Computer Vision and Image Processing - Fundamentals and Applications

Professor Dr. M. K. Bhuyan
Department of Electronics and Electrical Engineering
Indian Institute of Technology, Guwahati

Lecture-26

Object Boundary and Shape Representations - I

Welcome to NPTEL MOOCs course on Computer Vision And Image Processing - Fundamentals And Applications. I have been discussing about image features, I have already discussed about edges, textures and color, which are image features. Today I am going to discuss about object boundary and object shape information that I can consider as image features. So, what is object boundary and how to represent object boundary I will discuss now.

(Refer Slide Time: 1:07)

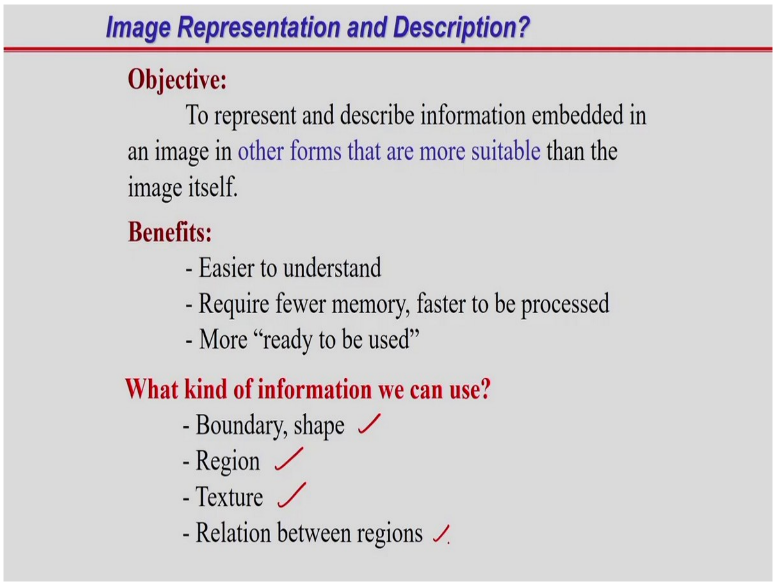


Image Representation and Description?

Objective:
To represent and describe information embedded in an image in other forms that are more suitable than the image itself.

Benefits:
- Easier to understand
- Require fewer memory, faster to be processed
- More "ready to be used"

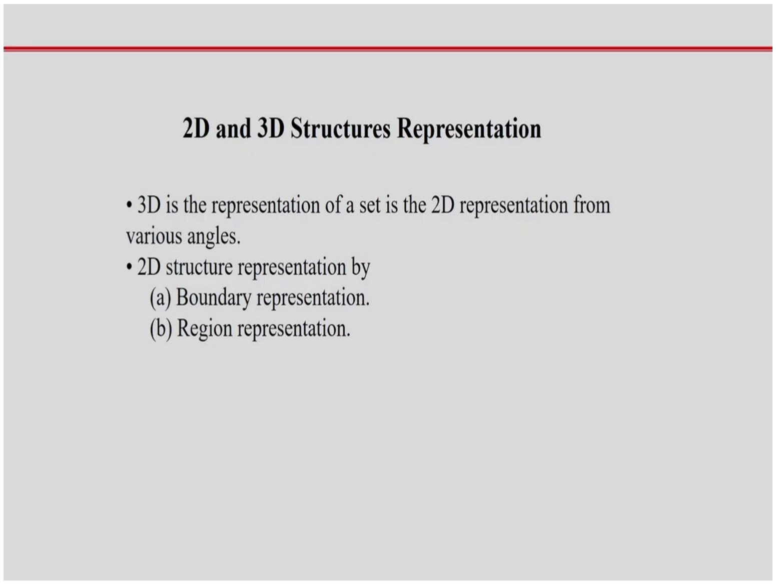
What kind of information we can use?
- Boundary, shape ✓
- Region ✓
- Texture ✓
- Relation between regions ✓

So, in this case you can see the image representation and the descriptions. So, how to represent a particular image, that is to represent and describe information embedded in an image in other forms that are more suitable than the image itself. So, that is the main concept. So, how to describe and how to represent information embedded in an image. So, instead of storing the entire image in the memory, I can store only the descriptors corresponding to that particular image.

So, what are the advantages, you can see easier to understand and require fewer memory, faster to be processed. That is, instead of storing or instead of processing the entire image, I can only consider the descriptors corresponding to a particular image. So, I can consider the descriptor for the texture, descriptor for the color, descriptor for the boundary, descriptor for the shape that I can consider and that I can store in the memory and that also I can process.

And what other information present in an image. So, other information maybe the boundary, the shape of the object I can consider, region information, how many regions are present in an image that I can consider, texture information I can consider, color information I can consider, the relationship between the regions present in an image. So, these are the information present in an image. So, already I have discussed about the texture and the color and also I have discussed the concept of the edge detection. So, now, I will discuss about how to describe boundary and the shape.

(Refer Slide Time: 2:57)

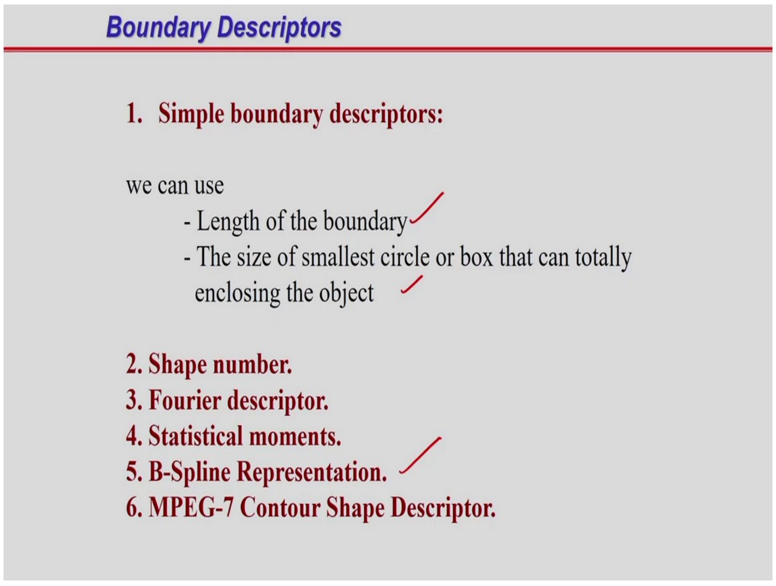


2D and 3D Structures Representation

- 3D is the representation of a set is the 2D representation from various angles.
- 2D structure representation by
  - (a) Boundary representation.
  - (b) Region representation.

So, first one is the 2D and the 3D structure representation. So, what is the 3D structure? 3D is a representation of a set of the 2D representation from various angles. So, if I consider the 2D representation from various angles, then in this case I can get the 3D representations. This 2D structure can be represented by considering the boundary and also by considering a region. So, that means, I am considering 2 image features, one is the boundary another one is shape.

(Refer Slide Time: 3:32)



Boundary Descriptors

1. **Simple boundary descriptors:**

we can use

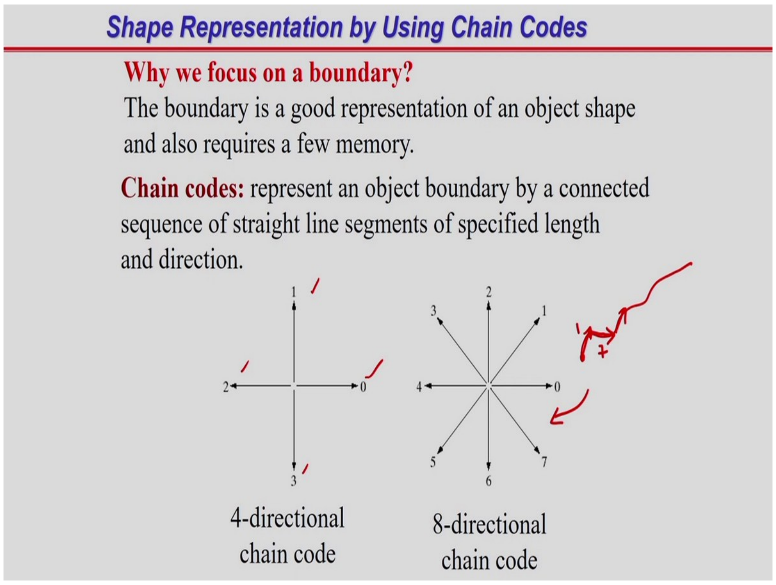
- Length of the boundary ✓
- The size of smallest circle or box that can totally enclosing the object ✓

2. **Shape number.**
3. **Fourier descriptor.**
4. **Statistical moments.**
5. **B-Spline Representation.** ✓
6. **MPEG-7 Contour Shape Descriptor.**

So, for boundary descriptors, I can consider information like this length of the boundary I can consider, the size of the smallest circle or box that can totally enclosing the object, that also I can consider as a descriptor for the boundary. If I consider these descriptors like the shape number I can consider, the Fourier descriptor is very easy very good descriptor to represent a particular boundary, statistical moments also I can consider, a B spline representation for boundary representation that is also very important, that I am going to discuss the B spline representations.

And one is MPEG 7 contour shape descriptors. So, by using these descriptors, I can represent a particular boundary present in an image. So, by using these descriptors, the shape number Fourier descriptors, the B spline, MPEG 7 contour shape descriptors, I can represent a boundary.

(Refer Slide Time: 4:35)



Shape Representation by Using Chain Codes

Why we focus on a boundary?
The boundary is a good representation of an object shape and also requires a few memory.

Chain codes: represent an object boundary by a connected sequence of straight line segments of specified length and direction.

1
2
0
3
4-directional chain code
2
3
1
4
0
5
7
6
8-directional chain code

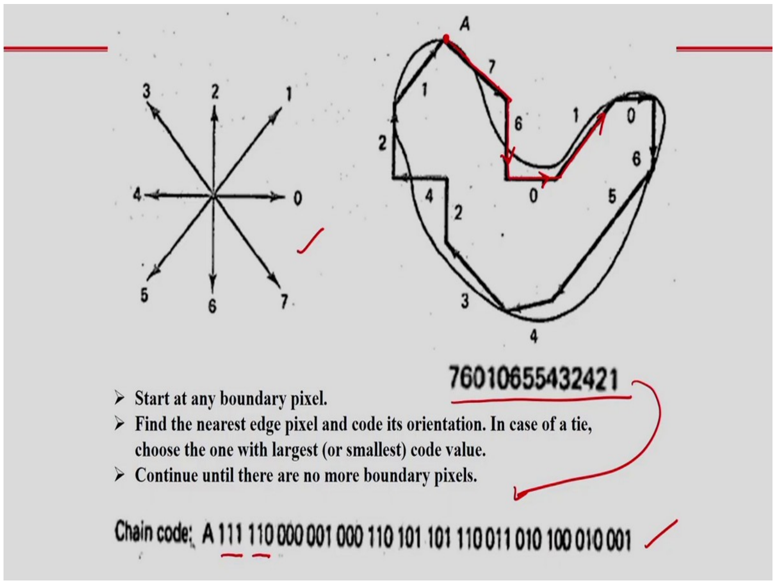
So, first one is the shape representation by using chain codes. So, in this case, you can see what is the chain code here. The chain code represents an object boundary by a connected sequence of straight line segments of specified length and direction. So, in this case in the first figure you can see I am considering 4 directions and direction is this 0, 1, 2,3. So, I am only considering 4 directions and from this I can determine 4 directional chain code.

In the second figure I am considering 8 directional chain code. So, in that case I am considering 8 directions 0, 1, 2, 3, 4, 5, 6, 7 like this I am considering So, that means, I am considering 8 directions for chain code. So, suppose I have a motion trajectory, suppose I have a trajectory suppose these type of trajectory, this trajectory I can represent by using the chain code because in this case you can define the direction, suppose corresponding to this segment I have 1 direction that direction I can see from the chain code, I can use the 8 directional chain code.

So, if you see this direction this is I can consider as 1 and suppose if I considered from this segment to this segment this portion I can consider as the direction 7 corresponding to this 8 directional chain code. And similarly, if I consider this direction also from this point to this point, I can find a particular direction from the 8 directional chain code or I can use 4 directional chain code. So, this is one application of the chain code So, representation of a motion trajectory by using the chain code.

So, for example, suppose, a gesture I am doing, gesture recognition and hand is moving in the space and in this case, I am getting the trajectory of the hand movements. So, this trajectory I can represent by using the chain code.

(Refer Slide Time: 6:45)



76010655432421

- Start at any boundary pixel.
- Find the nearest edge pixel and code its orientation. In case of a tie, choose the one with largest (or smallest) code value.
- Continue until there are no more boundary pixels.

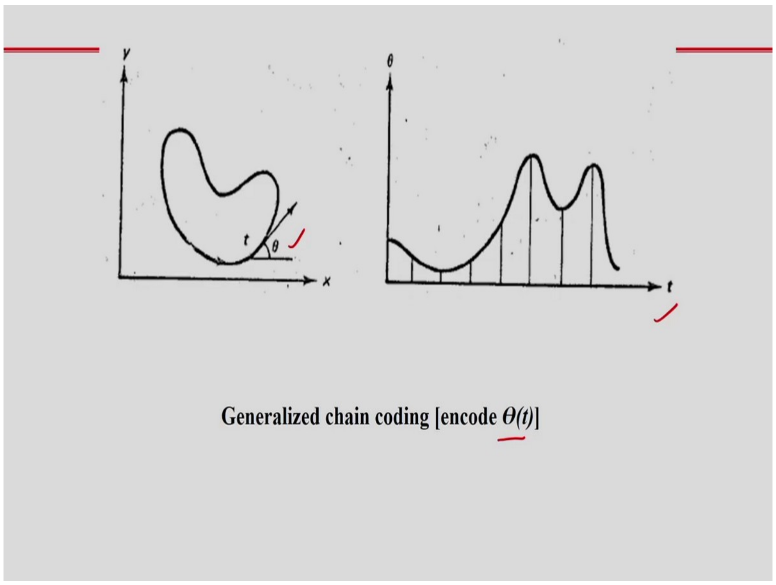
Chain code: A 111 110 000 001 000 110 101 101 110 011 010 100 010 001

And in this case, I have shown how to determine the chain code. Here I am considering 8 directional chain code. So, I have 8 directions 0, 1, 2, 3, 4, 5, 6, 7. So, I have 8 directions and I am considering one arbitrary boundary. So, what I have to do, start at any boundary pixels suppose, if I considered a pixel is suppose A. Start at any boundary pixels I am considering and suppose if I considered the first segment. So first segment is from this to this.

Corresponding to this if you see the direction the direction is 7 after this I am considering the next direction that direction is this. So, corresponding to this direction, if I see the value in the left figure that is the chain code it will be 6 and again if I consider this segment corresponding to that segment my direction will be 0 and corresponding to this direction, my direction will be 1, like this I can consider all the directions. So, corresponding to this if you see, so, I have these directions, the directions are 7, 6, 0, 1, 0, 6, 5, 5, 4, 3, 2, 4, 2, 1 corresponding to their particular boundary.

And after this number I can represent in the binary form. So, starting point is the A, now, corresponding to 7 I have 111, 6 is 110 that is represented in the binary form that I can represent in the binary form that is the chain code. So, you can understand the concept of the chain code, I can consider 4 directional chain code, but if I want to get more accurate boundary representation, then in this case you have to consider 8 directional chain code or maybe 18 directional chain code like this you can consider the possible directions.

(Refer Slide Time: 8:41)

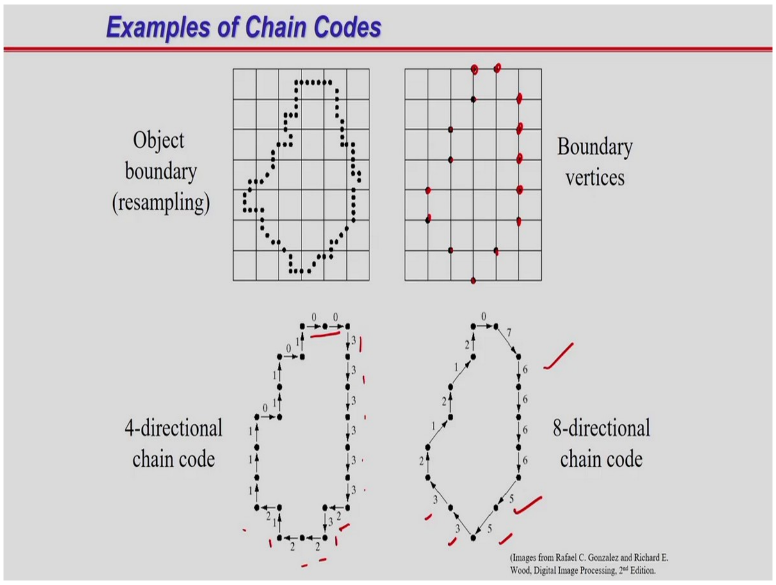


Generalized chain coding [encode $\theta(t)$]

And in this case, I am considering a generalized chain coding. So, in this case you can see I am considering the angle theta and the perimeter I am considering the perimeter is t. So, what I have to consider the encode Theta, t. So, for different angles you can see, I can determine theta t and in this case this curve I am showing a theta versus t, t is the perimeter and theta is the angle. So, this is the generalize chain coding thus I am plotting theta vs t.

(Refer Slide Time: 9:16)

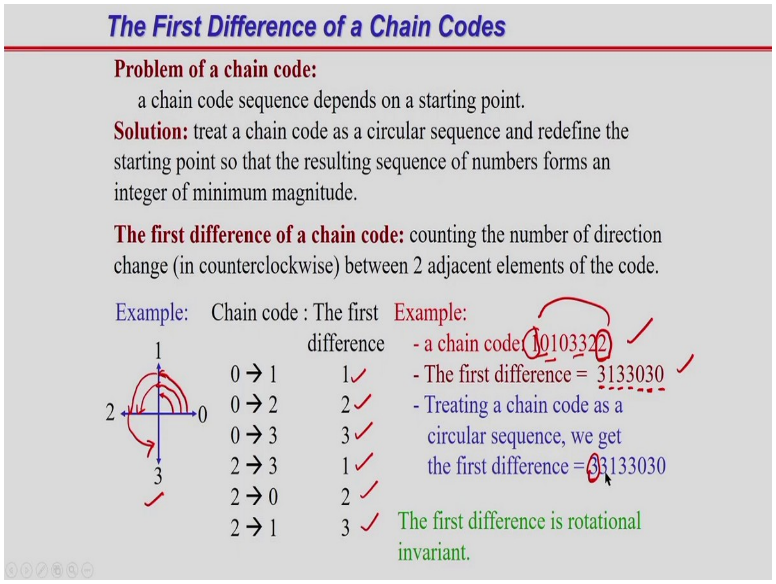
Examples of Chain Codes


Object boundary (resampling)
Boundary vertices
4-directional chain code
8-directional chain code
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

And in this case I am giving one example how to determine the chain code for a boundary. So, in this case I am considering one object boundary and I am doing the resampling, the resampling of the object boundary. So, after resampling you can see I am getting the boundary vertices. So, these are the vertices I am getting after resampling. So, these are the boundary vertices I am getting. After this, if you consider the segment by segment you can see if I consider a 4 directional chain code, can see the directions.

The direction is 00 333 like this and again 2, 3 221 like this, I am getting the boundary directions, all the directions I am getting. And if I consider 8 directionals chain codes, then you can see the directions all directions 076666 and 5, 5 3 3 like this, I am getting the directions. And based on this I am getting the chain code. The chain code can be converted into binary numbers.

(Refer Slide Time: 10:35)



The First Difference of a Chain Codes

Problem of a chain code:
a chain code sequence depends on a starting point.

Solution: treat a chain code as a circular sequence and redefine the starting point so that the resulting sequence of numbers forms an integer of minimum magnitude.

The first difference of a chain code: counting the number of direction change (in counterclockwise) between 2 adjacent elements of the code.

Example:

Chain code :	The first difference
$0 \rightarrow 1$	1 ✓
$0 \rightarrow 2$	2 ✓
$0 \rightarrow 3$	3 ✓
$2 \rightarrow 3$	1 ✓
$2 \rightarrow 0$	2 ✓
$2 \rightarrow 1$	3 ✓

Example:
- a chain code: 10103322 ✓
- The first difference = 3133030 ✓
- Treating a chain code as a circular sequence, we get the first difference = 33133030

The first difference is rotational invariant.

So, now, the problem of a chain code. The problem of the chain code is a chain code sequence depends on the starting point. In my last example. I have considered a starting point as the A, A is the starting point, where the chain code sequence depends on the starting point. So, that is why I can consider the chain code as a circular sequence and redefine the starting point, so that the resulting sequence of numbers forms an integer of minimum magnitude, that I can consider.

So, for this I have to consider and quantity, the quantity is the first difference of a chain code. So here in this case, I am considering the 4 directional chain code and how to determine the first difference. So first difference is determined like this. The counting the number of direction change in anti clockwise direction, that is the counterclockwise direction, between 2 adjacent element of the code.

Suppose in this example, I am considering the transition from 0 to 1 the direction is 0, and it is transition from 0 to 1. So, I have to see in the counterclockwise direction, from 0 to 1, the first difference will be 1 because only 1direction change from 0 to 2, how many directions change the first one is 0 to 1 and after this from 1 to 2, that means 2 directions change that is the first difference. And again from 0 to 3, if you see it should be in the anti clockwise direction.

So, first one is 0 to 1 and after this 1 to 2, and after this 2 to 3. So, if I consider the direction change from 0 to 3, then the first difference will be 3. Similarly, from 2 to 3, the first difference will be 1, from 2 to 0, it will be 2 from 2 to 1, it will be the first difference will be 3. So, in this case, you can see corresponding to this chain code the chain code is suppose 101 03322. So I am considering a chain code. And from this, I can determine the first difference. The first difference I am determining like this, so from 1 to 0, so how many directions change 1 to 0, the 3.

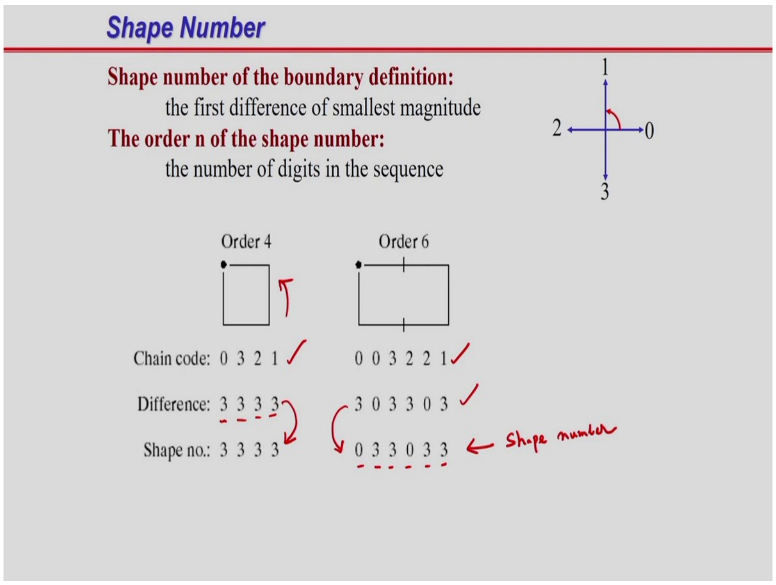
Next one is from 0 to 1, 0 to 1, how many directions change, one. So I am considering one next 1 to 0, so how many directions change 1 to 0? So directions in this 3, that is the first difference from 0 to 3, we have the first difference is 3 next from 3 to 3, the directions change is 0 and from 3 to 2 the directions change is 3 and from 2 to 2 the first difference will be 0. So, like this, I am calculating the first difference. After this the chain code is treated as a circular sequence and we get the first difference like this.

So, if I consider this as a circular sequence, then in this case, if I consider the direction change from 2 to 1, so direction since from 2 to 1 will be 3. So, that I am considering here. The direction change from 2 to 1, it will be 3 that I am considering, because I am considering the chain code as a circular sequence and the first difference is rotational invariant. So, in this example, I have shown, so from the chain code, how to determine the first difference and after this what we have to consider.

The chain code is treated as a circular sequence. And from this, we can determine that the first difference so circular sequence means you can see from 2 to 1 that is from 2, this to 1 I have to again consider how many direction change so 2 to 1 how many direction change it is

3. So there I am considering here, so that is the first difference of the chain code. The first difference is rotational invariant.

(Refer Slide Time: 14:53)



Shape Number

Shape number of the boundary definition:
the first difference of smallest magnitude

The order n of the shape number:
the number of digits in the sequence

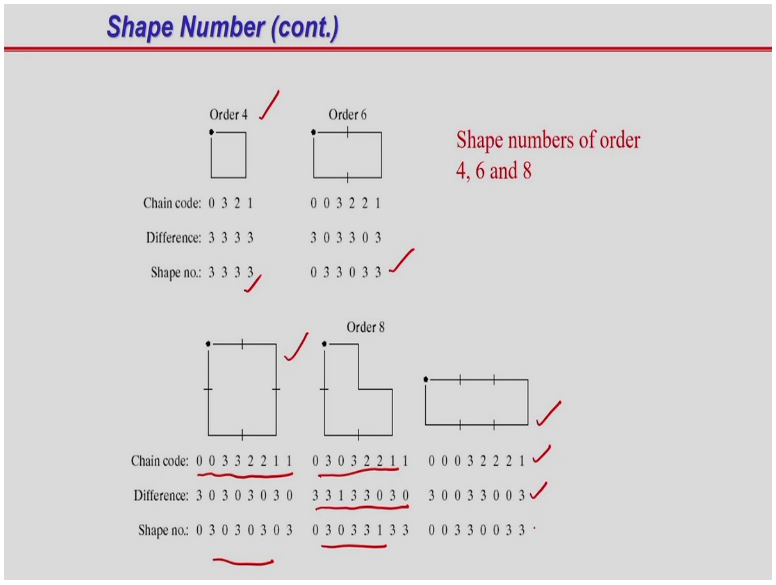
1
2
0
3
Order 4
Order 6
Chain code: 0 3 2 1
0 0 3 2 2 1
Difference: 3 3 3 3
3 0 3 3 0 3
Shape no.: 3 3 3 3
0 3 3 0 3 3
Shape number

After this there is another definition, that definition is the shape number. The shape number of the boundary, how to determine. This is the first difference of the smallest magnitude that is the definition of shape number, the first difference of smallest magnitude. So, for this what I have to consider, first I have to determine the first difference and after this I have to consider the first difference of smallest magnitude I have to consider. So, corresponding to this case you can see, I am considering the chain code the chain code is 0321.

So, corresponding to that boundary I am considering the order is 4 suppose and corresponding to this I can determine the first difference, the first difference will be 3333, this is the first difference. And what will be the smallest magnitude? In this case corresponding to this number the smallest magnitude will be the same the 3 3 3 3. In the second case, if I consider order 6, that 003221, that is a good I am getting corresponding to this order 6, that boundary I am considering.

And after this I can determine the first difference I can determine. And corresponding to this first difference what is the smallest magnitude of the first difference, the smallest magnitude will be 033 033 that is the smallest magnitude I am considering, that is the step number. So, this is my step number. So, in this case, I have explained how to determine the Shape number corresponding to a particular boundary.

(Refer Slide Time: 16:43)



Shape Number (cont.)

Shape numbers of order 4, 6 and 8

	Order 4	Order 6
Chain code:	0 3 2 1	0 0 3 2 2 1
Difference:	3 3 3 3	3 0 3 3 0 3
Shape no.:	3 3 3 3	0 3 3 0 3 3

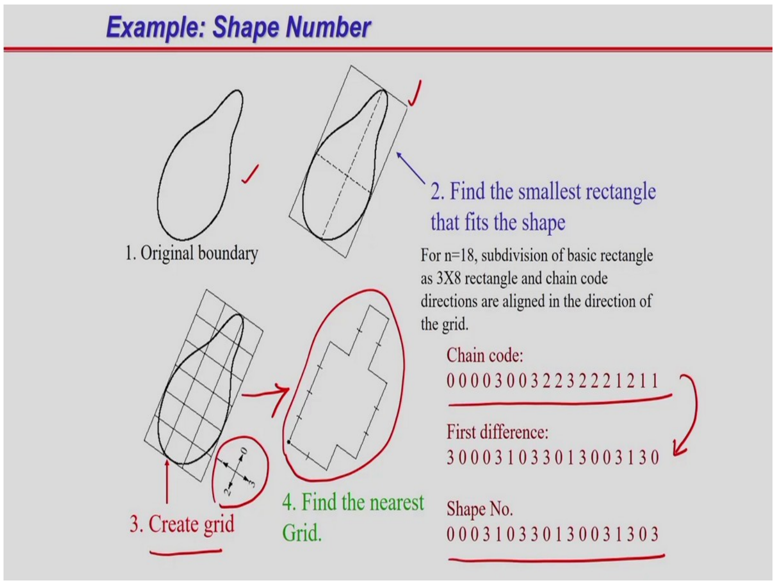
Order 8

Chain code:	0 0 3 3 2 2 1 1	0 3 0 3 2 2 1 1	0 0 0 3 2 2 2 1
Difference:	3 0 3 0 3 0 3 0	3 3 1 3 3 0 3 0	3 0 0 3 3 0 0 3
Shape no.:	0 3 0 3 0 3 0 3	0 3 0 3 3 1 3 3	0 0 3 3 0 0 3 3

Here I have given another example. So, already I have shown this example that is the order 4 and that boundary is considered and from this I can determine the Shape number. Similarly, corresponding to order 6 I can also determine the Shape number. And if you see here corresponding to this boundary my chain code is, this is my chain code 00 33 22 11 and from this I can determine the first difference and after this I can determine the Shape number.

Similarly, in the next example, you can see the chain code, you can see the order is 8 and you can see the first difference I can consider and after this I can determine the Shape number. So, for the last boundary also I can determine the chain code, there is the order is 8 and also I can determine the first difference and from the first difference I can determine the shape number.

(Refer Slide Time: 17:44)



Example: Shape Number

1. Original boundary
2. Find the smallest rectangle that fits the shape
3. Create grid
4. Find the nearest Grid.

For n=18, subdivision of basic rectangle as 3X8 rectangle and chain code directions are aligned in the direction of the grid.

Chain code:
0 0 0 0 3 0 0 3 2 2 3 2 2 2 1 2 1 1

First difference:
3 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0

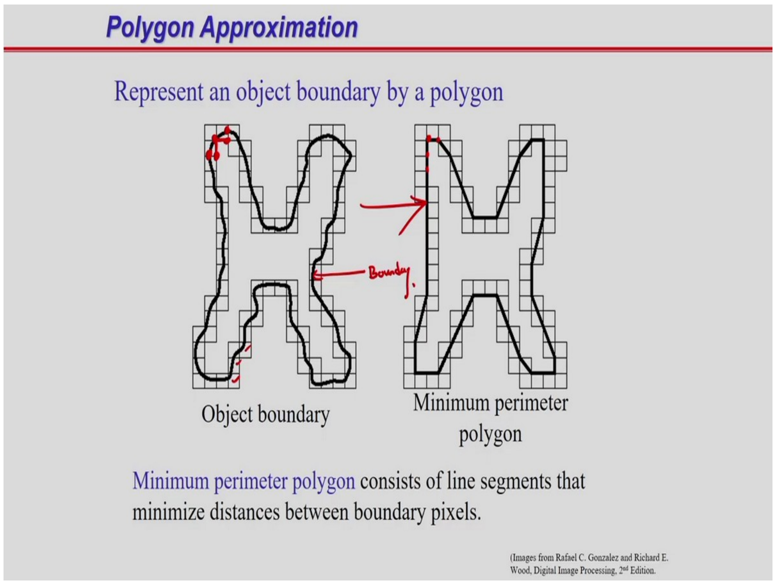
Shape No.
0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0 3

So, suppose if I consider one boundary, one arbitrary boundary, so how to determine the shape number. So, first I am considering one original boundary, that boundary I am considering. After this find the smallest rectangle that fits the shape. So, I am finding one smallest rectangle and that fits the shape. And after this what I am considering, I am creating the grid. So, you can see the, you can see the third figure. I am creating the grid for n is equal to 18 subdivision of basic rectangle is 3 cross 8.

That means, I am considering 3 columns and 8 rows and chain code directions are aligned in the direction of the grids. So, we can determine the chain code from the grids direction. So, corresponding to this one, if you see, I can find the chain code because you can see the chain code directions are aligned in the direction of the grid and I am considering this is the chain code the 4 directional chain code I am considering 0123.

And corresponding to this case, I am getting the chain code the chain code will be this. This is the chain code corresponding to that boundary. So, this is the approximated boundary I am considering, this is the approximated boundary and after this it is very easy to determine the first difference I can determine and the from the first difference I can determine the shape number. So, by using the shape number I can represent a particular boundary that is the boundary representation by using shape number.

(Refer Slide Time: 19:36)



Polygon Approximation

Represent an object boundary by a polygon

Minimum perimeter polygon consists of line segments that minimize distances between boundary pixels.

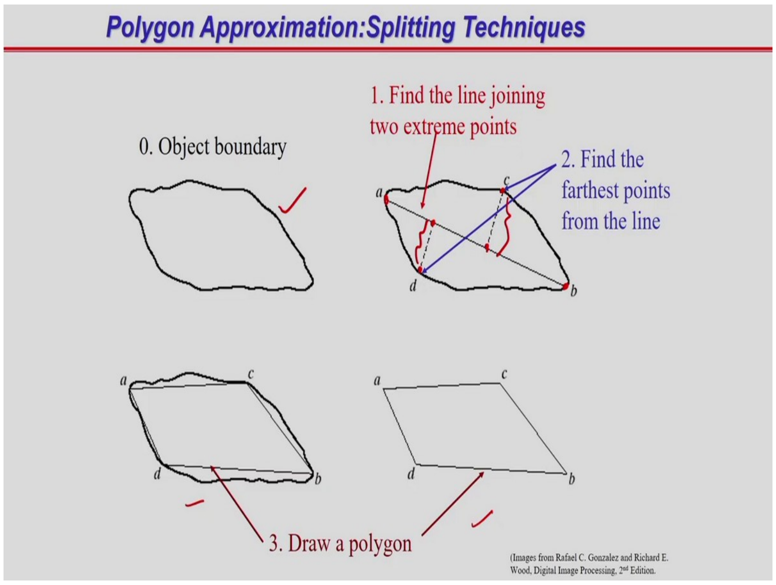
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

There is another method and that also we can use for representation of a particular boundary. The polygon approximation, represent an object boundary by a polygon. So, you can see the object boundary, so this is the object boundary. You can see this is the boundary and after this I am considering the polygons you can see I am considering the polygons like this. So, how to represent the boundary by polygon. So, minimum perimeter polygon consists of the line segments that minimize the distance between the boundary pixels.

So, in this case you can see that boundary is represented by the polygon. So, corresponding to this you see the boundary points I am representing. So, in this case what I have to consider the minimum perimeter polygon I am considering, it consists of a line segment, so I am considering the line segment that minimize the distance between the boundary pixels. So, corresponding to this boundary, if you see these 2 points, what will be the minimum distance between the boundary pixels, the minimum distance will be this. This is the minimum distance.

Similarly, corresponding to these 2 points, suppose this point and at this point what will be the minimum distance between the boundary pixels, the minimum distance of the boundary pixels will be this. So, from this I am getting this one, that is the object boundary representation by a polygon and I am considering the minimum perimeter polygon which consists of line segment that minimize distance between the boundary pixels.

(Refer Slide Time: 21:29)



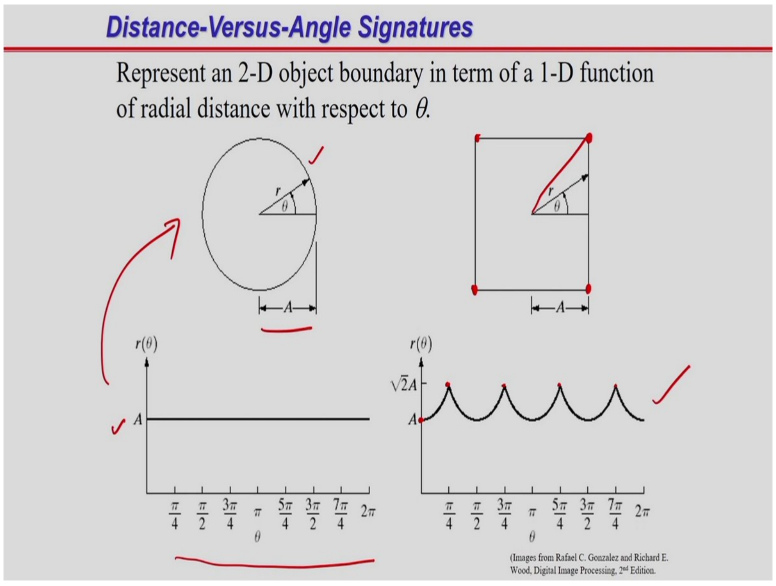
Polygon Approximation:Splitting Techniques

0. Object boundary
1. Find the line joining two extreme points
2. Find the farthest points from the line
3. Draw a polygon
a
b
c
d
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

And in this case, you can see how to represent the object boundary by polygon approximation. So, here you can see I am considering one arbitrary boundary, the object boundary and in this case you can see, find a line joining 2 extreme points. So, these lines if you see, the line connecting a and b and also I can determine, this distance also I can determine these distance also I can determine. So, that is I can determine the line between point A and B, find the line joining 2 extreme points I can determine that is the line AB.

After this I can find out this distance I can find, find the furthest point from the line, that is this distance I can determine. So, by using this information, I can represent the boundary. And here you can see the boundaries approximated by a polygon. So, if you see the boundary, the boundary is approximated by a polygon. So, like this I can represent a particular boundary.

(Refer Slide Time: 22:42)



Distance-Versus-Angle Signatures

Represent an 2-D object boundary in term of a 1-D function of radial distance with respect to θ.

r
θ
A
$r(\theta)$
$\sqrt{2}A$
$\frac{\pi}{4}$ $\frac{\pi}{2}$ $\frac{3\pi}{4}$ π $\frac{5\pi}{4}$ $\frac{3\pi}{2}$ $\frac{7\pi}{4}$ 2π

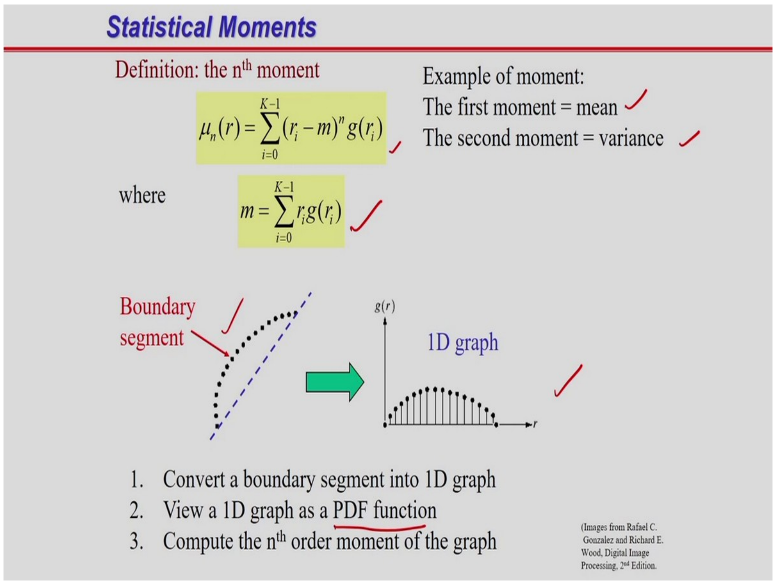
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

And another one is the distance versus angle signature. So, how to represent a 2D boundary in terms of 1d function. The 1d function is the radial distance with respect to theta. So, corresponding to the first boundary that is the circle, I am considering the radial distance and also angle theta. So, corresponding to this you can see corresponding to a circle the radial distance will be always constant, that is the value is always a. So, the radial distance is always a corresponding to all the angles starting from 0 degree pi by 4, pi by 2, 3 pi by 4, pi.

For all the angles the radial distances fixed that is constant, that is A corresponding to the boundary the boundary of the circle, corresponding to this. And if I consider a square, corresponding to square, if I considered a radial distance with respect to theta, then in this case you will be getting the 1d function like this. So, corresponding to these distance if I considered a diagonal this distance, the distance would be root 2a and for all other points, the distance is a, radial distance is A.

So, this distance is a, but corresponding to the diagonal points, the distance will be root 2A, root 2A, root 2A, root 2A. So, I have 4 diagonal points that is 1, 2, 3, 4. So, corresponding to 4 diagonal points, the radial distance is root 2 A and for other points the radial distance will be only A, so that I am considering. So, that means, by considering 1d function of radial distance with respect to theta, I can represent a particular boundary.

(Refer Slide Time: 24:38)



Statistical Moments

Definition: the nth moment

$$\mu_n(r) = \sum_{i=0}^{K-1} (r_i - m)^n g(r_i)$$

where

$$m = \sum_{i=0}^{K-1} r_i g(r_i)$$

Example of moment:
The first moment = mean
The second moment = variance

Boundary segment
g(r)
1D graph
r

1. Convert a boundary segment into 1D graph
2. View a 1D graph as a PDF function
3. Compute the nth order moment of the graph

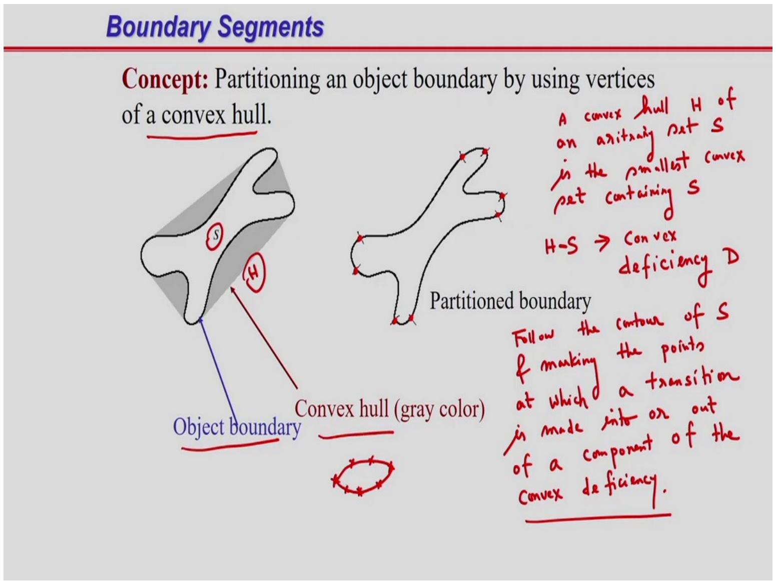
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

And also another method that by using the statistical moments I can represent a particular boundary. So, in this case, this is the definition of nth moment. So, first moment is called the mean you know and this one, the second moment is called the variance. So, what I can consider suppose I have the boundary, the boundary segment is given. The boundary segment I can convert into 1 D graph. So, you can see the conversion, the boundary is considered as 1d graph and that can be considered as the PDF, the probability density function.

And from the PDF I can determine all the moments, I can determine the first moment that is the mean I can determine, I can determine the second moment, third moment I can determine. So, by using these moments, I can represent the boundary. So, I am repeating this how to represent the boundary by using the statistical moments. So, boundary segment I am considering and this boundary segment I can consider is 1d graph and that I can consider as a PDF, the probability density function.

And after this from the PDF I can determine statistical moments. So, all the statistical moments I can determine, the first order moment and the second order moment I can determine. And by using these moments I can represent a particular boundary.

(Refer Slide Time: 26:07)


Boundary Segments

Concept: Partitioning an object boundary by using vertices of a convex hull.

A convex hull H of an arbitrary set S is the smallest convex set containing S

H-S $\rightarrow$ Convex deficiency D

Follow the contour of S & marking the points at which a transition is made into or out of a component of the convex deficiency.

And this is also a very important technique that is I can represent a boundary by convex hull, by a convex hull. So, first I have to define what is a convex hull. So, a convex hull I can represent as H suppose, a convex hull H of an arbitrary set S suppose, is a smallest convex set containing S. So, I am writing here the definition of the convex hull. A convex hull H I am considering of an arbitrary set S is the smallest convex set containing S.

So, in this case I am considering the set, you can see the set here this is the set S and I have shown the object boundary, you can see the object boundary. And if you see that gray color, that is nothing but the convex hull, H is the convex hull of an arbitrary set S is the smallest convex set containing S. So, I am considering H and within this H, the set S is contained. After this the H minus S that is the convex hull minus S, this is called convex deficiency, the convex deficiency I can consider as D.

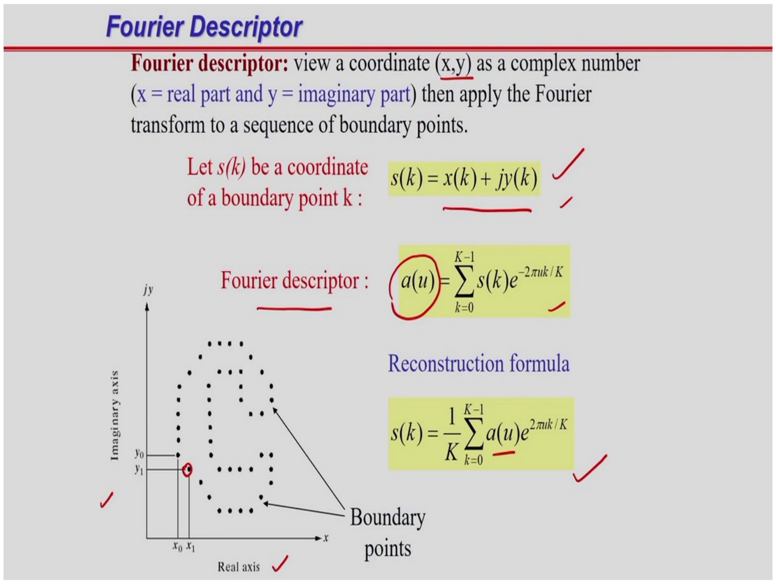
Now, in this case for partitioning the boundary. So, what is the procedure, I am writing the procedure for partitioning what we have to consider follow the contour of S, S is the arbitrary set, one arbitrary set I am considering. Follow the contour of S and marking the points at which a transition is made into or out of a component of the convex deficiency. So, I have to mark the points, that is how to partition the boundary.

So, for partitioning the boundary follow the contour of S and marking the points I am doing the marking, the marking of the points these points I am marking. Based on what, as to who is a transition is made into or out of a component of the convex deficiency. So, for partitioning the boundary I have to identify these points. So, I have to identify these points. So, and based on these points I can represent a particular boundary.

So, that means this is the procedure, that follow the contour of S and marking the points at which the transition is made into or out of a component of the convex deficiency. So, one practical example is suppose in the human eye suppose the cornea suppose the cornea shape something like this. So, how to represent this cornea, by using the convex hull. So, in this case I have to identify these points, I have to identify these points based on this concept of the convex hull.

So, I have to partition the boundary and this concept I can apply the concept of the convex hull I can apply to find this at the boundary points that is I am partitioning the boundary. So, like this in practical examples, the practical applications we can use the convex hull to partition a particular boundary.

(Refer Slide Time: 30:50)



Fourier Descriptor

Fourier descriptor: view a coordinate (x,y) as a complex number (x = real part and y = imaginary part) then apply the Fourier transform to a sequence of boundary points.

Let $s(k)$ be a coordinate of a boundary point k : $s(k) = x(k) + jy(k)$

Fourier descriptor : $a(u) = \sum_{k=0}^{K-1} s(k) e^{-2\pi uk/K}$

Reconstruction formula

$$s(k) = \frac{1}{K} \sum_{k=0}^{K-1} a(u) e^{2\pi uk/K}$$

The next important point is the Fourier descriptors. So, by using the Fourier descriptors we can represent a particular boundary. So, in this case you can see, a view the coordinate x y is a complex number. So, x y is the boundary point and I can consider as a complex number. So, x is the real part and y is the imaginary part. So, in this case, I am considering the boundary is like this x k plus j yk that is a complex number. And in this case, I am considering the boundary, boundary is SK, SK be the coordinate of the boundary point K.

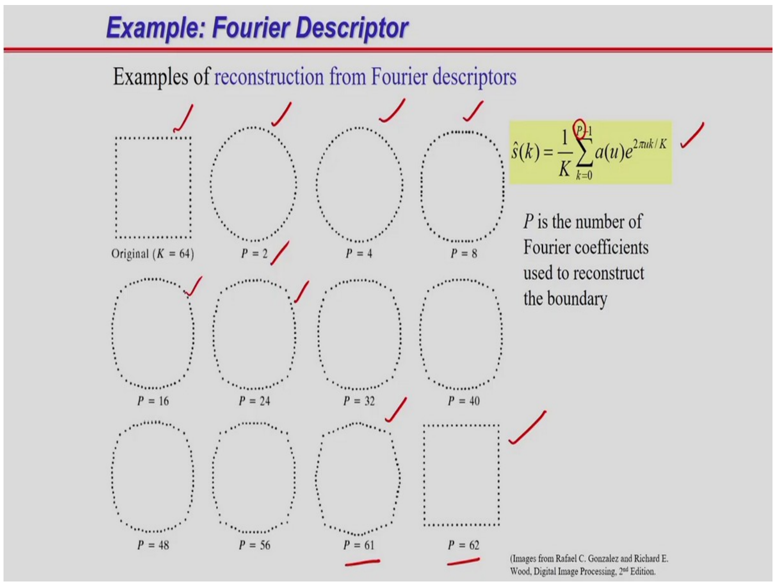
And I am considering x coordinate and the y coordinate, x is the real part and y is the imaginary part. So, I am getting the complex number. So, this is my complex number. And in this figure you can see, I have shown the boundary points and I have shown the x coordinate and the y coordinate. So, x coordinate is the real part and y coordinate is the imaginary part. So, how to represent the boundary by using the x coordinate and the y coordinate you can see here.

So, corresponding to this point, the x coordinate is x1 and the y coordinate is y1. So, x1 is the real part of the complex number and y1 is the imaginary part of the complex number. So, for this complex number I can determine that Fourier transform, that is called the Fourier descriptors. So, what is the Fourier descriptor, just I am taking the Fourier transform of SK that is the DFT I am taking, SK e to the power minus twice pi ul divided by k, I am determining the polar dress from I am considering.

And by this I can represent the boundary the boundary is SK and corresponding to this boundary I have the Fourier descriptor Fourier descriptor is au. So, that means the boundary is represented by the polar coefficients. So, I have the coefficients a u. So, you can see instead of storing the entire boundary, I have to consider only the Fourier coefficients. And also by using this inverse Fourier transform, I can reconstruct the original boundary that is the reconstruction formula.

So, you can see s k is equal to 1 by K and summation I am taking from k is equal to k minus 1, k number of points and au is the Fourier coefficients e to the power twice pi u k divided by k that I am considering. That is the reconstruction formula.

(Refer Slide Time: 33:33)



Example: Fourier Descriptor

Examples of reconstruction from Fourier descriptors

$$\hat{s}(k) = \frac{1}{K}\sum_{k=0}^{P-1} a(u)e^{2\pi uk/K}$$

P is the number of Fourier coefficients used to reconstruct the boundary

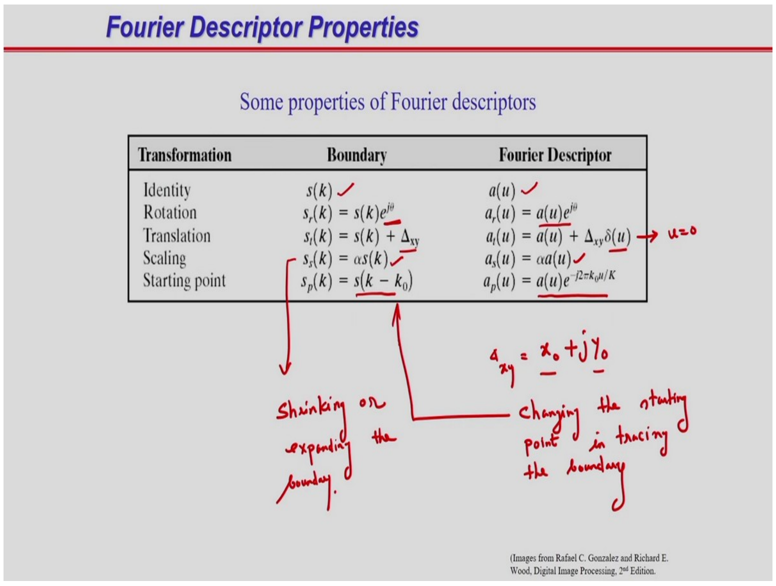
(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

So, in this example, I have shown, one boundary I have shown the boundary you can see the original boundary I am considering that is a square boundary, I want to reconstruct this boundary from the Fourier coefficients. So, original boundary, you can see the 64 points K is equal to 64. So, 64 points are available and by applying the inverse Fourier test from principal I want to reconstruct the original boundary. So, first I am considering only 2 coefficients p is equal to 2 and by using 2 coefficients, here you can see in this reconstruction formula I am considering p is equal to 2 this is p.

So, only by considering 2 coefficients, I want to reconstruct the boundary. So, this is my reconstructed boundary. And again if I consider p is equal to 4, that will be my reconstructed boundary. Next I am considering p is equal to 8, that is my reconstructed boundary. Next I am considering p is equal to 16 and that is my reconstructed boundary. Next I am considering p is equal to 24 like this, I have the reconstructed boundary. And if I consider p is equal to 61 that is my reconstructed boundary.

And if I consider p equal to 62 then you can see I can perfectly reconstruct my original boundary. So, you can see how to reconstruct the original boundary from the Fourier coefficients. So, this is about the Fourier descriptors. Now, main concept is instead of storing the boundary pixels, I can only store the polar coefficients to represent a particular boundary. We need to store all the pixels corresponding to the boundary. So, for this what I can do, I can determine the Fourier descriptors and what I have to store, I have to store the Fourier coefficients in the memory.

(Refer Slide Time: 35:33)



Fourier Descriptor Properties

Some properties of Fourier descriptors

Transformation	Boundary	Fourier Descriptor
Identity	$s(k)$ ✓	$a(u)$ ✓
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a_r(u) = \underline{a(u)}e^{j\theta}$
Translation	$s_t(k) = s(k) + \underline{\Delta_{xy}}$	$a_t(u) = a(u) + \Delta_{xy}\underline{\delta(u)} \rightarrow u=0$
Scaling	$s_s(k) = \alpha s(k)$ ✓	$a_s(u) = \alpha a(u)$ ✓
Starting point	$s_p(k) = \underline{s(k - k_0)}$	$a_p(u) = \underline{a(u)e^{-j2\pi k_0 u/K}}$

$\Delta_{xy} = \underline{x_0} + j\underline{y_0}$
Shrinking or expanding the boundary.
changing the starting point in tracing the boundary

(Images from Rafael C. Gonzalez and Richard E. Wood, Digital Image Processing, 2nd Edition.

And some of the properties of the Fourier descriptors you can see. Corresponding to the boundary SK, my Fourier descriptor is au and the next properties I am considering the rotation. So, rotation of the boundary by an angle theta. The rotation of the boundary by an angle theta causes a constant phase shift of theta in the Fourier descriptors. So, that means the Fourier descriptor is multiplied by again e to the power t theta, that is a constant phase shift of Theta in the Fourier descriptors.

Next, I am considering the translation, the translation by amount delta x y. So translation is delta x y is mainly, delta x y is nothing but in x direction that translation is x naught and in y direction, this translation is y naught, that I am considering. If the boundary is translated by delta x y that is in the x direction it is x naught in y direction it is y naught, then the new Fourier descriptor remains same except at point u is equal to 0, because in this case, I am considering the Dirac delta function.

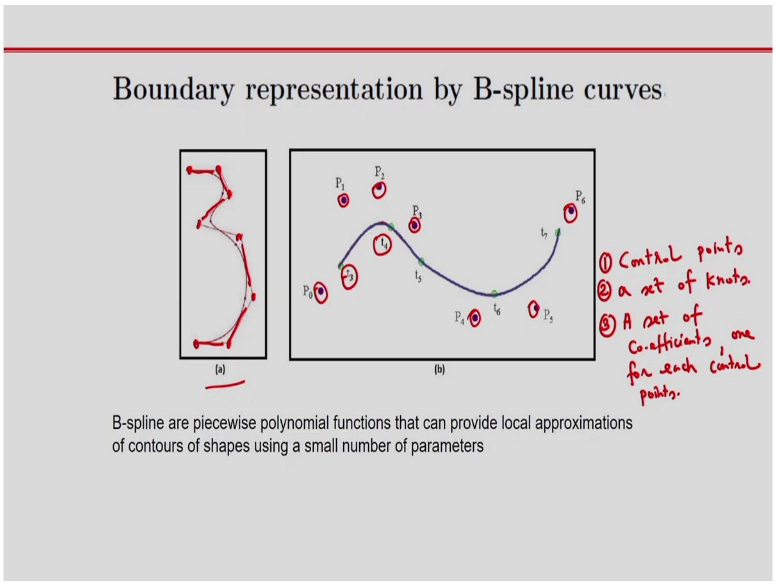
So, corresponding to u is equal to 0 here, so, I will be getting one, for other values it will be 0. So, that is why if the boundaries Translated by x naught in the x direction and the y naught in the y direction, the new Fourier descriptors remain same except at u is equal to 0. The next one is I am considering the scaling, the scaling of the boundary. This is nothing but the shrinking or the expanding the boundary alpha into S x, that is nothing but what I am doing, this is the shrinking or expanding the boundary, that I am considering this the scaling.

So, what is happening if I do the scaling, that is nothing but the scaling of the Fourier descriptors, because the Fourier descriptor is multiplied with alpha. And if I considered changing the starting point here you can see I am changing the starting point, changing the starting point in tracing the boundary what will happen, that means, in this case what I am considering? Changing the starting point in tracing the boundary.

So, what is happening here? It corresponds to the modulation of the Fourier descriptors because au is multiplied with exponential function, that is nothing but the modulation of au. So, these properties I am considering, one is the rotation, so if I consider rotation, what is happening, if I do the rotation of the boundary by an angle theta that corresponds to the constant phase shift of Theta in Fourier descriptors. And if I considered a boundary is translated.

So, except at the point u is equal to 0, it will not since. And also we have considered the concept of the scaling, if I do the scaling the Fourier descriptors will be multiplied by the scale factor and if I change the starting point, then in this case it is nothing but the modulation. So, these are the properties of the Fourier descriptors.

(Refer Slide Time: 39:28)



Boundary representation by B-spline curves

(a)
(b)
P_0
P_1
P_2
P_3
P_4
P_5
P_6
t_3
t_4
t_5
t_6
t_7

① Control points
② a set of knots.
③ A set of Co-efficients, one for each control points.

B-spline are piecewise polynomial functions that can provide local approximations of contours of shapes using a small number of parameters

Now, I will discuss the boundary representation by B spline curve. So, I have shown the figure a I have shown that one boundary. The B spline are piecewise polynomial functions that can provide local approximation of contours of shapes using a small number of parameters. So, because of this B spline, the B spline representation results in compression of the boundary data. So, instead of storing the entire boundary pixels what I can consider, I cannot consider some parameters.

And by using these parameters I can represent a boundary. So, that is why the B spline representation results in compression of boundary data. And in the figure a shows a B spline curve of degree 3, I am considering the B spline curve of degree 3. And it is defined by 8 contour points. So, I have shown the 8 contour points here 1, 2, 3, 4, 5, 6, 7, 8. So, 8 contour points are considered to represent the boundary. These little dots divide the B spline curve into number of curved segments. So, in this case I am getting the number of curve segments you can see this is one segment, this is one segment like this I have the segments.

The subdivision of the curve can also be modified. So, that is why the B spline curve have a higher degree of freedom for curve design. So, based on the contour points, I can represent the object boundary. And as seen in the bigger B, if you see the figure B, to design a B spline curve what actually we need, we need a set of control points. So, you can see we need set of control points, the control points are P0, P1, P2, P3, P4, P5, P6. So, these are the control points.

So, for B spline we need the set of control points and also we need a set of knots. So, what are the things we need one is the control points you need control points we need for B spline representation. Also we need a set of knots. So, here I have shown the knots T1, T 2, T3 like this, I have shown the knots T 3, T 4 like this, these are the knots. Number 3 also we need a set of coefficients, effect of coefficients one for each control points, one for each control points. So, I need this information, one is the control points after this I need a set of knots, I need a set of coefficients one for each control points.

And all the curve segments because you can see that we have the curve segments, these are the segments if you see the segments. All the curved segments are joined together satisfying certain continuity conditions. So, I have to define some continuity conditions and these segments can be joined by considering, by satisfying certain continuity conditions. The degree of the B spline curve or the degree of the B spline polynomial can be adjusted to preserve smoothness of the curve to be approximated.

So, I can consider the degree of the B spline polynomial, maybe I can consider it a second degree, third degree like this I can consider. The curve is approximated by these line segments connected between the control points and in this case the degree of a B spline polynomial I can adjust to preserve smoothness of the curve to be approximated. And the B spline allow the local control over the shape obey spline curve. So, based on these parameters, the control points, the set of knots and a set of coefficients and also the degree of the B spline polynomial I can represent a particular boundary.

So, this concept I am going to discuss in my next class, the concept of the boundary representation by B spline curve. So, in this class I discussed the boundary representation concepts. So, first I discussed the concept of the chain code and the chain code depends on the starting point. So, that is why we considered the first difference of the chain code I have considered. After this I defined the shape number. So, by using the same number I can represent a particular boundary.

After this I discussed one important representation that is the Fourier descriptors. So, by using the Fourier descriptors I can represent a particular boundary. So, instead of storing all the pixels of the boundary, I can only store the Fourier descriptors, that is the Fourier coefficients. And by using these polar coefficients, I can represent the boundary. I can reconstruct the original boundary by considering these Fourier descriptors. So, this is the advantage of the Fourier descriptors.

And also I discussed about the properties of the Fourier descriptors. Next, I have introduced the concept of the B spline curve. So, by using the B spline curve, I can represent a particular boundary. So, how to represent a boundary by using the B spline, that concept I will discuss in my next class. So, let me stop here today. Thank you.