Chapter 6

Image Segmentation

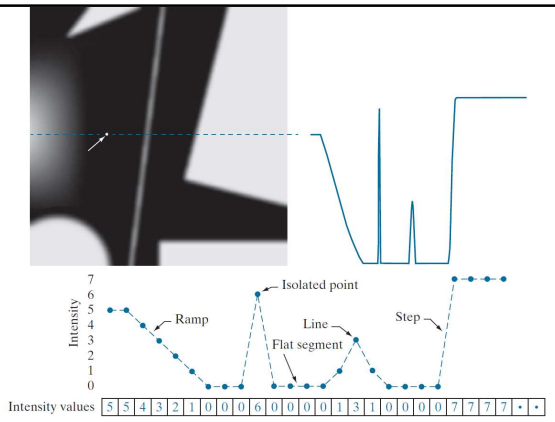
1

Image Segmentation

- Image segmentation process consists of several algorithms for point detection, line detection and edge detection in an image.

1. An isolated **point** may be viewed as a foreground (background) pixel surrounded by background (foreground) pixels.
2. A **line** may be viewed as a thin edge segment in which the intensity of the background on either side of the line is either much higher or much lower than the intensity of the line pixels.
3. **Edge** pixels are pixels at which the intensity of an image changes abruptly**.**

2



3

First order derivatives

- Let $f(x, y)$ be a function of two variables x and y.
- The first order partial derivative $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ are defined in terms of finite differences in following way.

$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

and

$$\frac{\partial f}{\partial y} = f(x, y+1) - f(x, y)$$

4

Second order derivatives

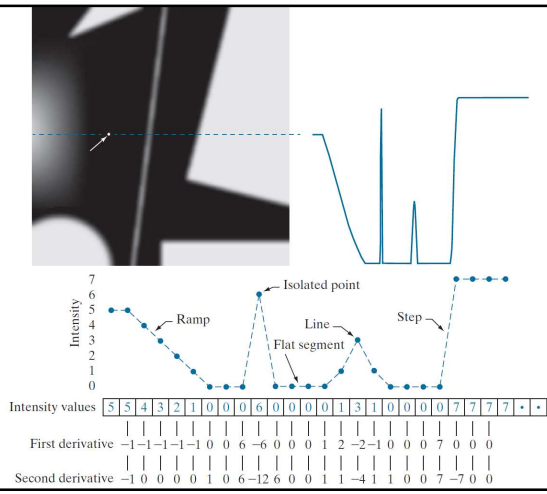
- Let $f(x, y)$ be a function of two variables x and y
- The second order partial derivative $\frac{\partial^2 f}{\partial x^2}$ and $\frac{\partial^2 f}{\partial y^2}$ are defined in terms of finite differences in following way.

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) - 2f(x, y) + f(x-1, y)$$

and

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) - 2f(x, y) + f(x, y-1)$$

5



6

Use of derivatives in image

- First order derivative is non-zero along the ramp.
- Second order derivative is non-zero only at beginning and end of the ramp.
- First-order derivative produces thick edges.
- Second order derivative produces much thinner edge.
- If there are sharp changes, second order derivatives are more responsive.

7

Use of derivatives in image

- First order derivatives have strong response to gray level strip.
- Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points.
- Second-order derivatives produce a double-edge response at ramp and step transitions in intensity.
- The sign of the second derivative can be used to determine whether a transition into an edge is from light to dark or dark to light.

8

Detection of Isolated Points

- Point detection is done using second derivative of Laplacian

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

- Here partial derivatives are computed using the following second-order finite differences

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) - 2f(x,y) + f(x-1,y)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) - 2f(x,y) + f(x,y-1)$$

9

- Using finite differences in Laplacian, we get

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

- This equation can be implemented using following kernel.

0	1	0
1	−4	1
0	1	0

10

- If we also consider two diagonals, then we will have,
$\nabla^2 f = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)$
$+f(x+1,y+1) + f(x+1,y-1) + f(x-1,y+1) + f(x-1,y-1)$
$-8f(x,y)$

- This expression can be implemented using the Laplacian kernel

1	1	1
1	−8	1
1	1	1

11

- The other two widely used kernels are

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

- These are negative of the one we derived earlier.

12

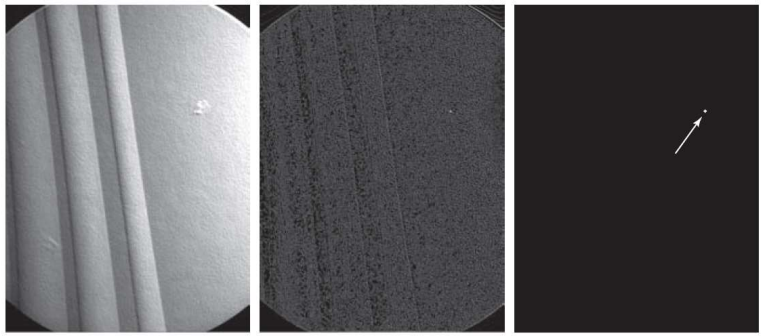
- A point has been detected at a central location (x, y) if the absolute value of the response of the filter at that point exceeds a specified threshold.

$$g(x,y) = \begin{cases} 1 & \text{if } |Z(x,y)| > T \\ 0 & \text{otherwise} \end{cases}$$

- Such points are labeled 1 and all others are labeled 0 in the output image, thus producing a binary image.
- $g(x, y)$ is the output image, T is a nonnegative threshold, and Z is given by

$$Z = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k$$

13



X-ray image of a turbine blade with a porosity

Result of applying Laplacian kernel

Result of Thresholding **Point is detected.**

14

Detection of Lines

- For line detection we can expect second derivatives to result in a stronger filter response, and to produce thinner lines than first derivatives.
- Thus, we can use the Laplacian kernel for line detection also.
- It works with the assumption is that lines are thin with respect to the size of the detector.
- Lines which do not satisfy this assumption handled by the edge detection methods

15

- Sometimes we need to detect lines in specified direction.
- For this purpose, we have many line detector kernels.

−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		

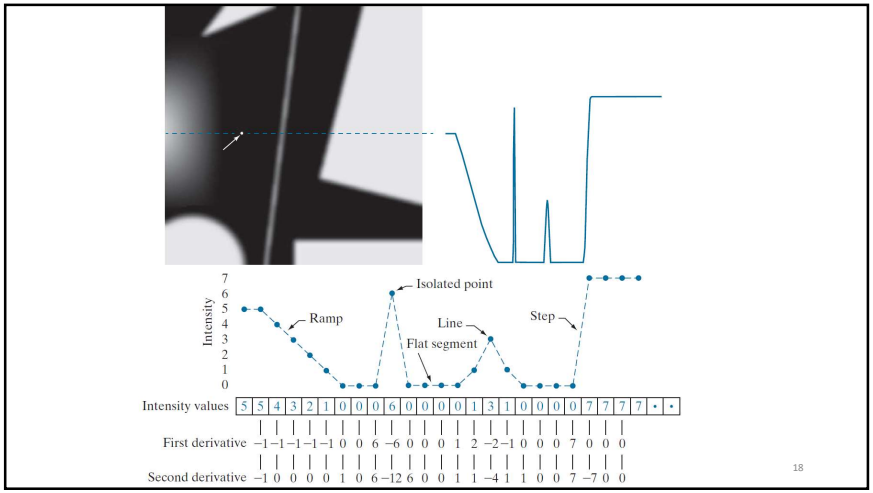
16

Image of a wire-bond template

Result of processing with the +45° line detector kernel

Zoomed view of the top left region

17



18

Edge detection

- Edge detection is an approach used frequently for segmenting images based on abrupt (local) changes in intensity.
- Edge models are classified according to their intensity profiles.
- We can classify edge in following 3 categories
  1. Step edge
  2. Ramp edge
  3. Roof edge

19

1. Step edge

- A step edge is characterized by a transition between two intensity levels occurring ideally over the distance of one pixel.
- Step edges occur in images generated by a computer for use in areas such as solid modeling and animation.
- They can occur over the distance of one pixel.
- They are used as edge models in algorithm development.
- The **Canny edge detection algorithm** is derived using a step-edge model.

20

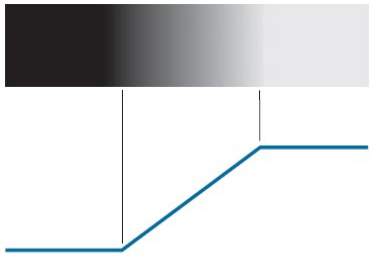
Figure shows a section of a vertical step edge and a horizontal intensity profile through the edge.

21

2. Ramp edge

- In practice, digital images have edges that are blurred and noisy.
- This occurs due to limitations in the focusing mechanism (e.g. lenses) and the noise level.
- In such situations, edges are modeled using an intensity ramp edged.
- The slope of the ramp is inversely proportional to the degree to which the edge is blurred.

22



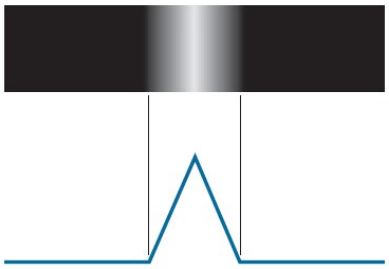
- In this model, we no longer have a single “edge point” along the profile.
- Instead, an edge point now is any point contained in the ramp, and an edge segment would then be a set of such points that are connected.

23

3. Roof edge

- Roof edges are models of lines through a region.
- The base (width) of the edge is determined by the thickness and sharpness of the line.
- When its base is one pixel wide, a roof edge is nothing more than a one-pixel-thick line running through a region in an image.
- Areas in which roof edges appear routinely are in the digitization of line drawings and also in satellite images, where thin features, such as roads, can be modeled by this type of edge.

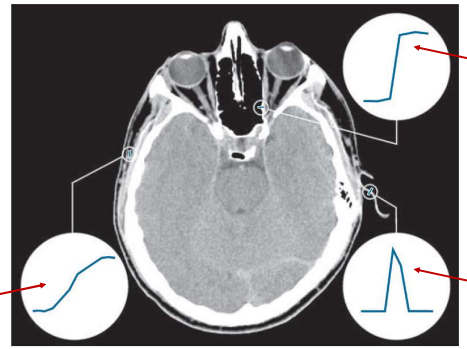
24



- Roof edges arise in range imaging, when thin objects (such as pipes) are closer to the sensor than the background (such as walls).
- The pipes appear brighter.

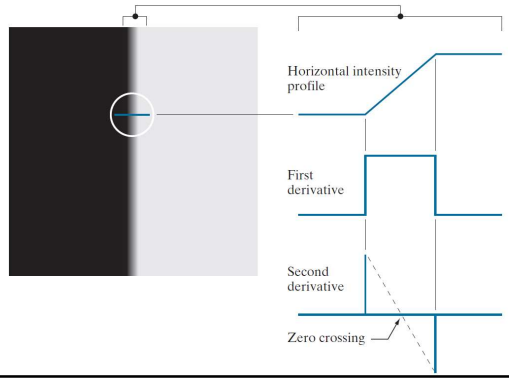
25

- Images can contain all three types of edges.



26

Use of Derivatives to detect Edges



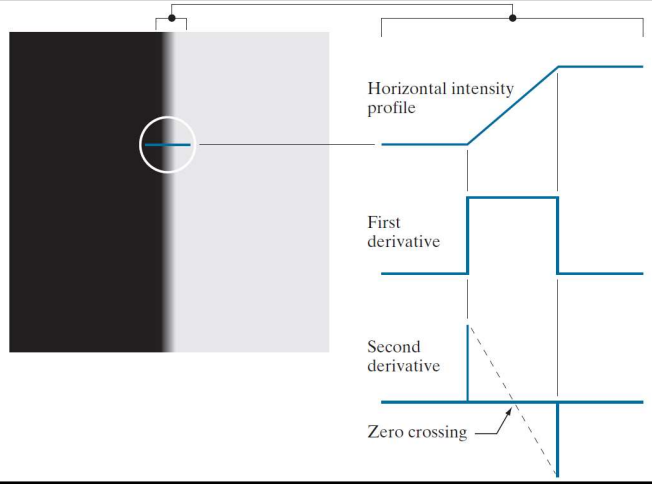
27

- Above Figure shows the image and its horizontal intensity profile.
- This figure also shows the first and second derivatives of the intensity profile.
- Moving from left to right along the intensity profile, we note that the first derivative is positive at the onset of the ramp and at points on the ramp, and it is zero in areas of constant intensity.
- The second derivative is positive at the beginning of the ramp, negative at the end of the ramp, zero at points on the ramp, and zero at points of constant intensity.

28

- The signs of the derivatives would be reversed for an edge that transitions from light to dark.
- The intersection between the zero intensity axis and a line extending between the extrema of the second derivative marks a point called the **zero crossing** of the second derivative.

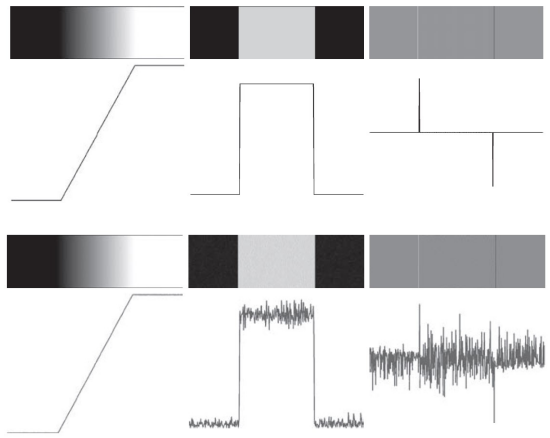
29



30

- We conclude from these observations that
  1. The **magnitude of the first derivative** can be used to detect the presence of an edge at a point in an image.
  2. **Sign of the second derivative** can be used to determine whether an edge pixel lies on the dark or light side of an edge.
  3. **Second derivative** produces two values for every edge in an image.
  4. **Zero crossings** of second derivative can be used for locating the centers of thick edges.

31



It is important to remove noise from image before applying the Edge detection algorithm.

32

Steps to perform edge detection

1. **Image smoothing for noise reduction**.
2. **Detection of edge points**. As mentioned earlier, this is a local operation that extracts from an image all points that are potential edge-point candidates.
3. **Edge localization**. The objective of this step is to select from the candidate points only the points that are members of the set of points comprising an edge.

33

Edge detection technique

- Detecting changes in intensity for the purpose of finding edges can be accomplished using first or second order derivatives.
- The tool of choice for finding edge strength and direction at an arbitrary location (x, y) of an image, f, is the gradient, denoted by ∇f and defined as the vector

$$\nabla f(x,y) \equiv \text{grad}\left[f(x,y)\right] \equiv \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix}$$

34

- ∇f has the well-known property that it points in the direction of maximum rate of change of f at (x, y).
- Above equation is valid at an arbitrary single point (x, y).
- When evaluated for all applicable values of x and y, $f(x, y)$ becomes a vector image, each element of which is a vector given by ∇f.
- The magnitude, $M(x, y)$, of this gradient vector at a point (x, y) is given by its Euclidean vector norm:

$$M(x,y) = \left\|\nabla f(x,y)\right\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$

- This is the value of the rate of change in the direction of the gradient vector at point (x, y).

35

- $M(x, y)$, $\|\nabla f(x, y)\|$, $g_x(x, y)$, and $g_y(x, y)$ are arrays of the same size as f, created when x and y are allowed to vary over all pixel locations in f.
- $M(x, y)$ and $\|\nabla f(x, y)\|$ is known as the gradient image.
- The summation, square, and square root operations are elementwise operations.
- The direction of the gradient vector at a point (x, y) is given by

$$\alpha(x,y) = \tan^{-1}\left[\frac{g_y(x,y)}{g_x(x,y)}\right]$$

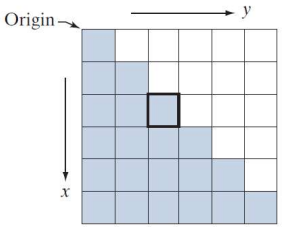
- Angles are measured in the counterclockwise direction with respect to the x-axis.

36

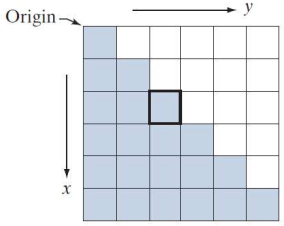
- $\alpha(x, y)$ is also an image of the same size as f, created by the elementwise division of g_x and g_y over all applicable values of x and y.

Computing the gradient

- Figure shows a zoomed section of an image containing a straight edge segment.
- Each square corresponds to a pixel.
- We are interested in obtaining the strength and direction of the edge at the point highlighted with a box.
- The shaded pixels in this figure are assumed to have value 0, and the white pixels have value 1.



- Derivatives in the x and y directions are computed using a 3×3 neighborhood centered at a point.
- Partial derivative in the x direction is obtained by subtracting the pixels in the top row of the neighborhood from the pixels in the bottom row.
- Partial derivative in the y direction is obtained by subtracting the pixels in the left column from the pixels in the right column of the neighborhood.



- Using these differences as our estimates of the partials we get $\frac{\partial f}{\partial x} = -2$ and $\frac{\partial f}{\partial y} = 2$ at the point in question. Then

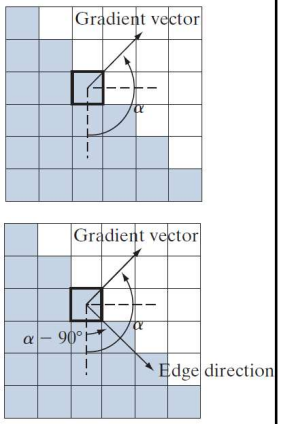
$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

- From this we get, $\|\nabla f\| = 2\sqrt{2}$ at that point.
- The direction of the gradient vector at that point is

$$\alpha = \tan^{-1}(-2/2) = \tan^{-1}(-1) = -45°$$

- This is the same as 135° measured in the positive (counterclockwise) direction with respect to the x-axis in our image coordinate system.

- The direction of an edge at a point is orthogonal to the gradient vector at that point.
- So the direction angle of the edge in this example is $a - 90° = 135° - 90° = 45°$.
- All edge points have the same gradient, so the entire edge segment is in the same direction.
- The gradient vector is sometimes normalized to unit length by dividing it by its magnitude.



41

Gradient Operators

- Obtaining the gradient of an image requires computing the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at every pixel location in the image.
- For the gradient, we use a forward or centered finite difference.
- Using forward differences we obtain

$$g_x(x,y) = \frac{\partial f(x,y)}{\partial x} = f(x+1,y) - f(x,y)$$

−1
1

$$g_y(x,y) = \frac{\partial f(x,y)}{\partial y} = f(x,y+1) - f(x,y)$$

−1	1

42

- These two derivatives can be implemented in an image using kernel

−1
1

and

−1	1

- When diagonal edge direction is needed, we need 2-D kernels.
- The Roberts cross-gradient operators are used in this case.
- Consider the 3×3 region

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

43

- The Roberts operators are based on implementing the following diagonal differences.

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

- These derivatives can be implemented by filtering an image with the kernels

−1	0
0	1

0	−1
1	0

44

- But Kernels of size 2×2 are not as useful for computing edge direction.
- We required kernels of size at least 3×3 because these kernels take into account the nature of the data on opposite sides of the center point.
- Partial derivatives using kernels of size 3×3 are given by

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

45

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

- The difference between the third and first rows approximates the derivative in the x direction.
- The difference between the third and first columns approximate the derivative in the y direction.
- These kernels are called the **Prewitt operators**.
- They more accurate than the **Roberts operators**.

46

- Prewitt operators

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

47

- A slight variation of the Prewitt operators uses a weight of 2 in the center coefficient.

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

- Using a 2 in the center location provides image smoothing.
- These kernels are called the **Sobel operators**.

48

- Sobel operators

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

49

- The Prewitt kernels are simpler to implement than the Sobel kernels.
- The slight computational difference between them typically is not an issue.
- Sobel kernels have better noise-suppression (smoothing) characteristics.
- Sobel kernels are more preferable because noise suppression is an important issue when dealing with derivatives.
- The coefficients of all the kernels sum to zero.
- So it gives a response of zero in areas of constant intensity, as expected of derivative operators.

50

- These two partial derivative arrays are then used to estimate edge strength and direction.
- We can find magnitude of the gradient using

$$M(x,y) = \|\nabla f(x,y)\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$$

- This implementation is not always desirable because of the computational burden required by squares and square roots.
- An approach used frequently is to approximate the magnitude of the gradient by absolute values is

$$M(x,y) \approx |g_x| + |g_y|$$

- This equation is more attractive computationally, and it still preserves relative changes in intensity levels.

51

- Prewitt kernel and Sobel kernel are good for vertical and horizontal edges.
- The Kirsch compass kernels are designed to detect edge magnitude and direction (angle) in all eight compass directions.

−3	−3	5
−3	0	5
−3	−3	5

N

−3	5	5
−3	0	5
−3	−3	−3

NW

5	5	5
−3	0	−3
−3	−3	−3

W

5	5	−3
5	0	−3
−3	−3	−3

SW

5	−3	−3
5	0	−3
5	−3	−3

S

−3	−3	−3
5	0	−3
5	5	−3

SE

−3	−3	−3
−3	0	−3
5	5	5

E

−3	−3	−3
−3	0	5
−3	5	5

NE

52

Edge Detection example

53

Image of size 834×1114 pixels, with intensity values scaled to the range $[0,1]$.

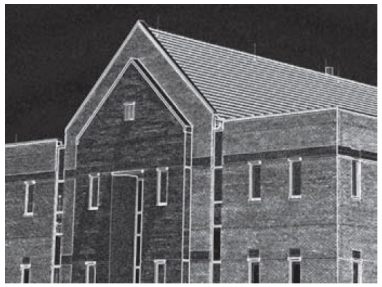
54

The component of the gradient $|g_x|$ in the x direction, obtained using the Sobel kernel

55

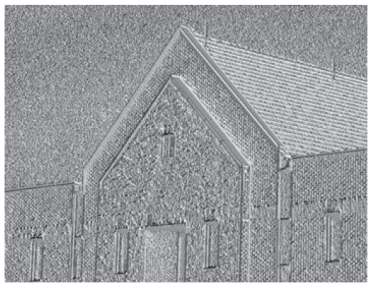
The component of the gradient $|g_y|$ in the y direction, obtained using the Sobel kernel

56



The gradient image
$|g_x| + |g_y|$

57



Gradient angle image. Areas of constant intensity in this image indicate that the direction of the gradient vector is the same at all the pixel locations in those regions.

58

Effect of smoothing on Edge detection

59

Original image smoothed first using a 5×5 averaging filter

60

The component of the gradient $|g_x|$ in the x direction, obtained using the Sobel kernel

61

The component of the gradient $|g_y|$ in the y direction, obtained using the Sobel kernel

62

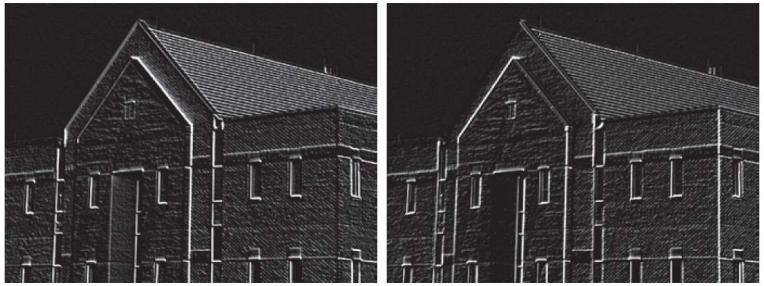
The gradient image
$|g_x| + |g_y|$
Edge detection is better because of smoothing

63

Diagonal edge detection

- The horizontal and vertical Sobel kernels do not differentiate between edges in the $\pm 45°$ directions.
- If it is important to emphasize edges oriented in particular diagonal directions, then one of the Kirsch kernels should be used.

64



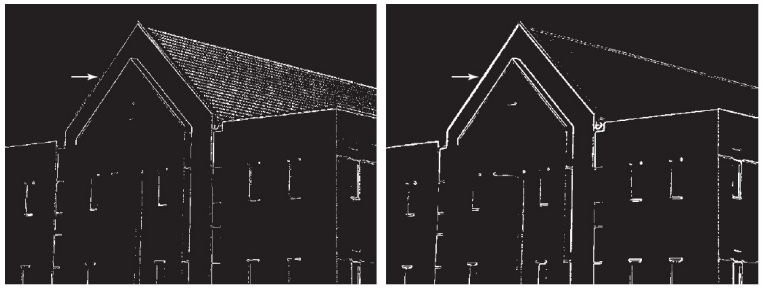
Figures show the responses of the 45° (NW) and −45° (SW) Kirsch kernels, respectively.

65

Combining the Gradient with Thresholding

- Edge detection can be made more selective by smoothing the image prior to computing the gradient.
- To do this, we can threshold the gradient image.
- Pixels having values greater than or equal some threshold value are shown in white, while pixels below the threshold value are shown in black.
- Due to this, the edges obtained in the image are much sharper.

66



(a) Result of thresholding on the gradient of the original image.
(b) Result of thresholding on the gradient of the smoothed image.

67

The Canny Edge Detector

- The algorithm is more complex, the performance of the Canny edge detector for detecting edges is reasonable.
- It is superior in general to the edge detectors
- It was introduced by Canny in 1986.
- It can detect edges in all directions.

68

- Edge detection using Canny algorithm


69

Segmentation by region

- The objective of segmentation is to partition an image into regions.
- In earlier topics, we approached this problem by attempting to find boundaries between regions based on discontinuities in intensity levels (Point, Line, Edge detection).
- Now we will discuss segmentation techniques that find the regions directly. It includes
  1. Region growing
  2. Region splitting and merging
  3. Region segmentation using k-means clustering

70

1. Region growing

- Region growing is a procedure that groups pixels or sub-regions into larger regions based on predefined criteria for growth.
- Here we start with a set of "seed" points.
- Then we grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as ranges of intensity or color).
- Selecting a set of one or more starting points (seeds) can often be based on the nature of the problem.

71

- Let $f(x, y)$ denote an input image.
- Let $S(x, y)$ denote a seed array containing 1's at the locations of seed points and 0's elsewhere.
- f and S are of same size.
- Let Q denote a predicate to be applied at each location (x, y).
- A basic region-growing algorithm based on 8-connectivity may be stated using following steps.

1. Find all connected components in $S(x, y)$ and reduce each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at each point (x, y), $f_Q(x, y) = 1$ if the input image satisfies a given predicate, Q, at those coordinates, and $f_Q(x, y) = 0$ otherwise.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.
4. Label each connected component in g with a different region label (e.g.,integers or letters). This is the segmented image obtained by region growing.

72

Segmentation by region growing

- Figure A shows an 8-bit X-ray image of a weld (the horizontal dark region) containing several cracks and porosities (the bright regions running horizontally through the center of the image).
- We illustrate the use of region growing by segmenting the defective weld regions.
- These regions could be used in applications such as weld inspection or for controlling an automated welding system.

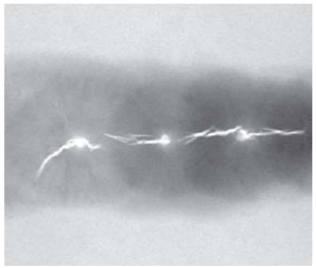


Figure A

73

- The first thing we do is determine the seed points.
- Regions containing Cracks and porosities appears to be significantly brighter than other parts of the X-ray image.
- We can extract the seed points by thresholding the original image.
- Figure B shows the histogram of the image.
- Figure C shows the threshold result obtained with a threshold equal to 254.

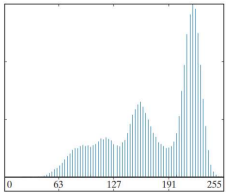


Figure B

Figure C

74

- Figure D shows the result of each connected component to a single point. These are our seed points.
- Next, we have to specify a predicate. In this example, we are interested in appending to each seed all the pixels that are 8-connected to that seed, and are "similar" to it.

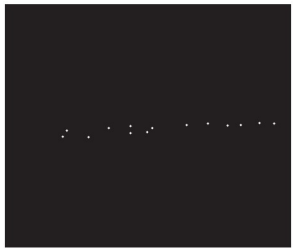


Figure D

- So the predicate applied at each location (x, y) is

$$Q = \begin{cases} \text{TRUE} & \text{if the absolute difference of intensities between the seed and the pixel at } (x, y) \text{ is} \le T \\ \text{FALSE} & \text{otherwise} \end{cases}$$

75

- We know that all seed values are 255 because the image was thresholded with a threshold of 254.
- Figure E shows the difference between the seed value (255) and Figure A.
- The image in Figure E contains all the differences needed to compute the predicate at each location.
- Figure F shows the corresponding histogram

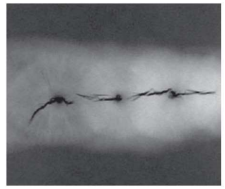


Figure E

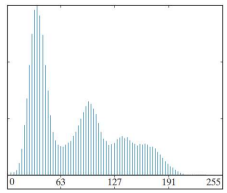


Figure F

76

- We need a threshold to use in the predicate to establish similarity.
- We can do this by applying to the difference image the dual thresholding.
- The resulting two thresholds in this case were T1 = 68 and T2 = 126, which we see correspond closely to the valleys of the histogram.
- So we segmented the image using these two thresholds.
- The result is shown in Figure G.

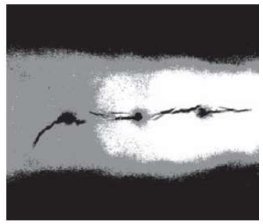


Figure G

77

- Figure G shows that segmenting the defects cannot be accomplished using dual thresholds, despite the fact that the thresholds are in the deep valleys of the histogram.
- Figure H shows the result of thresholding the difference image with only T1.
- The black points are the pixels for which the predicate was TRUE.

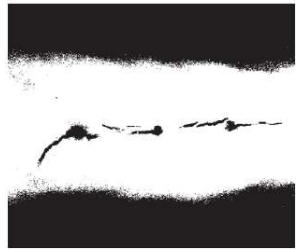


Figure H
Difference image thresholded with the smallest of the dual thresholds

78

- The points in the outer region will be considered by the region-growing algorithm as candidates.
- However, Step 3 will reject the outer points because they are not 8-connected to the seeds.
- Figure H shows, this step resulted in the correct segmentation, indicating that the use of connectivity was a fundamental requirement in this case.
- Finally, note that in Step 4 we used the same value for all the regions found by the algorithm.

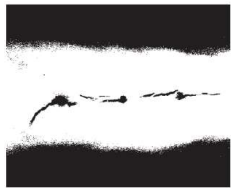


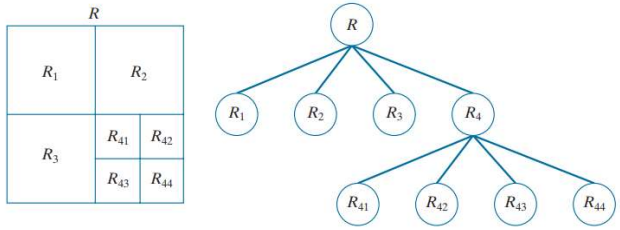
Figure H

Figure I

Region splitting and merging

- In this method, an image is subdivided into a set of disjoint regions and then the regions are merged and/or split in an attempt to satisfy the conditions of segmentation.
- Let R represent the entire image region. Let Q be a predicate.
- One approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $Q(R_i) = True$.
- We start with the entire region R. If $Q(R) = FALSE$, We divide the image into quadrants. If Q is FALSE for any quadrant, we subdivide that quadrant into sub-quadrants, and so on.

80



- This splitting technique can be represented by a tree in which each vertex has exactly four descendants, as shown in the figure.
- The root of the tree corresponds to the entire image.
- Each vertex corresponds to the subdivision of a vertex into four descendant nodes.
- In this case, only R_4 was subdivided further.

81

- If only splitting is used, the final partition normally contains adjacent regions with identical properties.
- This drawback can be remedied by allowing merging as well as splitting.
- Satisfying the constraints of segmentation requires merging only adjacent regions whose combined pixels satisfy the predicate Q.
- That is, two adjacent regions R_j and R_k are merged only if

$$Q(R_j \cap R_k) = TRUE.$$

82

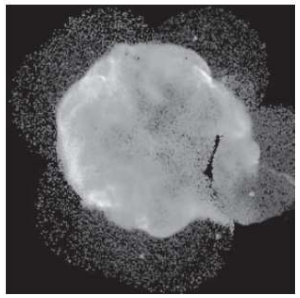
Steps:

1. Split any region R_i into four disjoint quadrants for which $Q(R_i) = FALSE$.
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cap R_k) = TRUE$.
3. Stop when no further merging is possible

83

Example

- Figure A shows a 566 x 566 X-ray image of the Cygnus Loop supernova, taken by Hubble space Telescope.
- The objective is to segment the "ring" of less dense matter surrounding the dense inner region.



84

- Here the data in this region of interest has a random nature, so its standard deviation should be greater than the standard deviation of the background (which is near 0) and of the large central region, which is smooth.
- Similarly, the average intensity of a region containing data from the outer ring should be greater than the mean of the darker background and less than the mean of the lighter central region.
- Thus, we can segment the region of interest using the following predicate:

$$Q(R) = \begin{cases} \text{TRUE} & \text{if } \sigma_R > a \text{ AND } 0 < m_R < b \\ \text{FALSE} & \text{otherwise} \end{cases}$$

- Here σ_R and m_R are the standard deviation and mean of the region being processed, and a and b are nonnegative constants.

85

- From the analysis it is found that mean intensity of pixels in those regions did not exceed 125, and the standard deviation was always greater than 10.
- Figures B, C, D show the results obtained using these values for a and b, and varying the size sub regions from 32 to 8.

Figure B (32x32)

Figure C (16x16)

Figure D (8x8)

86

- The pixels in a sub-regions that satisfied the predicate were set to white; all others in that region were set to black.
- The best result in terms of capturing the shape of the outer region was obtained using sub-regions of size 16 x 16.
- The small black squares in Figure D are sub-regions of size 8 x 8 whose pixels did not satisfy the predicate.

Figure B (32x32)

Figure C (16x16)

Figure D (8x8)

87

- Using smaller sub-regions would result in increasing numbers of such black regions.
- Using larger region would result in a more “block-like” segmentation.
- In all cases the segmented region (white pixels) was a connected region that completely separates the inner, smoother region from the background.
- So the segmentation effectively partitioned the image into three distinct areas that correspond to background, a dense region, and a sparse region.

88

Region segmentation using k-means clustering

- The basic idea behind the clustering approach is to partition a set, Q, of observations into a specified number, k, of clusters.
- In k-means clustering, each observation is assigned to the cluster with the nearest mean.
- Each mean is called the prototype of its cluster.
- A k-means algorithm is an iterative procedure that successively refines the means until convergence is achieved.

89

- Let $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_Q\}$ be set of vector observations (samples).
- These vectors have the form $\boldsymbol{Z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$.
- In image segmentation, each component of a vector $\boldsymbol{Z}$ represents a numerical pixel attribute.
- For example, if segmentation is based on just grayscale intensity, then $\boldsymbol{Z} = [z]$ is a scalar representing the intensity of a pixel.
- If we are segmenting RGB color images, $\boldsymbol{Z}$ typically is a 3-D vector.

90

- The objective of k-means clustering is to partition the set Q of observations into $k \leq Q$ disjoint cluster sets $C = \{C_1, C_2, \ldots, C_k\}$ so that the following criterion of optimality is satisfied:

$$\arg\min_{C} \left(\sum_{i=1}^{k} \sum_{\mathbf{z} \in C_i} \|\mathbf{z} - \mathbf{m}_i\|^2 \right)$$

- Here $\boldsymbol{m}_i$ is the mean vector (or centroid) of the samples in set C_i.
- $\|\cdot\|$ is a Euclidean norm. So $\|\boldsymbol{z} - \boldsymbol{m}_i\|$ is the is the Euclidean distance from a sample in C_i to center $\boldsymbol{m}_i$.
- This equation says that we are interested in finding the sets $C = \{C_1, C_2, \ldots, C_k\}$ such that the sum of the distances from each point in a set to the mean of that set is minimum.

91

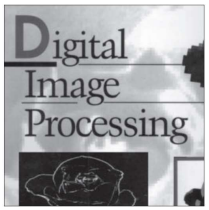
Given a set $\{\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_Q\}$ of vector observation and a specified value of k, the algorithm is as follows:

1. Specify an initial set of means $\boldsymbol{m}_i(1), i = 1, 2, \ldots, k$.
2. Assign each sample to the cluster set whose mean is the closest.
3. Update the cluster centers (means) using $\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \quad i = 1, 2, \ldots, k$
4. Compute the Euclidean norms of the differences between the mean vectors in the current and previous steps. Compute error E, as the sum of the k norms. Stop if $E \leq T$, where T a threshold. Else, go back to Step 2.

92

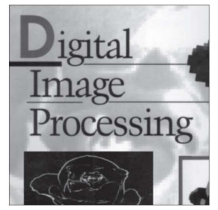
Example

- Figure A shows an image of size 688 x 688.
- Figure B is the segmentation obtained using the k-means algorithm with $k = 3$.
- The algorithm was able to extract all the meaningful regions of this image with high accuracy.



93

- Compare the quality of the characters in both images.
- Here entire segmentation was done by clustering of a single variable (intensity).
- Because k-means works with vector observations, its power to discriminate between regions increases as the number of components of vector ***Z*** increases.



94