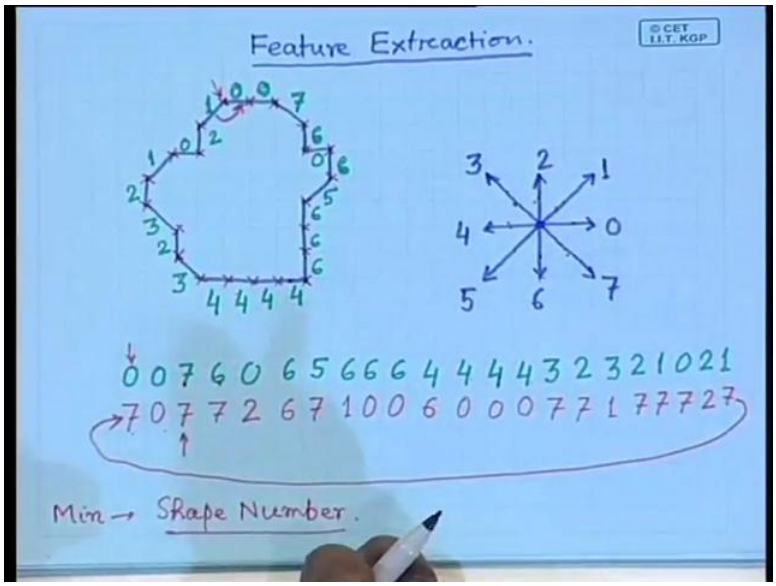
Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian institute of Technology, Kharagpur

Lecture - 3
Feature Extraction – II

Good morning. So, today we will continue with our discussion on feature extraction. In the last class we have started discussion on chain code and we have said that to make this chain code rotation in variant, we have to make some use something called Differential chain code. So, today as some of you have requested, I will just repeat the differential chain code formation, before I move to other ways of feature extraction. So, when I talk about chain code, let us have that let us assume that we have some sort of boundary pixels represented by a boundary something like this.

(Refer Slide Time: 00:59)



So, let us take a boundary of this form and to discuss about this chain code, we will assume the chain code which is formed by 8 neighbours. So, the convention that we used in the last class was like this, so these are the different directions of the line segments, which is used to piece wise linear approximate the given arbitrary boundary and this direction we took as 0, this direction we took as 1, this direction we took as 2, 3, 4, 5, 6 and 7. So, these are the 8 different directions which are used to form the chain code. So, in this figure if I assume that I start the chain code operation from this point.

So, this is the starting point, then first movement I will have, so these are the points on my boundary actually, these are the boundary points, wherever I put across, so this becomes the set of boundary points. So, when I form this chain code starting from the first location that is this one, my initial direction of the line segment is from left to right, that is towards the east and that direction is marked as 0, so I put 0 over here. Similarly, the second line segment, that is also in the direction of 0, so I also put 0 over here. The third line segment it is in the south east direction and south east is marked as 7, so I put 7 over here.

The next movement is in the south direction which is marked as 6. The next one is again in the direction of east, so which is marked as 0; next direction is gain in south, so it is marked as 6. So, this way this way if you continue you find that your chain code will be represented like this. So, starting from my starting point, the inter chain code becomes 0 0 7 6 0 6 5 6 6 6 4 4 4 4 3 2 3 2 1 0 2, then followed by 1. So, this becomes the chain code which represents a given boundary like this. Now, as I said that if the figure is rotated, in that case this chain code will be entirely different.

So, for example, instead of putting the figure like this, if I simply rotate the figure by 90 degree, if I make the figure of this form, in that case earlier one which was 0, that will now become this one which will be 7. So, the inter chain code representation which we had having the boundary position like this, if I rotate the boundary the inter chain code will be different. So, using this chain code I cannot recognize a shape which is rotated by an angle Theta. So, what I need to do is I have to modify this chain code so that it becomes rotationally invariant.

So, for doing that what I said is instead of using this chain code, what I will do is to move from one segment to another segment, how many steps of rotation that I have to perform in my coding scheme that is in this scheme I will simply put that in my chain code. So, while doing this, let me start this way initially I will skip the first code that I have generated, because though I know that the first code was 0, but I do not know that from previous code to this code, when I move how many steps of rotation I have to make, because I do not know what is the previous code.

So, here, but I know that this direction is 0, next direction is also 0, so I know that from to move from 0 to 0 how many steps of rotation I have to make? So, in this case, because

the previous direction is also 0, the next direction is also 0, so the number of steps of rotation will be 0, so I will put this as 0 only. The next one from 0 to 7, here I have to find out how many steps of rotation I have to perform? So, this is 0 and this is 7 and if I compute the steps of rotation in the anti clockwise direction, then you find that I have to make 1 2 3 4 5 6 7, 7 rotations, so this will also be 7.

To generate the next code I know this was 7, the next one is 6, so how many steps of rotation I have to find, I have to compute? So, this is 7, this is 6, so again I have to compute 7 steps of rotation, so this will also be coded as 7. Then from 6 to 0, this is 6 this is 0, I am following anti clockwise rotation, so I have to perform two steps of rotation, so 1 and 2, so this will be coded as 2. Next, is from 0 to 6 I have to perform 6 steps of rotation, next is from 6 to 5 I have to perform 7 steps of rotation, next is from 5 to 6 I have to perform only 1 step of rotation, next is 6 to 6. So, the number of steps of rotation is 0, 6 to 6 again number of steps of rotation is again 0, then 6 to 4, so this is 6 and this is 4 which is in the west direction.

So, here you have to you find that 1 2 3 4 5 and 6 I have to perform 6 steps of rotation, that is why I put 6 over here. Then 4 to 4 again 0 steps of rotation, 4 to 4, 0 steps of rotation 4 to 4, 0 steps of rotation, then 4 to 3, 4 this is 3, so if you compute in the anti clockwise direction, you find that I have to perform 7 steps of rotation. Then 3 to 2 again I have to perform 7 steps of rotation, then 2 to 3 following anti clockwise direction I have to perform only 1 step of rotation, then 3 to 2 again 7 steps of rotation, 2 to 1 again 7 steps of rotation, 1 to 0 again 7 steps of rotation, 0 to 2 I have to perform 2 steps of rotation, then 2 to 1 I have to perform 7 steps of rotation.

Now, I find that is 1 to 0, because the last code was 1 and the first code was 0, so when I move from this segment to this segment, from here to here rotation is 1 to 0. So, this is 1, this is 0 number of steps of rotation that I have to perform is 7. So, this completes my differential chain code. Now, again this differential chain code as I have formed, this is still dependent on the starting position. So, instead of taking the starting position as this point if I take the starting position as this point, then my chain code will start from this location. So, instead of 7 0 7 7 2 6 7 1 0 0 6 0 0 0 7 7 1 7 7 7 2 7 it will be 7 7 2 6 7 1 0 0 and this will continue and it will end with 7 0.

So, though I have, though I have made this code as independent of rotation, but still it is dependent on the starting position, the starting pixel that we considered to form this chain code. So, to make it starting position independent, what I have to do is, I have to consider this as a cycle, so it is not a open chain, but it is a cycle. So, if I break the cycle at any point what I make is a chain, so from this cycle I have to come to a chain and when I come to a chain, what I have to do is, I have to identify a point within a cycle where if I break I will get either a maximum number or a minimum number.

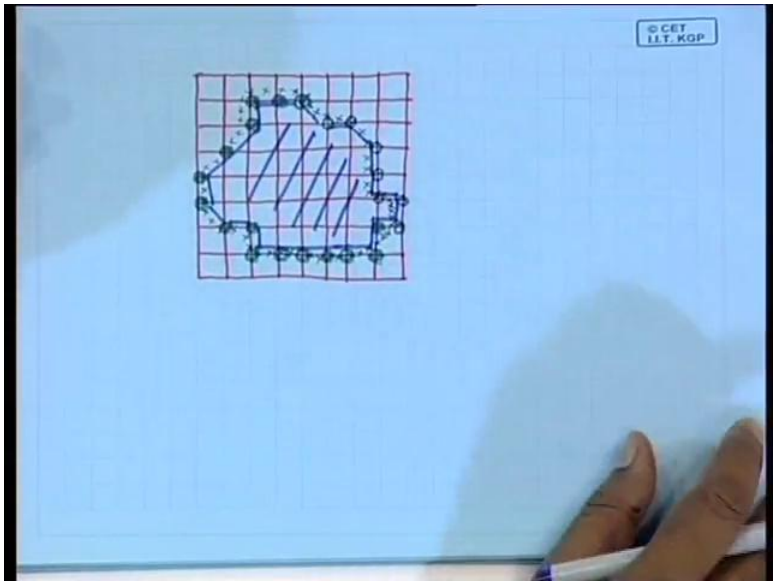
So, if I follow I will always break it in such a way that I will always get a maximum numerical value I get unique code. If I follow the other way that wherever I break I will get the minimum numerical value, I will also get a unique chain code. So, if I follow the concept that I will break the chain code in such a way that I will get the minimum numerical value, then the number that I get is what is called a shape number. So, is the concept of differential chain code clear now? So, from the differential chain code what we initially form is a cycle not a chain and once I have this cycle, I decide a point where the cycle can be broken, so that I can get a chain code having either maximum numerical value or minimum numerical value.

So, if I follow the concept of maximum numerical value, then always I have to form the maximum numerical value, if I follow the concept of minimum numerical value, always I have to form the minimum numerical value. Both of them will serve our purpose in terms of recognition, whether we use the maximum numerical value or minimum numerical value. Now, the next point comes that can we make it scale independent, we have made it rotation invariant, rotation independent by taking the differential chain code, and can we make it scale independent as well? Obviously the chain code is translational invariant, because the shape is here or the shape is here I will get the same code, so it is independent of translation.

By taking the differential chain code, we have made it independent of rotation. Now, whether can we, whether we can make it independent of scaling as well? That means, if the size of the boundary changes, if it becomes more or less, whether or this uniqueness of the chain code still can be retained. So, let us see that how we can make this chain code scale independent. So, to do this, let us assume that we have a set of boundary points something like this, so initially the set of boundary points that we have, the

distance between two consecutive boundary points is just 1 pixel, they are at a distance of 1 pixel apart.

(Refer Slide Time: 14:30)



So, if I have a boundary something like this, let us take an arbitrary boundary. So, assume that these boundary pixels are 1 pixel apart, so all this crosses are my boundary pixels. So, these points are nothing but only regularly spaced grid, where the grid spacing is just 1 pixel. So, what I will do is I will embed this set of boundary pixels on another grid, where the grid's spacing is much more, much more than 1 pixel. So, if I put the grid spacing like this, so these are the grids of larger spacing in which I embed this boundary pixels. Now, once I have this grid, then what I will do is; these boundary pixels will be associated with one of the grid crossings of this larger spacing, whichever is nearest to it.

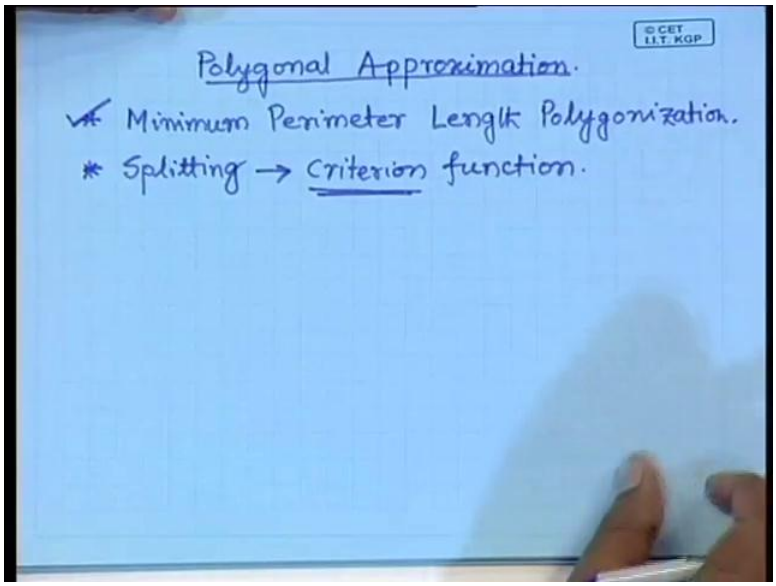
So, if I do that you find that this point will be associated with this, this point will also be associated with this, and this point will also be associated with this. So, the grid crossing that I will have is this one. Similarly, this is another grid's crossing, this is another grid's crossing to which some of the pixels will be associated, this will be another grid crossing, this will be another grid crossing, this will be another grid crossing, maybe this will be another grid crossing, this one will be another one, this one will be another, this will be another one, and this. So, these are the grid points of this larger grid in which the points will be associated.

Now, I can form up approximate boundary of the boundary represented by my initial set of boundary pixels by these grid points, so my boundary now will be this. So, we find that initially I had large number of pixel boundary, boundary pixels which are now scaled down to a lower number of boundary pixels and what I get is an approximate representation of the original boundary. So, now if I use this approximate boundary to form the chain code, then to some extent I take care of scaling, because as if I have represented the same boundary by less number of line segments.

So, like this if I increase the spacing of these grids, the number of line segments in my chain code will still be, will still be lesser, if I reduce the grid spacing, in that case number of line segments in my chain code will be larger. Of course, maximum that I can have is my initial set of boundary points, unless I go for super resolution concept, where I can interpolate a point between two boundary points. So, that is a concept of super resolution where I can still increase or improve the resolution of the given image, is that ok?

So, these are the different forms of chain code that we can form and this chain code can also be used for recognition of objects, where this recognition is based on shape, because this chain code is nothing but a shape descriptor, where this shape is obtained using only the boundary information. Because we have not any used any point which is within the shape, I have not used any information which is inside the shape, I have used only the information of the boundary. So, this chain code can be used for recognition of the objects where this recognition will simply be based on the boundary information. Now, the other type of descriptor that can be used for recognition purpose is based on polygonal approximation.

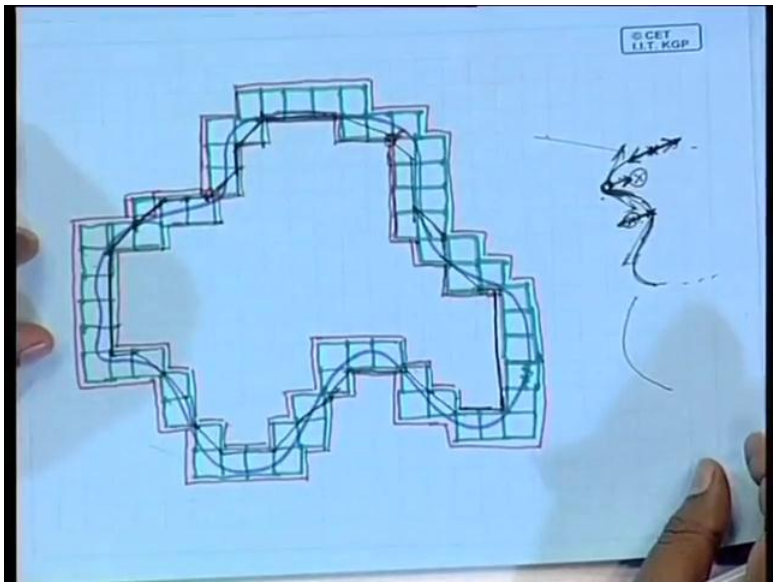
(Refer Slide Time: 21:03)



So, chain code also you can say that it is some sort of polygonal approximation, because the arbitrary boundary is represented by linear line segments, piece wise linear line segments, so it is also nothing but polygonal approximation. But when we talk about polygonal approximation, what we do is we do something else to represent an arbitrary boundary by a polygon, so how do we do that? When I go for polygonal approximation there are different types of approaches which can be used to represent a boundary by a polygon. So, out of this we will discuss here two techniques, one of the techniques is called minimum perimeter length, polygonization.

The other kind of polygonal approximation technique is based on splitting, splitting of the boundary. Now, when I go for splitting, obviously there has to be different types of criteria based on which an arbitrary boundary has to be split into number of boundary segments, where every boundary segment will be represented by a linear segment or a straight line segment. So, when we talk about splitting, normally we have a criteria function associated with the splitting technique and based on the criteria function I can have different types of splitting algorithms. So, we will discuss them one after another, so firstly let me take this minimum perimeter length polygonization approach, what I mean by that?

(Refer Slide Time: 23:42)



So, to do this let me again take a boundary of arbitrary shape, so I have a boundary something like this or let me make it a bit more complicated. So, I have a boundary something like this, so to go for this minimum perimeter length polygonization, what we will do is, we will embed this boundary into a set of concatenated cells. So, what are those set of concatenated cells? So, the cells are, I can use this set of concatenated cells, it is nothing but a concept of grid. So, this is the set of uncatenated cells in which this boundary will be placed and this is nothing but if I put the same concept of grids over here and then I have to identify that which of the cells, actually enclose this particular boundary.

So, once I have this set of uncatenated cells, and then you find that I can define something like an inner boundary and outer boundary, this cell will also be there. So, my inner boundary will be this one, this is the inner boundary and this one is the outer boundary or inner wall and outer wall. So, this cell will also be included, because so the outer wall will be like this. So, I have an inner wall and outer wall. Now, if I assume this boundary to be a rubber cord, a closed rubber cord which is kept within this inner wall and outer wall.

Then you know that a rubber cord under tension which is kept under a constant place like this, it always try to attain the minimum length. So, as a result this cord will try to shrink in whichever ways possible and it will try to attain the minimum perimeter. So, you find

that this part of the cord will try to be contracted and come towards this direction; similarly this part of the cord will try to be contracted and move towards this direction. So, likewise if you find out at each and every point on this boundary, how the cord will move, whether it will move in the outer direction or it will move in the inner direction, until and unless it reaches the limiting points.

So, when it tries to move in the inner direction at this point the limiting point is this, it cannot move further because this is the wall. Similarly, over here when this point tries to move outer, in the outer direction this is the limiting point, it cannot move beyond this. So, if you continue this way you will find that I will have a boundary representation of this particular boundary which is given, which will be given approximate representation something like this. So, this is my approximate representation of this boundary when this cord tries to shrink, because it will try to move both in the outward direction, as well as in the inward direction wherever is possible, until and unless the cord reaches the limiting point. So, what I get is a polygonal representation of this boundary.

Now, so far as the concept is concerned it is very clear, but how do we implement it? For implementation, what you have to do is you find that at every point, either the cord is straight or the cord is concaved or the cord is convex. So, if I have a boundary position something like this, where there is continuing in this direction, over here the cord is convex, over here the cord is concave, over here the cord is convex and I can have a linear portion something like this.

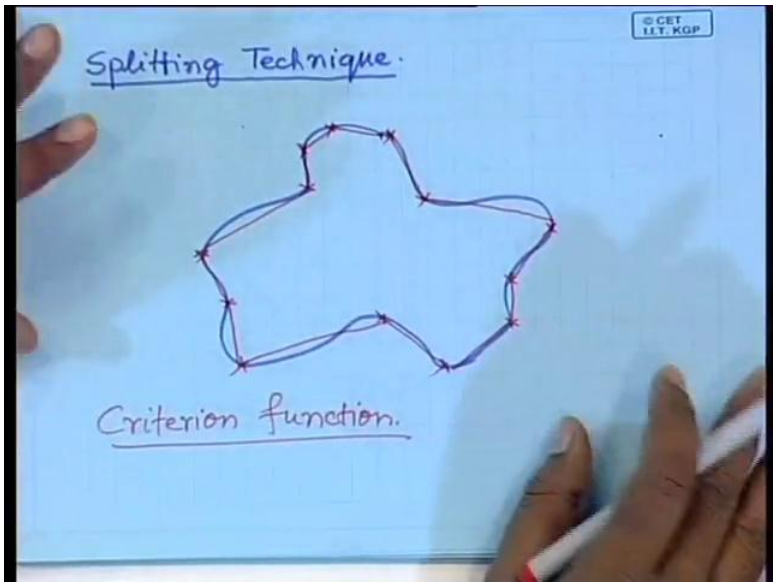
So, if I consider any point on this cord segment, if the point lies on the linear part, then at this point I will have tension in this direction, as well as in this direction, in two opposite directions. That means, if this point has to move, it has to move only along the cord, because I have tension either in this direction or in this direction in two opposite directions.

So, if I assume that the tension in both the directions are same, this point will not move at all, however this point will move due to the influence of the movement of this point. Here it is in the convex part and I have tension in this direction and I have tension in this direction. So, as a result the resulting tension which is acting at this point is in this particular direction. So, this point will try to move in this direction and here I have to find out that what is the limiting point, limiting points are given by these inner walls and

outer walls. So, I have to identify that what are the limiting points and I have to allow this point to move until and unless it reaches a limiting points.

Similarly, a point at this location it has tension in this direction and it also has tension in this direction, so the resulting tension is in the direction like this. So, under the influence of this resulting force this point will also try to move in this direction and it will move until and unless it reaches a limiting point somewhere over here. So, when this point comes up to this position I have to stop movement of this point. So, that is how I can find out that what are the limiting points up to which this cord can move while it tries to attain the minimum perimeter and those limiting points in the inner wall or in the inner wall gives you the vertices of the polygon. So, if you represent, if you simply connect those vertices by straight line segments you get a polygonal approximation of this arbitrary boundary. So, this is what minimum perimeter length polygonization is. Any question? No. So, next let us see that what is the other type of polygonization that is splitting technique.

(Refer Slide Time: 36:39)

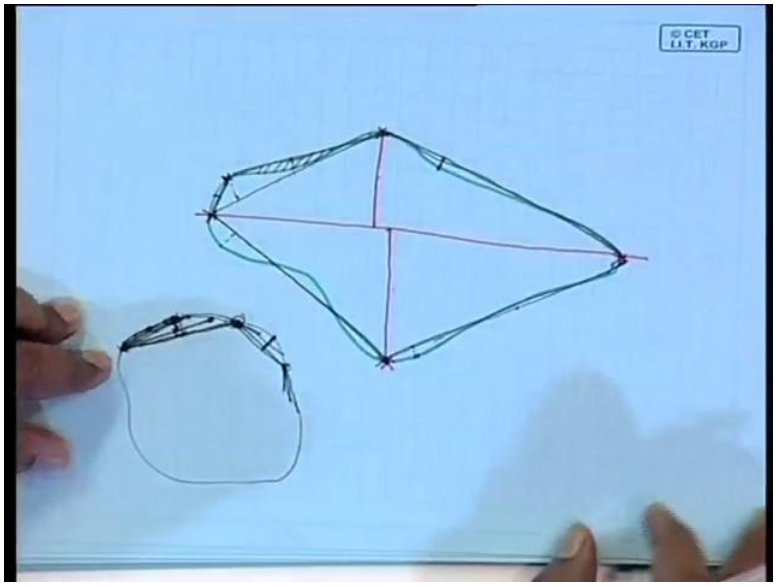


In splitting technique again if I have an arbitrary boundary something like this, what I have to identify is, I have to identify a set of points on this arbitrary boundary along which this boundary can be split. These points at which the boundary can be split into different boundary segments they become the vertices of the polygon. So, if somehow I select this set of points along which this boundary can be split then the polygonal

approximation of this arbitrary boundary is this one. Now, the criteria function as I said comes into picture which helps to decide that what are the points on this arbitrary shaped boundary are to be taken along which this boundary can be split.

So, there are different criteria functions, the different functions gives you different result. The results may not be identical, but they are more or less similar and help in all practical applications of this polygonization technique. So, let us talk about, say one or two such criteria functions based approach.

(Refer Slide Time: 38:50)



So, let us take a boundary something like this, so one of the criteria function says that initially you decide two points on the boundary along which this boundary can be split. So, once I decide two points on the boundary then the boundary splitted into two sub boundaries or two segments of the boundary. So, first let us assume that somehow we have been able to select one point over here and one point over here. So, these are the two points along which the boundary can be split, so once I select these two points, then I have one boundary segment which is the upper part of the boundary and I have another part of the segment which is the lower part of the boundary.

Then the criteria function says that once I select these two points, I can have a straight line passing through these two points, so this straight line divides the boundary into two halves. Now, for each of the halves what I do is, I find out the maximum distance perpendicular distance of a boundary points from this straight line. So, on this half you

may find that this will be a point where the perpendicular distance from the straight line joining these two split points is maximum. Similarly, on this half this may be a point from where the perpendicular distance to this straight line may be maximum. Then the criteria function says that if this maximum distance is greater than its threshold, then this point I have to take as the next splitting point.

So, suppose in this case both these distances are greater than threshold, so this will be one split point, this will be another split point, but I have to take it one by one, because I have to write an algorithms. So, I take the maximum distance first, so I assume that this is my next split point. So, once I select the next split point, in that case I have divided this boundary in three boundary segments. So, once I have these three split points, again I can have straight lines passing through pair wise split points, so this is one straight line and this is another straight line.

Again I follow the same concept, same criteria that for each of these straight lines I find out a point on the boundary between these two split points whose distance from this straight line segment is maximum. So, possibly I will get a point somewhere over here, whose distance is maximum. Now, suppose this distance is less than the threshold that is pre defined, so if this distance, because this is maximum, so all other points on this boundary will have a distance from this line segment which is less than that. So, if this maximum distance is less than threshold, then all other points on this boundary segment is within a distance which is less than the threshold. So, because it is less than the threshold I do not have to try to split this boundary anymore.

Similarly, over here I find out the maximum distance, if the maximum distance is less than the threshold, I do not have to split this boundary anymore. However, for this boundary segment, this was my distance and suppose, this distance is greater than the threshold, so this becomes my fourth splitting point. Once I have this fourth splitting point, again I join the straight line segments, the straight lines to this splitting point. Now, this is the maximum distance of this particular segment from this straight line and this is the maximum distance of this segment from this straight line.

Suppose, this distance is less than the threshold, so this part of the segment is not to be splitted anymore. If I assume that this is greater than the threshold, then this will be my fifth splitting point. So, once I have this fifth splitting point, again I will have line joining

these vertices, over here this is the maximum distance and over here this is the maximum distance. Now, if these distances are less than the threshold, then neither I have to split this nor I have to split this. So, my final polygon now becomes this one, so this is the polygonal representation of this boundary that I have obtained.

Similarly, there are many other criteria functions which can be used for polygonal approximation. One of the criteria functions tries to find out what is the area of the portion which is enclosed between the line, this straight line approximation and the original boundary and that criteria function tries to minimize this area.

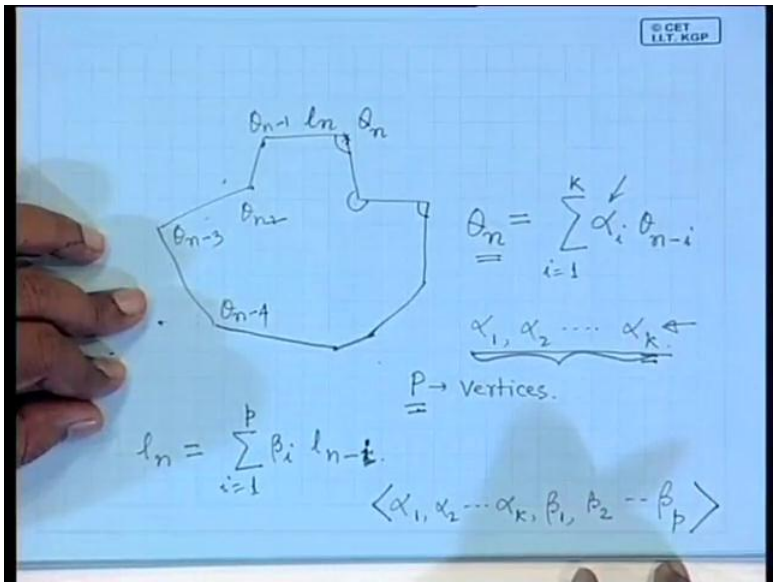
The other criteria function can be something like this that given a boundary segment of this form, I start with a from an initial point, then go on considering the points across the boundary and try to approximate them by straight lines like this. Every time I approximate them by straight line, again I have a concept of threshold that is I find out the maximum distance of a boundary point from this particular point.

When it exceeds the threshold then my starting point becomes this, so suppose I started from here, initially I have considered this straight line to approximate this boundary segment, here the distance was less than the threshold, so this is not my vertex, I go to the next point. Again find out what is the maximum distance between this line segment and a point on the boundary. Suppose, this is also less than the threshold I consider the next point, join, try to approximate this boundary segment by this straight line. Now, suppose this maximum distance is greater than the threshold or has been equal to the threshold.

So, the moment I get that I decide that this line segment cannot be expanded further. So, this part of the boundary, this line segment is represented by this line segment. Now, I start the same operation taking this as my starting point, so I have decided about one line segment. Now, I perform the same operation over here and suppose after I come to this location, this distance becomes greater than or equal to the threshold.

So, this becomes my next line segment. Then I start the same operation starting from this point, it continues like this. So, there are different criteria functions which can be used for polygonal approximation of an arbitrary boundary. So, following any of them what I get is, given an arbitrary boundary I get a polygonal approximation of the arbitrary boundary.

(Refer Slide Time: 47:03)



So, suppose I have a polygon something like this, which represents arbitrary boundary. Now, this polygon by itself is not much helpful for recognition purpose, so what I have is, I have an arbitrary boundary and I have been able to approximate that arbitrary boundary by linear segments, by piece wise linear segments and the combination or the collection of those linear piece wise linear segments give me a polygonal approximation of the boundary. But for recognition purpose, finally I have to find out the feature vector or a set of features, using which I have to go for recognition.

So, how do you generate that set of feature vectors, the set of features or the feature vector from here? So, the one approach to generate the feature vector is something like this, once I have this polygonal approximation I can always find out what is the angle, inner angle subtended at each of the vertices of the polygon. So, suppose I take the nth the angle subtended at the nth vertex to be Theta n. So, once I decide this, I can have an auto regressive model, so that this nth angle is represented by a linear combination of k number of previous angles or effectively I will have a linear regression equation which is given by this, Theta n is equal to summation Alpha i Theta n minus i, where this i will vary from 1 to some k.

So, what I am doing? This angle is represented by some alpha 1 times Theta n minus 1 plus Alpha 2 times Theta n minus 2 plus Alpha 3 times Theta n minus 3 plus Alpha 4 times Theta n minus 4 and so on. So, if I take k number of such previous angles to form

this auto regressive model, this is what kth order auto regression is. So, for every such vertex I will have one such equation and the set of Alpha these are the coefficients of auto regression. So, because I have kth order auto regression, so there will be k number of Alpha, Alpha 1, Alpha 2 up to Alpha k and if there are say P number of vertices. If this polygon contains P number of vertices, then for each of the vertex I will have one such equation. So, I will have P number of linear equations like this.

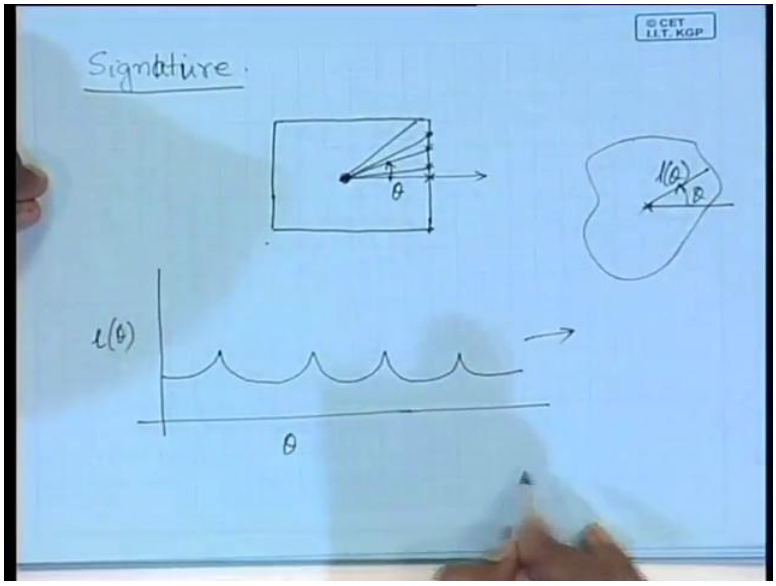
So, now the problem becomes, that I have to find out a solution of those P number of simultaneous linear equations. Now, if P is exactly same as q, exactly same as k, then this model is well defined, because I know there are k numbers of variables Alpha 1 to Alpha k and I have k number of equations, so I can get a unique solution. Whereas, if P is greater than k I will have number equations which is less than the number of variables, so the system is over specified. If P is less than k, that means the number of equations is less than number of variables, then I cannot find unique values of this set of coefficient Alpha 1 to Alpha k.

So, this model, order of the model, auto regressive model I have to decide suitably depending upon how many vertices I have in the polygon. If value of P is greater than k as I said that the model is over specified, because I have more number of equations than the number of variables. So, in such case we can go for a solution approach which is mean square error solution, which is I have to find out the set of Alphas which gives minimum error.

So, those solution approaches we will talk later on in respect when I discuss about other linear classifiers and all that. But right now what is important to us that, we have given such an auto regressive model, we can find out a set of features Alpha 1 to Alpha k and this set of Alpha 1 to Alpha k can be used as feature vector, representing this particular polygon. Now, again you can find that, I have made use of only the angle I have not made use of the side lengths or the links of the edges. So, even if I elongate this polygon either horizontally or vertically or diagonally, I will get the same set of coefficients. So, that can be taken care of, that I can have an another auto regressive model, along with this say if I consider the side length l n and this l n can also be represented by an auto, can also be modelled by an auto regressive model something like this, where this i may vary from 1 to say some value P, so Pth order auto regressive model.

So, I get a set of coefficients Alpha 1 to Alpha k considering the angle subtended at the vertices and I can also get another set of coefficients, P number of coefficients Beta i, i varying from 1 to P considering the lengths of the edges. Now, if I concatenate these two, Alpha 1, Alpha 2 up to Alpha k, then Beta 1, Beta 2 up to Beta P, so I get a feature vector of dimension k plus P and this feature vector can be used for classification purpose or for recognition purpose. So, here we take care of both the angle values at the vertices and also the lengths of the edges of the polygon. So, that will take care of, if there is any deformation either the polygon is elongated in certain direction or the polygon is compressed from some other direction.

(Refer Slide Time: 55:18)



So, these are the feature vectors that can be generated based on polygonization approach, there are other approaches as well, one is to find out the signature. I just, I forgot to mention one thing that, here as I said that initially somehow I have decided these two points, but how to decide these two points, because selection of these initial points will determine what is the accuracy of or how unique this polygonal approximation is. So, one way to decide this initial split points is that, I think all of you have done k l transformation, so in k l transformation it gives you the Eigen directions. So, I can decide the points along the first Eigen direction, because first Eigen direction gives you the direction along which your shape is maximally elongated.

So, I can take these points, these two points along the first Eigen direction, set of points between which your distance is maximum. Starting from there I can continue this algorithm iteratively to give me the polygonal approximation. Now, what is signature? The signature is something like this, which let us assume that we have a square or a rectangle, something like this. I am taking a simple example this is true for any type of arbitrary shapes and within this I have a reference direction, so this is my reference direction. What I do is, starting from the reference direction I find out the distance of all the boundary points from the centroid of this particular shape.

That is I find out the distance of the boundary point, over here I find out the distance of the boundary point over here, I find out the distance of the boundary point over here and so on. That means, I am taking the boundary point along the rotational scan lines something like this, in different directions and this direction is at an angle Theta from the reference line. Then what I do is, I plot this distance say l Theta versus Theta. So, if it is a square or a rectangle something like this, then you find that this plot will appear to be something like this, where these peaks correspond to the corners or the vertices of the square or rectangle.

Now, this particular pattern itself can work as a feature or a feature vector or I can transform this pattern to get some transformation coefficients and those transformation coefficients can work as feature vector, which can be used for recognition purpose. So, find that if I have an arbitrary shape something like this, starting from its centroid and with respect to some reference. If I go, if I go on computing the distances, l Theta at an angle Theta, then the plot of l Theta versus Theta will give you some signature of this particular shape. The signature itself or a transformed version of it can be used as a feature for recognition purpose. So, let us stop here today.