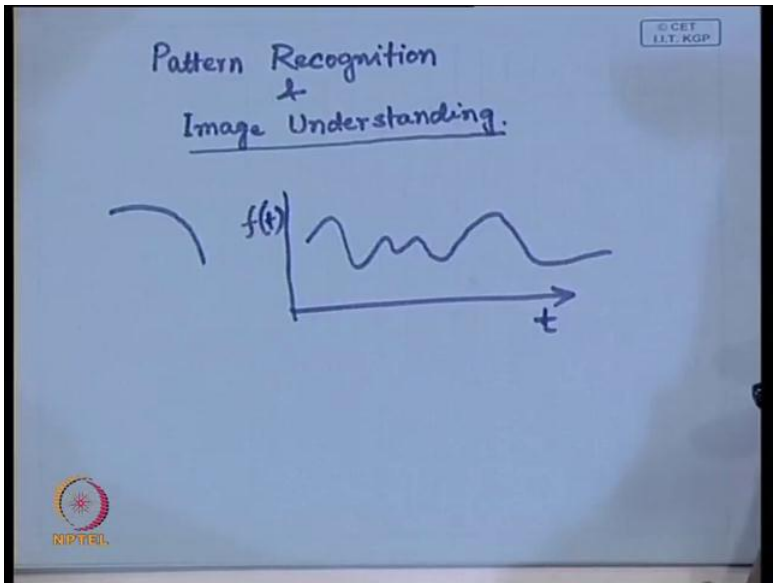
Pattern Recognition and Application
Prof. P. K. Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

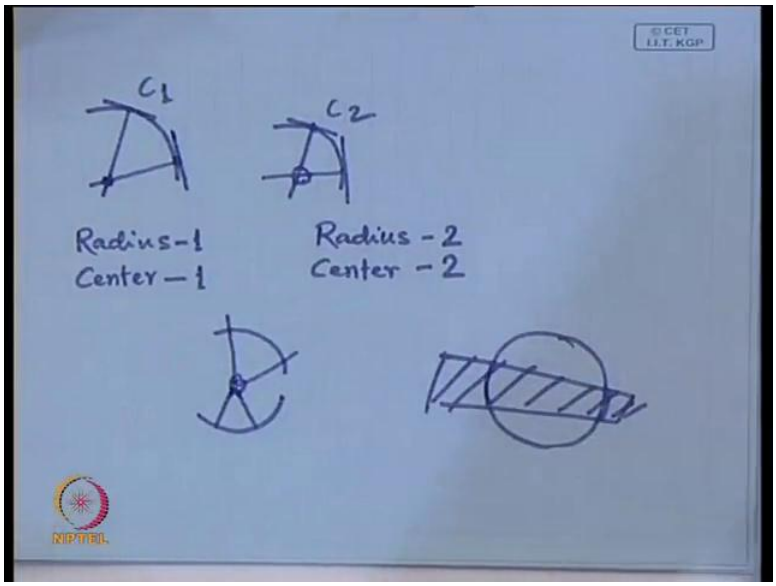
Lecture - 2
Feature Extraction – 1

(Refer Slide Time: 00:26)



So, let us take a very simple example to start with. Suppose that we have a circular arc say something like this. So, this circular arc is also a pattern. Similarly, if I have some function something like this, some variation of signal, so you can say this is a signal which varies with time. So, I can say that it is a function f (t) which varies with time t. So, this is also a pattern. Now, given different such patterns, which may be similar or which may be dissimilar. How do we say that 2 patterns are similar or 2 patterns are dissimilar, that is the basic problem of pattern recognition. So, whenever I say or I can find out that one pattern is very, very similar to another given pattern, then I say that the first pattern is recognized as the second pattern, which is nothing but the knowledge that we have or something that we have in our knowledge base.

(Refer Slide Time: 01:35)



So, coming to this simple problem say circular pattern recognition say given two circular arcs. One circular arc is like this. Another circular arc is something like this. So, how can we say that these 2 circular arcs are same or they are part of the same circle. So, as you know that if we go to parametric domain a circle, it is defined by a parameter, which is nothing but the radius of the circle.

If I do not consider the case of conciliation in variance, then I will have 2 parameters. One is the centre of the circle and other one is the radius of the circle. So, if I describe the circle only by the radius in that case, the circle is translation in variant, whereas along with radius. If I also specify the centre of the circle in that case, the circle is translation invariant because the moment I translate the circle from one location to another location. Its center is going to change even though the radius will remain the same.

So, that is why we have different kinds of features, the translation invariant features, rotation invariant features. Of course, if you rotate a circle, the features will also remain rotation invariant or else once I specify the center of the circle, it is no more translation invariant. This is because the moment you translate the circle, or you rotate the circle from one position to position to another, the centre of the circle is going to change. So, given these 2 circular arcs, if I want to say whether these 2 circular arcs are same or part of the same circle or these circular arcs are similar, then what I have to do?

I have to find out what is the radius of the circle. I have to find out what is the location of the centre of the circle. So, naturally to find out the radius of the circle, it is a matter of school level mathematics. What we will usually do is we will try to find out a perpendicular at one point on the perimeter on the circle to draw a perpendicular like this. Similarly, you take another point on the perimeter of the circle. You also draw a perpendicular to the circle at this particular location and the point of intersection of these 2 perpendiculars is nothing but the centre of the arc.

Once I get the centre, then finding out the radius is very simple. It is nothing but the distance of a point on the perimeter from the centre. So, I can easily find out the radius. Also, I can also find out the centre. Similarly, if I perform the same task for the second circular arc, I can find out what is the location of the centre or taking perpendiculars from 2 points on the circle, I can find out what is the location of the centre. Once I have the location of the centre, I can find out what is the radius of the circle.

So, in this case, let me call it this is radius 1 because it is of the first circular arc. Let me call it as centre 2 because it is the centre of the first circular arc. Similarly, in this case, let me call it as the radius 2 because it is the radius of the second circular arc and the center of the second circular arc. Now, if I find that radius 1 is same as radius 2, these 2 radii are equal.

In most of the cases, I cannot get the condition that these 2 are exactly equal because there is error of contestation, error of measurement. So, because of those error and similarly, there is also error in segmentation, it is very difficult to get perfect segmentation. If the segmentation is varying, then obviously the position of the circular arc or the points, which belong to circular arc, may deviate by 1 or 2 pixels. So, in such case, it is extremely difficult that even if they are part of similar circle, the radii are same. But, after doing this computation, radius 1 may not be exactly same as radius 2.

So, what I have to do is I have to find out a major assimilative, which is nothing but it is the difference between radius 1 and radius 2. So, if I find that the difference between radius 1 and radius 2 is very small that is negligible, I assume that they are same. So, in which case, I can say that this circular arc c 2 and this circular arc c 1, they are part of the same circle or there radii are same. If the positions of the centers are different, then

though the circles are same, these 2 circular arcs are not the part of the same circle. So, one circle is translated with this pattern or else if I have a case something like this.

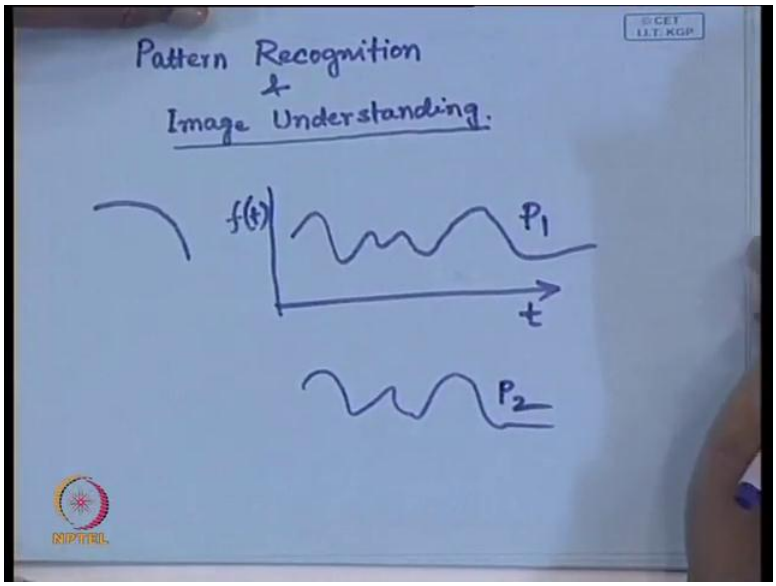
So, here what I am trying to do is I am trying to defect a case. I have 2 different circular arcs of the same circle, but these 2 circular arcs are not connected. They may not be connected because of various reasons. I can have a situation something like this say I have a circle. On this circle, I have another object. So, this part is occupied. So, this object is on top of the circle. When I take the image from the top side, naturally this part of the circular arc and this part of the circular arc or this part of the periphery will not be visible to our camera.

So, once I do this segmentation, I will get a situation something like this. So, those thought these 2 circular arcs are part of the same circle, but they become discontinuous. Now, in this case, I want to find out whether they are part of the same circle or not. I will perform similar type of operation. Find out the centroid location, location of the centre. Here, you will find that the location of the centre considering this arc and considering this arc, they will be same or they may not be exactly same. They will be very close to each other or distance between these centers is very small. So, it is this same.

The centre will be very close. The radius the length of the radius will also be similar. So, in this case, we can say that both this circular arcs are part of the same circle. So, what I am trying to do is whenever I want to recognize a pattern, what I have to do is I have to extract certain features of the pattern. Using those features in the feature domain, I have to find out whether the patterns are similar, the patterns are same or the patterns are different. So, here in this particular example, I have taken it is a very simple case of circular arc.

So, I have simply calculated the length of the radius and the position of the centre or else over here, this is not a simple pattern like a circular arc. So, in these cases, we have to find out that what are the kinds of features that we should be able to extract? Which are useful for recognition purpose or which are useful for classification purpose? So, we can have another pattern of similar form. I can say that this p 1 and this pattern p 2.

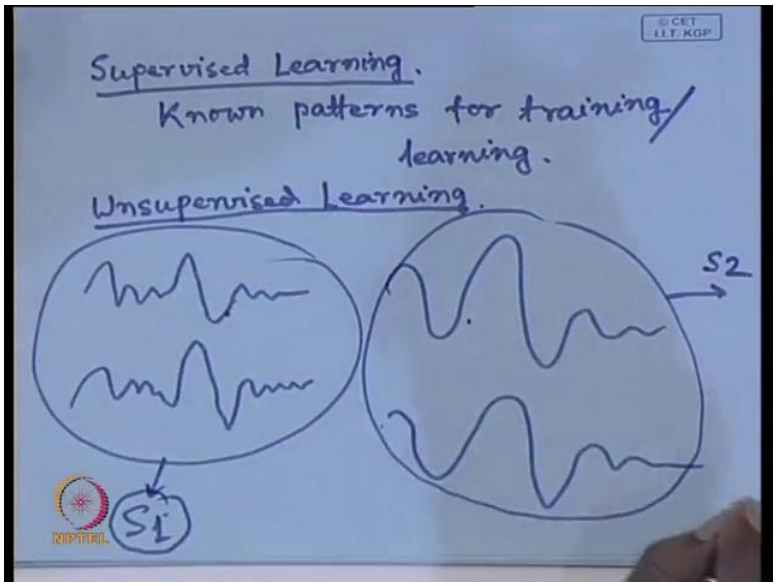
(Refer Slide Time: 09:47)



These 2 patterns are same. They are similar or they are widely different. So, this pattern recognition problem is nothing but a matching problem. So, whenever you find that the similarity between 2 patterns is very high or the dissimilarity is very low, in that case, take one of the patterns as a reference or as a model, which is there in our knowledge base. I say that the second pattern is recognized or associated with the first one.

So, in any recognition problem, I have to have set knowledge or set of model. Given an unknown pattern, I have to identify that unknown pattern with one of the pattern, which is known to us. So, that is basically the pattern recognition problem. To generate these known patterns, obviously I said that we have different kinds of learning. One is supervised learning and the other one is unsupervised learning.

(Refer Slide Time: 10:55)



So, in case of supervised learning, I have to have a set of patterns, which are known. So, a set of similar such patterns, may be this pattern is generated by some measuring instrument. So, I have to take similar such patterns from the same measuring instrument and number of such patterns and whatever feature I extract for one pattern. I have to generate similar features for all other patterns. Using these features, I have to generate a model that is a feature vector, which is representative of these classes of patterns. That is the representative feature of vector, which has to be kept in the knowledge first.

So, any time a new pattern comes, I will generate similar such feature vector and compare these 2 feature of vector. What is the distance between these 2 feature vectors? So, if the distance between the 2 features of vectors is very small, I will say that this unknown pattern is same as the pattern that I have in my knowledge base. If the distance is large, I will say this unknown pattern is different from the pattern that I have in my knowledge base.

So, in case of supervised learning, what I have to do is I have to take a set of known patterns for which I have to find the feature vectors and for those feature of vectors, I have to form a representative feature of vector. This represents the same class of patterns. So, that is what has in case of supervised learning. This makes the use of known patterns for training and learning. Another class of learning or training is unsupervised

learning. So, what is the difference between supervised learning and unsupervised learning?

In case of supervised learning, we have to make use of a number of known patterns for training purpose or for learning purpose. So, the classifier or the recognizer learns using a set of known patterns. In case of unsupervised learning, I do not have any known pattern. So, I will have patterns like this. Similar such patterns might also be there. I can have some patterns of this form, similar pattern again of this form. In case of supervised learning you know that these 2 patterns are similar or same though there might be difference or deviation. These 2 patterns are same.

This set of pattern and this set of pattern are different. That is known in case of supervised learning. In case of unsupervised learning, it is not known. So, you have a mixture of such patterns. Whenever I have a mixture of such patterns naturally for all the classification problem, I said that I walk in the feature domain, in the domain of feature vectors not in the patterns themselves. So, I generate set of feature vectors for each of these patterns. So, what I have is a mixture of feature vectors.

The features may be generated from this pattern. It might be generated from this pattern. It might be generated from this pattern. It might have been generated from this pattern. From where ever it comes, I just have a mixture of feature vectors without having any information of what is the source of that feature vector.

In case of supervised learning, I know the source of the feature vector. So, for unsupervised learning, what I have to do is I have to process those feature vectors. By processing those feature vectors, I have to partition the set of feature vectors into number of subsets. Quite the feature vectors in 1 set are generated from similar patterns, feature vectors in another subset are generated from another set of similar features.

So, the feature vectors generated from these 2 classes will be going to 1 set. The feature vectors generated from these patterns will move to another set. This partitioning of feature vectors into different sets has to be done by data processing. This knowledge is not known before hand. In case of supervised learning, this knowledge is known before hand. So, once I have.

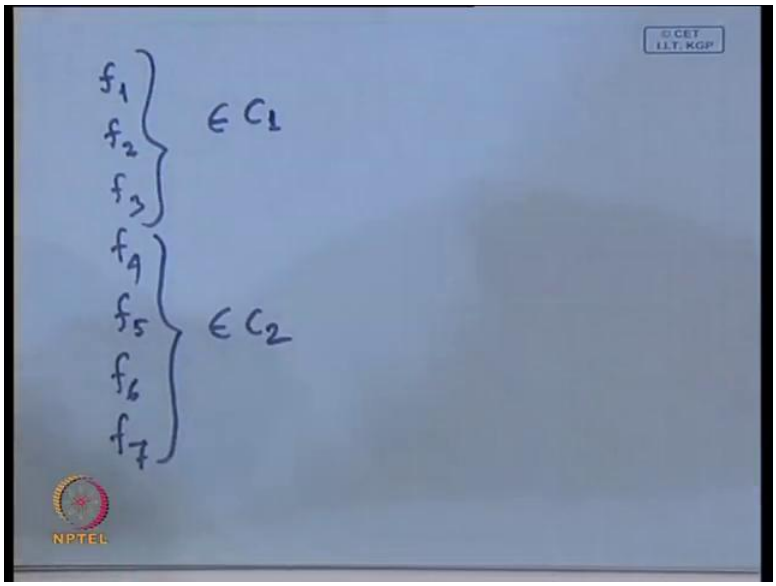
Once I partition the feature of vectors into different subsets, then I can take the feature of vectors from every set. From s 1, I take all the feature of vectors and using those features of vectors, I generate a representative feature of vector, which in case of supervised learning. This is because I knew the class belongingness of the feature of vector. Straight way, I could have used those feature of vectors to generate the representative feature of vector. In this case, the first step which was already known in case of supervised learning that is not known.

So, I have to do it. First is partitioning of feature of vectors into different subsets Once that partitioning is done, I have to take the feature of vectors from empty different partition. Using feature of vectors belonging to a single partition, I have to generate a representative feature of vector for that class of vector. Then going for recognition, the task in supervised learning, unsupervised learning remains same. So, only one step will be additional in case of unsupervised learning that I have to go for partitioning of the feature of vectors in to different partitions depending upon similarity of feature vectors.

So, when you do this partitioning, you have to make sure that all the feature of vectors which I have put into this subset s 1 they are similar all the feature of vectors, which I have put into subset s 2. They are also similar, where as if I take a feature of vector from s 1 and another feature of vector from s 2. These 2 features of vectors have to be widely different. They should not be similar. One step is less in case of unsupervised learning, in case of unsupervised learning.

One step is more that is I have too agglomeration based on similarity, which I do not have to do in case of supervised learning. That has been done. That is also learnt, but what is learnt? What it is learning that is known. See, in case of supervised learning, coming to the feature of vector domain, suppose that I have a set of feature of vectors f 1, f 2, f 3, f 4, f 5, f 6, f 7 and so on.

(Refer Slide Time: 18:27)



Suppose that this f 1 to f 3, these are the feature of the vectors that belong to say c 1. f 4 to f 7, these feature of vectors they belong to class c 2. In case of supervised learning, this is already told that these feature of vectors f 1, f 2 and f 3; they belong to class c 1. Similarly, f 2, f 3, f 4, f 5, f 6, f 7 belong to class c 2. That is told. So, when I am training my recognizer, recognizer has to make a decision that given a feature of vector, what is the class belongingness of that feature of vector?

That is the decision that the recognizer has to do. So, when the recognizer is taking this decision say given this feature of vector f 1 or another feature of vector, which is very close to f 1, which are widely different from any of these 4 features of vectors; that feature of vector the recognizer should decide that it should belong to class c 1.

So, what in case of supervised learning you have? You have the input and for this input, what decision what these recognizers has to take that is also known. So, if the recognizer takes any decision other than the correct one, you know there is an error. So, by making use of that error your approach will be to train your recognizer in such a way that the error is minimized, there are different techniques for that. We will discuss that later on.

So, the part, the class belongingness of the training samples or the training feature of vectors are known before. In case of supervised learning similarly, over here if f 4 is presented to the recognizer, the recognizer has to take a decision that f 4 belongs to class c 2. If the recognizer decides that f 4 belongs to c 1 that means there is error. So, to

minimize this error, I have performed something. So, that is what is learning or training of the recognizer.

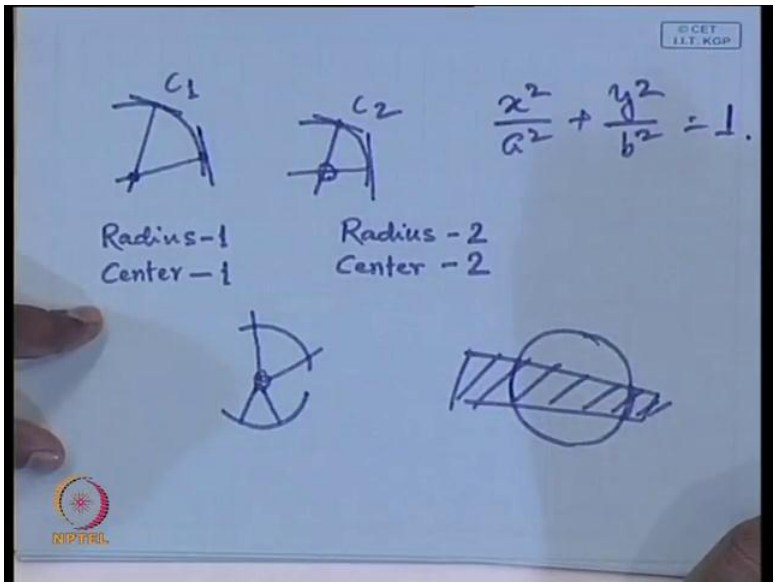
In case of unsupervised learning, I simply have this set of feature vectors. I do not know whether the feature of vectors belong to class c 1 or the feature of vector belong to class c 2. So, straight way, I cannot give this f 1 to the recognizer and expect that this recognizer will give me an output, which I can understand. So, that is why, I need a pre partitioning of the features of vectors based on similarity of feature of vectors. If I say this thought of processing after doing that, all the feature of vectors which are in same partition, they are coming from the centre. Though I may not know what that is in case of supervised learning, I know what that class in case of unsupervised is learning.

I may not know what that class is, but definitely I know that they are coming from the same class, whatever class it is. So, that is how I can form while doing this partitioning. What I am doing is I am making a shorter class association of a set of feature vectors. Once this partitioning is done, then by making use of this, I can go for a recognition process or training of the recognizer. Following the training, the recognizer can be used for pattern recognition purpose.

So, this is difference between supervised and unsupervised learning. So, in case of unsupervised learning, I have this job of partitioning of the feature of vectors into different partitions or different subsets. Once that is done, the remaining of part of learning will be same as in case of the supervised learning operation. I am coming to that. So, for determination of this feature of vectors, the simple example I have taken is circular arcs. What feature of vector you have to generate or what are the features that you should look in to that depends upon your problem.

If I know that all the patterns that I am going to get, they are basically circular arcs, then features I try to generate. It is simply the radius and the centre because given the radius and the centre that uniquely determines the circle. I do not need any other information. If it is an ellipse, then radius and circle is not sufficient. Isn’t it? I have to know the parameters a and b.

(Refer Slide Time: 23:50)



Equation of an ellipse is x square by a square plus y square by b square. I have to know what are this parameters a and b. So, what are the features or what is the feature of vector that you have to generate? That depends upon your problem domain because every feature is not equally acceptable. For every pattern depending upon the pattern, I have to decide what kind of feature I have to follow.

In many cases given a pattern, I can uniquely find out a feature vector. I can uniquely find out a feature. Given a feature, the same pattern may not be generated and may not be able to generate the same pattern given a feature of vector. So, your feature vector to pattern mapping that is actually one too many it is not immediate. Given a pattern, if my procedure for generation of feature of vector is fixed, I will always generate the same factor same feature of vector.

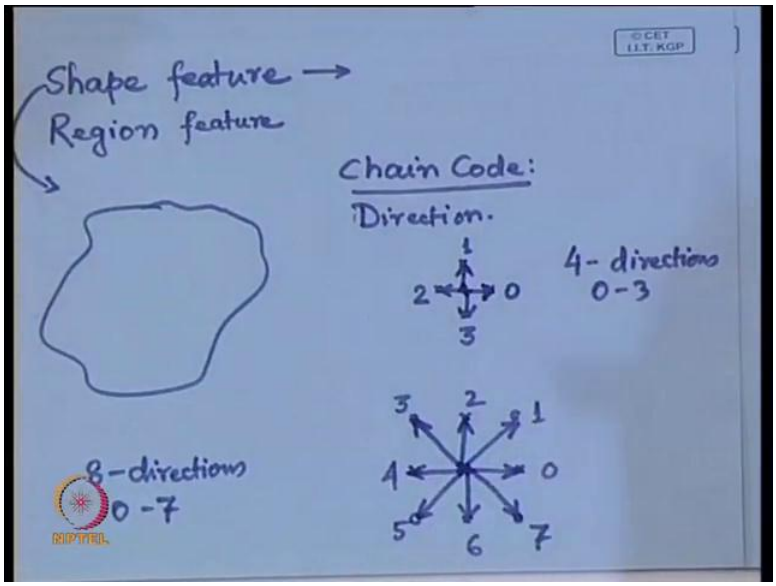
So, the mapping form pattern to features that is unique where as mapping from feature vector to pattern it is not immediate. This is because it is not unique. So many patterns may be matched to the same feature of vector. Then the question comes that if many patterns are mapped to the same feature of vector, then using the feature of vector, how do I distinguish the patterns?

This is the only reason that we do not talk about features, but we talk about feature vectors. Feature of vectors will definitely have different components. Every individual component tries to capture some property of the pattern. So, all these components taking

together in the form of feature of vector to some extent pin points to a particular vector though may not be uniquely even in that case. I can have a smaller set of pattern which will generate the feature of vector.

As you reduce the dimension of the feature of vector, this set goes on increasing. The size of the set will go on increasing. So, in all these pattern recognition problems, we never talk about a single pattern. You always talk about a single feature. We always talk about feature vector. The feature of vector has different components where every component tries to capture a certain property of the feature of the pattern. So, we are coming to the case of the feature extraction or the discreet errors extraction.

(Refer Slide Time: 26:55)



We said in the last class that we can have 2 different features; one is shape, feature or shape based feature. The other one is region based feature. So, this shape feature simply tells you what is the shape of the object whether it is a rectangle, it is a square, it is a circle, it is an ellipse and so on. If have some object, so that is what it is told by the shape, feature. The second kind of feature that is the region feature tells about talks about what is the region, what is the property of the region, which is enclosed inside the boundary by this region. What I mean is in simple grey level image, it may the intensity values.

If we have color images, it may be a color feature. It may also be texture feature if the region has different types of textures. If it is a colored texture, I can have color feature

along with the texture feature. Even a colored texture can have different intensity values. So, when you talked about colorless processing, particularly in each s i model, we have talked about hue, saturation and intensity. We have said that he gives you the color information. Saturation gives you the purity of the color and intensity is black and white equivalent of the intensity value.

So, if the intensity is high, the image will be bright. If the intensity is low, the image will be dark. So, I can have with in a region the texture feature, color feature, intensity feature, combination of all 3 or any one of them or any 2 techniques at a time. So, that is why I said that a single feature does not give you a proper description. What I have to take is multiple numbers of features. Every feature tries to capture certain property of the pattern. All those features take together in the form of a vector. Whenever I form a vector, the position of the element within the vector is very important.

So, I have to put these features in a particular order and all through, I have to maintain the same order. So, all these features taken together put in the form of a vector to some extent pin point to a particular pattern or to a class of pattern, while the domain of this class is very narrow. As you reduce the dimension of the feature of vector, the domain will become wider. This is because more and more patterns of different types will come into the same class. So, first let us concentrate on this shape feature or shape based feature.

So, suppose that we can have arbitration like this. Earlier, we said that whenever we want to detect a shape feature, it is not necessary that I have to know what is there inside. If I know or I code only the boundary of this shape that is sufficient to tell me what is the shape feature. Now, one of the kind of features or the descriptors, which can be used to describe or represent this particular shape, is what is called a chain code. What is chain code? If I want to represent this boundary arbitrators boundary by linear segments that is some sort of linear approximation of piece wise linear approximation of arbitrator boundaries, so if I want to represent this arbitrator boundary by piece wise linear approximation, then I can represent this boundary by linear segments of some specified length and specified direction.

So, I represent this boundary by sequence of linear segments of specified length and specified direction. That is what is called a chain code is that. So, when I say that I have

to have linear segments of specified lengths specified directions, so I have to specify beforehand what the length is and what is the direction.

So, firstly let me talk about how to decide the direction. This direction can be specified in either 4connectivity or in 8connectivity. I hope all of you know what is connectivity, 4 connectivity and 8connectivity. So, this direction can be specified either in terms of 4connectivity or in terms of 8connectivity. So, if I talk about 4connectivity, suppose I have a central pixel somewhere over here. Then these pixels in 4 connectivity has 4 neighbors one on top, one to the right, one at the bottom and one to the left.

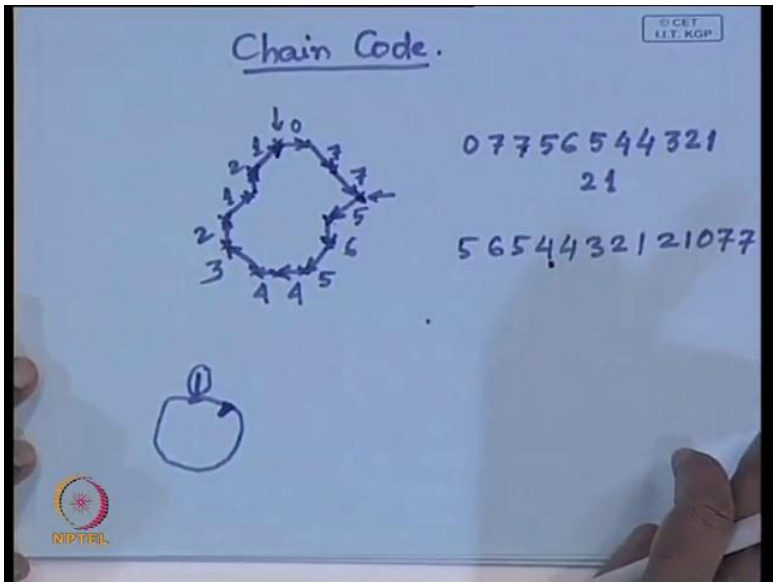
So, when I want to move from this central pixel to any of these 4 connected neighbors, I can have only 4 possible moves. I can move towards east. I can move towards north. I can move towards south. I can move towards east. So, these are the only 4 possible moves that I can make. This is what I mean by direction. So, if I specify this as direction 0, this is code as direction 1, this west I will code as 2 and south I will code as direction 3.

So, you find that every direction is now coded either as 0 or as 1 or as 2 or as 3. So, that gives me all 4 possible movements directions in which I can move following 4connectivity. If I go for 8connectivity, suppose this is my derived pixel. I have these 4 pixels, which are 4 neighbors, one to the east North West and south. In addition, I have these diagonal pixels north south, north east, North West, south east and south west direction.

Now, from the centre pixel, if I want to move, I can move in any of the 8 directions. I can move either in the north, in the east direction; I can move in the north east direction, I can move in the north direction, I can move in the North West direction or in the west direction or south west direction or south or south east. So, these are the 8 possible moves that I can make from the centre pixel following 8connectivity.

Now, if I code this movements as 0 1 2 3 4 5 6 7, so following 4 connectivity, I have 4 directions, 0 to 3. In case of 8connectivity, I have 8 directions 0 to 7. So, the directions are fixed. Now, we have to decide about the length of the given segment. I will come to the length concept a bit later. Now, let me see that how using these directions and assuming length to be 1, how I can go for coding the boundary of a shape? So, suppose that I have a shape something like this.

(Refer Slide Time: 36:11)



So, we are talking about chain code. Suppose that I have a boundary something like this. You find that can you see the grids. Actually, all these are placed on grid intersections. So, suppose that I start from any particular point. Let me assume that I scan the figure in raster scan fashion and whichever is the first boundary point, I reach that my starting point. Now, once I reach my starting point from the starting, I will move in the clockwise direction. I try to trace the boundary in the clockwise direction. So, my starting point following the raster scam fashion is this one. I want to make a move from the starting point to the next boundary point in clock wise direction.

So, when I try to make a move, obviously it will move in this direction and come to the case that if I assume that I am following this 8connectivity, this particular movement is given a code 0. So, here I will get 0 next movement over here is in the south east direction and for south east direction, following 8connectivitythe code is given as 7. So, this movement will be coded as 7. Similarly, over here, this is also 7 again south west direction. This is in the east direction. Next movement is in the south west direction and south west direction is coded as 5.

So, I will code this as 5. Next one is in the south, which is coded as 6. Again, south west coded as 5. Then in the west, it is coded as 4. One more west is again coded as 4. Then I have north east, North West which is coded as 3. Then I have movement in the north direction, which is coded as 2. Then I have north east, which is coded as 1. Then again I

have movement north coded as 2 and then again on north east, which is coded as 1. This completes the tracing of inter boundary.

Now, once I have this, you find that this sequence of movements can be used as a descriptor of this boundary. So, the sequence becomes 0 7 7 5 6 5 4 4 3 2 1 2 1. So, this codes the directions of the line segments. In this case what have in fact, here all the pixels are more than 1. All the pixels are one above another. If it is a boundary, then ideally the pixels will have only 2 numbers. If it is boundary, otherwise the third point that you have is a externs bond. So, I can have situation something like this.

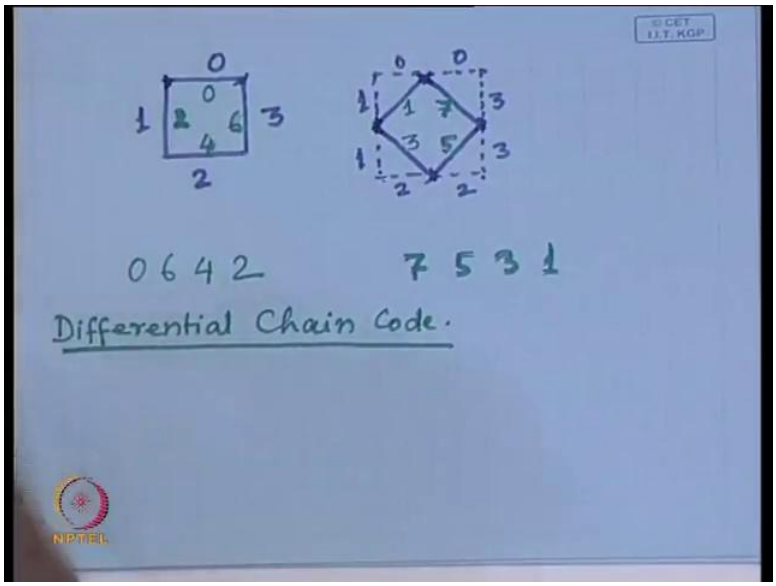
What you said is say I can have a boundary of this point, so over here, something like this is it. So, when I go for chain code, this has to be taken as externs' boundary, which has to be removed before hand. For removal of this, you must have heard about morphological filtering. Do just opening and closing operation and these externs points will be removed; those who have done mythos.

I did not talk about the morphological operations. So, by using the morphological filtering operation, such externs' points can be removed. Similarly, I can have why something inside also say something like this, which may come due to noise. So, again by using morphological operations, such kind of noise can be removed. So, coming back to this, boundary is represented by sequence of directions like this over here.

The length of the line segment is assumed to be 1. This is because I have moved from empty pixel to its neighboring pixels. So, this boundary is represented by this kind of chain code. You find that the problem with this chain code is that instead of taking this as my starting point, if the starting point comes out to be somewhere over here, then the chain code will be totally different.

The chain code will be 5 6 5 4 4 3 2 1 2 1 0 7 7. If I take this point as my starting point, then this is the chain code. If I take this point as my starting point, then this is the chain code. So, the chain codes are totally different not only that if the shape is rotated. Then I have even what situation in this case at least even if the chain code is different, but the same symbols appear in the same sequence. The shape is rotated.

(Refer Slide Time: 42:48)



Let us take very simple case, say I have a rectangle. Let us assume that length of each side of the rectangle is my unique distance. Let me also assume to make the situation even simpler that instead of this 8connectivity, I am considering 4connectivity. So, following this 4 connectivity and taking this as the starting point, the chain code for this particular figure assuming that every side length is of unit distance is of unit length will be 0321.

Now, if I simply rotate this by 45 degree following 4connectivity, I cannot generate the chain code for this figure. This is because these movements are not defined in that case or even if I take the concept of chess board movement. So, these are the boundary points. This is one boundary point. This is one boundary point. This is one boundary point. This is one boundary point. So, the kind of movement that I have to make is assuming that this is my starting point. I have to make a movement over here.

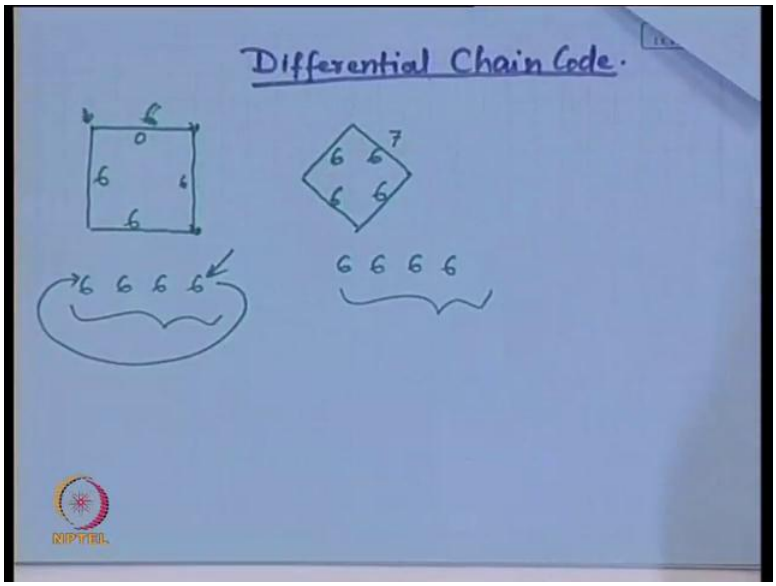
We keep this 0. I have to make a movement over here, which gives me 3. I have to make a movement over here, which gives me 3 again, movement over here 2, movement over here 2, movement over here 110. So, you find that here the chain code was 0 3 2 1. Now, my chain code is 03322110. The chain codes are totally different though over here it is simply a case of screening. In earlier case, I had every line segment in only one appearance.

Now, I have every line segment appearing twice. So, you can say that it is just a matter of scaling. I can scale it down to have the same chain code, but what happens to 8connectivity? If I go for 8connectivity, you can see they have different color. If I go for 8connectivity, then this will be coded as. This will be coded as 0. This will be coded as 6. This will be coded as 4. This will be coded as 2, where as if the same figure is rotated by 45 degrees; this will be coded as 7. This will be coded as 5. This will be coded as 3. This will be coded as 1. Now, it is visible.

So, find that over here. My chain will become 0642 and over here, the chain code becomes 7541 following 4connectivity. Though I could say that one is just a scaling version of the other, the moment I got 8connectivity. This is because this gives me more precision. The chain codes are totally different. So, when I go for such chain code representation of boundary, you find that if I simply translate this figure move from one position to other position, the chain code will remain the same.

If I rotate the figure, the chain code becomes totally different. So, this sort of representation, boundary representation is translation invariant because it does not vary with translation. It is not rotation invariant. It depends upon rotation the problem of selection of starting point. So, over here, as I change the starting point, shape, the starting point, the chain code again becomes different. So, how to take care of these problems? If I want to represent a boundary or a shape by chain code, there is a concept of differential chain code. To go for this differential chain code, let us start with the same figure. Let me take another page.

(Refer Slide Time: 48:04)



So, we are talking about differential chain code. So, let us start with the same figure. Now, over here what I do is when I move from the first point to the second point, I do not give any code to this. When I move from the second point to the third point, so here the code was supposed to 0. Here, the code was supposed to be 6 following 8connectivity. So, instead of giving this direct code, what I give is that how many rotations either in the clock wise direction or in anti clockwise direction of the rotations in my coding scheme has to be made.

So, I move from 0 to 6. So, we come to this coding scheme. So, this is 0 and this is 6. So, if I follow anti clockwise rotation, I have to move 1, 2, 3, 4, 5, 6 steps. So, I code this as 6. However, to this, I have not given any code. Now, when I move from this to this, this was originally 6. This is originally 4. So, how many such rotational steps I have to perform? So, this is 6. This is 4. So, number of steps will be 1, 2, 3, 4, 5, and 6. Again, so this also becomes 6 from here to here. I have initially 4.

Now, it has to be 2. Initially, it was 4. Now, it has to be 2. So, again how many steps I have to move 1, 2, 3, 4, 5, and 6. So, again it is 6 steps. This was 2. This has to be 0. So, 2, 2, 0, how many steps are there? There are 1, 2, 3, 4, 5, 6, again 6 steps. So, this 6, 6, 6, and 6 that becomes one differential chain code. What happens in this particular case? Here, my initial direction was 7, but I do not code it. The direction of meant was 7 same as this.

What is the next direction? Next direction is 5. You come over here. Next direction is 5. So, from 7 to 5, how many rotations I have to perform? 1, 2, 3, 4, 5, 6, so I have to perform 6 rotations. I did that. Now, this was 5 and this is 3, 5, and 3. How many rotations I have to perform? It is 1, 2, 3, 4, 5, 6 again. So, this also becomes 6. So, likewise, if you continue, you will find that all these directions in the differential code will be 6, 6 only.

So, this becomes 6666. This also becomes 6666. The codes are identical. Now, it has become very simple only 6, 6, 6, 6, 6, and 6 similarly, because my figure is simple. If the figure is a complicated figure or an arbitrary figure, then my code will not be as simple as that. So, I will get the differential chain code, which is rotation invariant. If even then the differential chain code does not remain starting point invariant depending upon the starting point, I will have to find chain codes for an arbitrary figure.

So, to take care of that, what you can do is I can consider this code as a circle. Instead of taking this as a chain, I can take it as a circle. Once I have a circle, I can decide where to cut that circle, so that I can have a numerical representation, which is either maximum or minimum. That becomes my chain code because now instead of chain, initially I considered that to be a circle. So, it will be a starting point invariant because circle does not have any starting point or any end point. Then I enforce a starting point or end point depending upon a position.

So, if I cut the circle at that point, the numerical representation of the code that I will get would be either maximum or minimum because it is maximum or minimum. Given a circle, I can find out only 1 maximum or 1 minimum; not multiple unlike I have a situation like this. I mean wherever I cut the circle, I get the same number. So, it is having all maxima or all minima. That is only because this is symmetric and simple. In case of arbitrary figure, I will not have that king of unfortunate situations.

So, symmetricity, though we like it in our daily life most likely, but it makes our life fail. Whenever for such auto machine job is an example. There is problem. Then this example is a very simple one. But, when you work with the real life problems, you do not get such simple situations. You will have arbitrary figures. So, we will stop here today. Next day, we will continue with other features and feature description techniques.

Thank you.