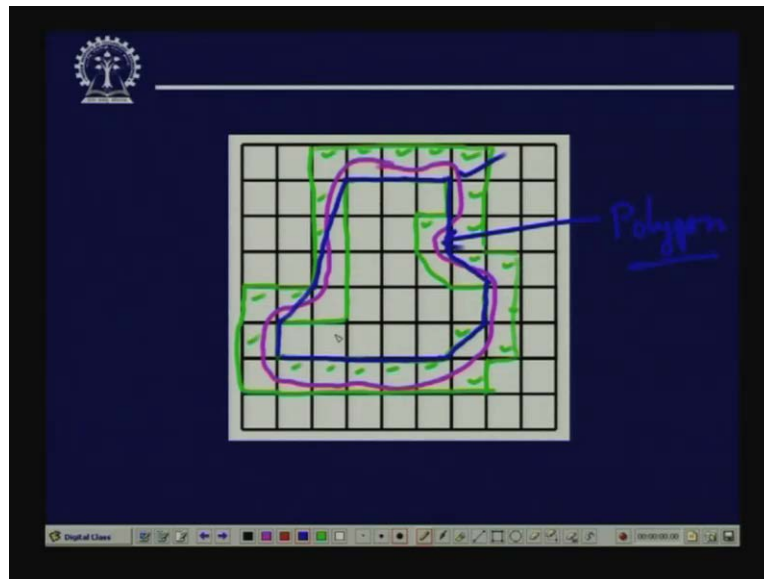
So, by polygonal approximation, what I mean is this given arbitrary boundary, we want to represent by a polygon. So, a polygon may be like this; say what I do is I represent the boundary by a polygon of this form. So, this is a polygonal approximation which I have shown in pink colour is a polygonal approximation of this given boundary. Now, let us see that how we can obtain such a kind of polygonal approximation? Now, one of the polygonal approximations that we will discuss first is what is called minimum perimeter polygon. This is called minimum perimeter polygon.

Here, the concept is something like this; you enclose this original boundary, the arbitrary boundary by a set of connected cells. So, when I enclose this boundary by a set of connected cells, then those connected cells define 2 sorts of restriction or 2 sorts of valves; one is the external wall, the other one is the internal wall. And once I enclose this boundary by such

connected cells, then if I assume that this boundary is composed of say rubber code and let the rubber code contact itself within that within that set of cells.

So, if we allow the rubber code to contract, following the constraints that we have put by using the connected cells that means by specifying that what is the outer wall and what is the inner wall; then within that restriction, this rubber code will try to fit itself such that its perimeter will be minimum. So, that is a kind of polygon that we get which is called a minimum perimeter polygon. Now, let us illustrate this with an example.

(Refer Slide Time: 40:42)

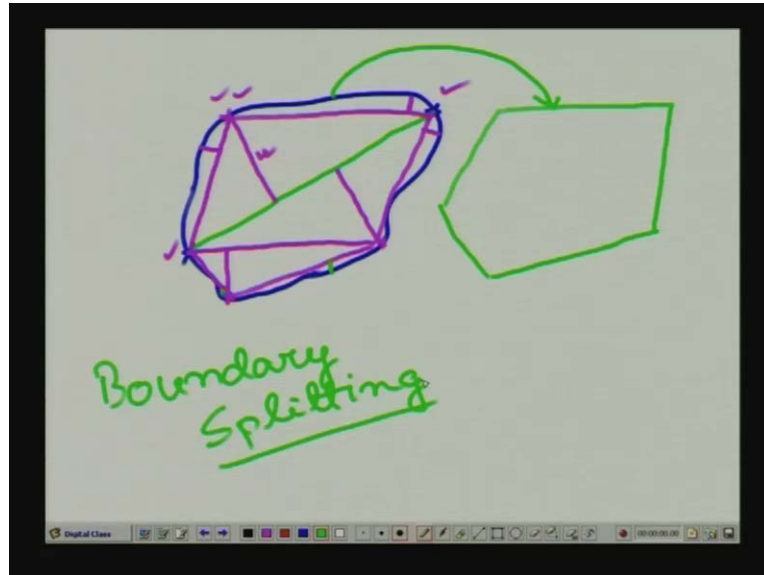


So suppose here, I have a boundary say something like this and when I do that you find that I have a set of polygons which enclose this boundary, a **connected** set of connected cells which enclose this particular boundary. Now, what are those connected cells? This is 1 cell, this is 1 cell, this is another cell, this is another cell, this is another cell, this is another one, this is another one, this is another one, this is another one, similarly this, this, this, this. So, these are the connected cells which actually enclose this particular boundary.

And once we have that, you find that I have a set of inner walls. So, in this particular case, the set of inner walls is this. So, these are the inner walls and these are the outer walls. So, given this kind of situation, if I now allow this boundary to contract; what this boundary will do is it will fit itself within these cells, these connected cells following the restriction given by the inner wall and the outer wall and possibly, the kind of boundary representation, the polygonal representation that I will get in this particular case will be something like this as given by this blue colour.

So, this blue coloured one will be a polygon representation of this given boundary which is in the pink colour. So, this is a kind of polygonal approximation that we get which we term as **minimum polygon** minimum perimeter polygon. Now, we can have other kind of polygonal approximation as well. We can have a polygonal approximation by splitting the boundary.

(Refer Slide Time: 43:51)



Here, the concept is something like this; say suppose I have a boundary of arbitrary shape, so what I do is somehow following some criteria, I choose 2 points on this boundary and split this original boundary in 2 halves by joining these 2 points by a straight line. So, you find that this boundary has now been divided into 2 different codes; one on the upper side and one on the lower side. Then what I do is for each of these boundary segments; as the boundary has been divided into 2 different boundary segments, for each of the boundary segments what I do is I find out a point lying on every boundary segment whose distance from the line joining the end points is maximum.

So, in this particular case, you find that may be on the left side, this is a point whose perpendicular distance from the line joining these 2 end points, these are the end points; these 2 end points are maximum. Similarly in this case, this is a point on this boundary whose perpendicular distance from the line joining these 2 end points is maximum. Now, what I do is I put a threshold on this perpendicular distance. So, if I find that this maximum perpendicular distance is more than the threshold, then again I split the corresponding boundary segment.

So, in this particular case, suppose this one, this particular distance is more than the threshold; so what I do is I immediately break this upper segment of the boundary into 2 sub segments. **One segment** for one segment, this is the starting point and this is the ending point and for the other segment, this is the starting point, this is the ending point. Again I do the similar operation; I join a line between these 2 end points, I join a line between these 2 end points, again I find out what is the maximum distance of a point on each of the segments from the line joining the end points. Again, I compare these maximum distances against the threshold.

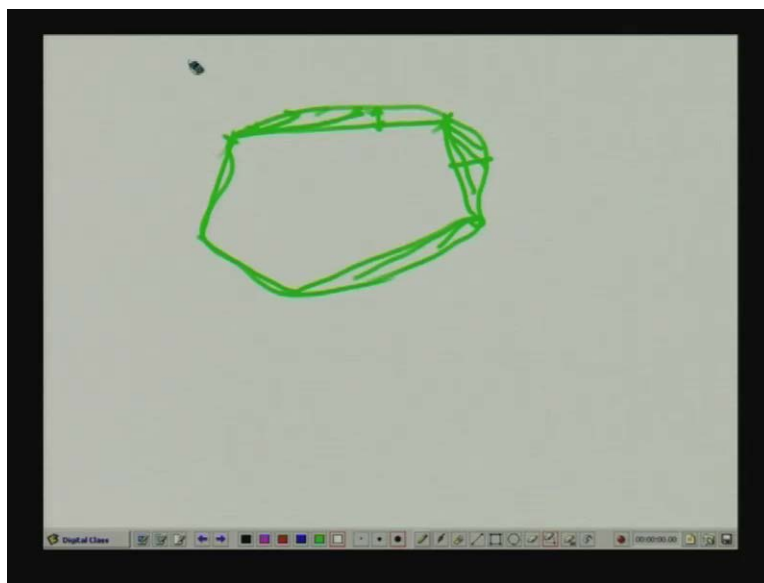
Now, in this case, it may so happen that this maximum distance is less than the threshold. So, if this maximum distance is less than the threshold, I do not break them further. Similarly, in this particular case; I join here, I join here, I find out what is the maximum distance, I find out what is the maximum distance.

Now, suppose this maximum distance is less than the threshold but this maximum distance is greater than the threshold; so what I will do is I will again segment, break this particular boundary segment into 2 parts, one will be represented by this and the other one will be represented by this. Now, at this point, suppose these perpendicular distances are less than the threshold, so I can stop my algorithm there.

So, what I get is given this arbitrary boundary, this arbitrary boundary has now been represented by a polygon which is this. So, this polygon shape that I get is this one. So, this is just a replica of this pink coloured polygon that we have drawn within this arbitrary boundary. So, I can get a polygonal representation of this particular boundary in this manner. This is what is called a splitting technique or boundary splitting technique because at every stage, we are splitting the boundary into 2 halves depending upon whether the maximum distance of a point lying on the boundary from the line joining the 2 end points of the boundary segment is more than the threshold or less than the threshold.

So, if it is more than the threshold, we again sub divide that particular segment of the boundary into 2 halves. If it is less than the threshold, then we do not go for sub division any more. So, based on this, we have also got a polygonal approximation, a polygonal representation of this arbitrary given boundary. So, there are various such ways for polygonal approximation.

(Refer Slide Time: 48:40)

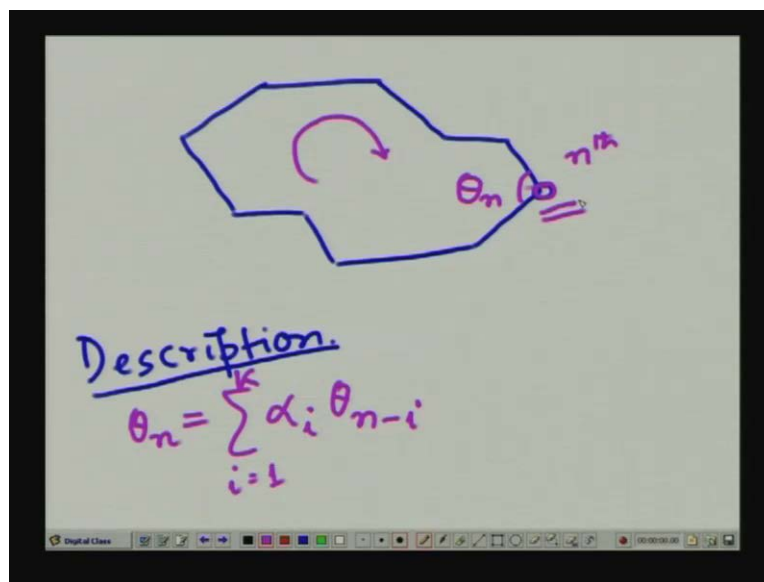


Say for example, one of the polygonal approximation techniques may be something like this; say start from a particular point, then you keep on then you trace other points lying on the boundary and go on representing these 2 end points by a straight line something like this and for every such position, you find out what is the maximum distance of a point lying on the boundary from this straight line segment. If this maximum distance which is the threshold, then I consider this as my next starting point.

So, I start the same operation from here again, something like this. Again, I find that this maximum distance touches the threshold; so this becomes my starting point, I start from here, so like this. So, there are various ways in which a polygonal approximation of a given boundary can be done and a number of such techniques have been reported in the literature. We will not discuss all of them but the essence is once I have a polygonal representation of a boundary, the polygon captures the basic essence of the shape of the boundary that is what we want to generate our descriptor.

Now, the question is once I have a polygonal approximation of an arbitrary closed boundary; what are the kinds of descriptors that we can obtain from this polygonal approximation?

(Refer Slide Time: 50:12)



So, what we are interested in is say we have a polygonal approximation of a boundary, so I have any such polygon, say something like this. Now, what is the descriptor that I can generate from this polygonal representation? So, I have to find out a description of this polygon. Now, there is a technique where to find out this descriptor, people have gone for an autoregressive model. So, in case of an autoregressive model, what is done is suppose I consider any of the vertices in this polygon, so I consider this particular vertex; now this vertex if I call it say n'th vertex, it has a subtended angle say  $\theta_n$ . Now, if I trace this polygon in say clockwise direction, suppose I trace it in the clockwise direction, then the given autoregressive model, what is assumed is this angle  $\theta_n$  is represented by a linear combination of k number of previous angles.

So, what I do is in this case, this  $\theta_n$  that is the angle subtended at the n'th vertex is represented by a linear combination of k number of previous angles. So, I represent it as  $\alpha_i$ ; say  $\theta_{n-i}$ , take the summation for i equal to 1 to k. So, I have an autoregressive model like this where I consider k number of previous angles to represent, the linear combination of which represents this n'th angle.

(Refer Slide Time: 52:30)

$$\theta_n = \alpha_k \theta_{n-k} + \dots + \alpha_1 \theta_{n-1}$$

$\Downarrow$   
 $k \rightarrow \text{unknowns.}$

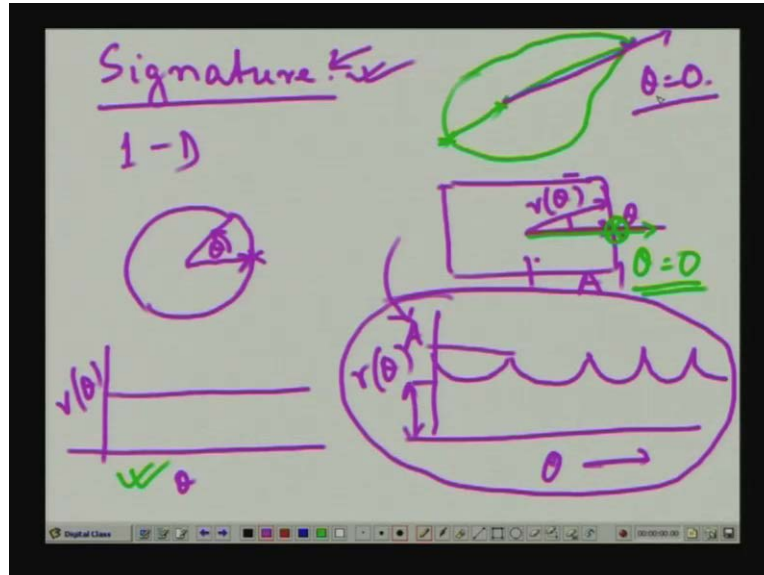
$\alpha_i$   
 $i = 1, 2, \dots, k.$

So, what I have in this case is if I expand this equation, what I have is theta n is equal to say alpha k theta n minus k plus continuing like this, say alpha 1 theta n minus 1. So, you find that here I have k number of unknowns; k number of alpha k is which are the coefficient of this linear equation, so I have k number of unknowns. To solve for this, I have to generate k number of equations; so you take k number of such vertices which are represented by such linear equations.

So, for each of these k number of vertices, I can generate k number of linear equations and by solving those k number of linear equations, I can solve for alpha i where i varies from 1, 2, k and this set of alpha i can be used as a descriptor which can be used to describe a given polygonal shape or because the polygon is an approximate representation of our arbitrary boundary. So, this can be used as a descriptor of the boundary. So, these are the 2 types of descriptors that we can use. We can also use another descriptor, another kind of descriptor which is known as signature.



(Refer Slide Time: 54:08)



Now, what is this signature? This signature is some 1-dimensional mapping of a boundary. So, as we have seen, the boundary is nothing but a 2-dimensional closed curve. So, if it is an object ideally, I should get a closed curve and that closed curve is a 2-dimensional closed curve. The signature represents, signature is a 1-dimensional mapping of this closed curve. So, how do we get it? This is represented by a function. Say for example, I have so let us have we have a circle like this and I find out the centroid of the circle. I find out what is the distance of different points on this closed curve on this boundary from this centroid and if I trace this boundary in say anticlockwise direction, then what I have is from this starting point, I have a displacement angular displacement theta.

So, all these distance values, I represent it I represent as a function of theta. So, you find that if it is a circle, in that case, the distance  $r$  theta versus theta it will be a straight line which is parallel to the horizontal axis. Whereas, if I have a boundary which is say square or a rectangle like this and this is the centroid, I compute  $r$  theta and this is my angle theta; so if I plot  $r$  theta versus theta in this particular case, this is theta and this is  $r$  theta, then what I have is a representation of this form, like this.

So, what I am doing is this 2-dimensional boundary is converted to a 1-dimensional function and this 1-dimensional function that is  $r$  theta versus theta is what is known as signature of this boundary. Now obviously, in this particular case, as we said that the descriptors or the representations have to be translation, rotation and scale invariant; in this case, you find that this particular representation is translation invariant but it is not scale invariant because if this particular dimension, from centroid to this, it has a value equal to  $A$ , then the maximum value that we will get is also  $A$ .

So, to make it scaled invariant, what we can do is we can normalize this particular signature between 0 and 1 by dividing it and by giving a dizzy offset equal to the minimum value and

dividing by the maximum value. So, we can have a normalization between 0 and 1 that will make this signature scale invariant.

Now, the signature is already translation invariant. But if we want to make it rotation invariant, then what we have to do is I have to define or identify a starting point, from where to start scanning this boundary or what is the direction which will correspond to say theta equal to 0. So, to obtain this, what I have to do is I have to get a unique point on the boundary if it is obtainable. Say for example, in case of a circle, I cannot find out a unique point. So, whatever be my starting point or whichever location corresponds to theta equal to 0, I will always get this particular signature.

But in such cases, where the boundary is not uniform, it is not of circular shape; it is possible to have a unique point on the boundary and what we can do is we can choose 2 furthest points, I can choose the pair of points on the boundary whose distance is maximum and out of that whichever is at a larger distance from the centroid, that I can use as a starting point.

Say for example, if I have a shape something like this; so you find that these are the 2 points which are furthest and the centroid is somewhere here. Say point on this boundary which is furthest from the centroid, so I can start my scanning. So, this may be my direction of theta equal to 0 or similarly I can have a similar I can choose such a point by using the principal I can access. So, I can choose 2 points on the boundary which lies on the principal I can access and out of these 2 whichever is furthest from the centroid, I can choose that my starting point or the reference theta equal to 0 and using this I can have a 1-dimensional mapping of a boundary which give us a signature.

So with this, we stop our lecture today. We will continue this topic in our next lecture.

Thank you.