Digital Image Processing

Chap 10: Image Segmentation

清華大學電機系林嘉文
cwlin@ee.nthu.edu.tw
Tel: 03-5731152

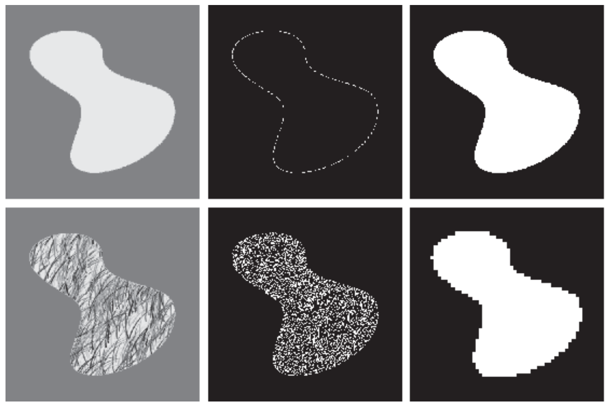
Image Segmentation

- Segmentation subdivides an image into its constituent **regions** or **objects**.
- Segmentation is based on the **discontinuity** and **similarity**.
- **Discontinuity**: abrupt changes in intensity, such as edges.
- **Similarity**: partitioned into similar regions according to a set of predefined criteria, such as thresholding, region growing, region splitting and merging.

Image Segmentation

a b c
d e f

FIGURE 10.1
(a) Image of a constant intensity region.
(b) Boundary based on intensity discontinuities.
(c) Result of segmentation.
(d) Image of a texture region.
(e) Result of intensity discontinuity computations (note the large number of small edges).
(f) Result of segmentation based on region properties.



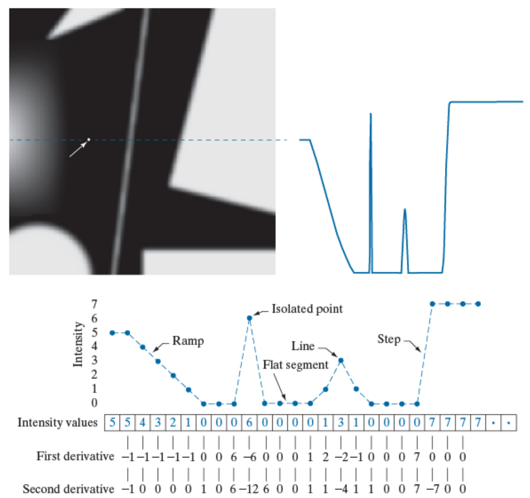
Point, Line and Edge Detection

- Image features: Local **changes** in intensity
- **Isolated points**
- **Lines**
- **Edges**
  - Edge points
  - Edge segment: a set of connected edge pixels
- $\partial f/\partial x = f'(x) = f(x+1) - f(x)$
- $\partial^2 f/\partial x^2 = \partial f'(x)/\partial x$
  $= [f(x+1) - f(x)] - [f(x) - f(x-1)]$
  $= f(x+1) - 2f(x) + f(x-1)$

Point, Line and Edge Detection

a b
c

FIGURE 10.2
(a) Image.
(b) Horizontal intensity profile that includes the isolated point indicated by the arrow.
(c) Subsampled profile; the dashes were added for clarity. The numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10-4) for the first derivative and Eq. (10-7) for the second.



Point, Line and Edge Detection

- An edge is modeled as a “meaningful” transitions in gray-levels.
- **Ideal (or Step) edge**
- **Ramp edge**
- **Roof edge**
- 1st derivative and 2nd derivative on the edge profile.
- The 1st derivative: Detect the existence of an edge.
- The 2nd derivative: “Zero-crossing” property is used to identify the location of edge.

Point, Line and Edge Detection

- 1^{st} order derivative produces thicker edges.
- 2^{nd} order derivative has a strong response to fine details, such as thin lines, isolated points, and noise.
- 2^{nd} order derivative produce **double-edge** response at ramp and step transition
- 2^{nd} order derivative can determine transitions **from light to dark** or **from dark to light**.

Point, Line and Edge Detection

- The response of the mask at any point in the image is given by $R = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9$

FIGURE 10.3
A general 3 × 3 spatial filter kernel. The w's are the kernel coefficients (weights).

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Laplacian Operator

$$|\nabla^2 f| = |\partial^2 f/\partial x^2 + \partial^2 f/\partial y^2|$$
$$= |[f(x+1,y) - 2f(x,y) + f(x-1,y)] +$$

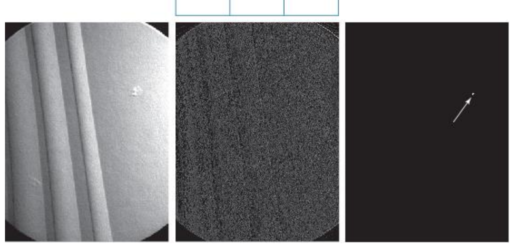
0	1	0
1	−4	1
0	1	0

Detection of Isolated points

- Point detection: detect isolated points
- A point is detected if $|R| \geq T$

Laplacian operator

1	1	1
1	−8	1
1	1	1



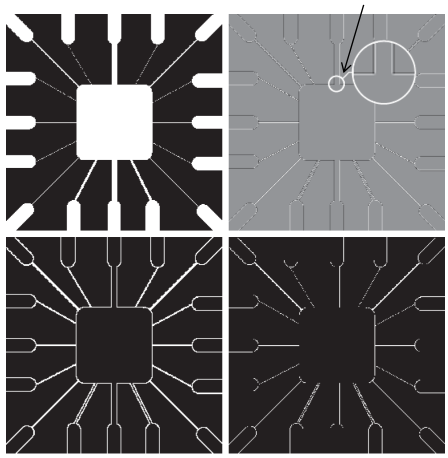
a
b c d

FIGURE 10.4 (a) Laplacian kernel used for point detection. (b) X-ray image of a turbine blade with a porosity manifested by a single black pixel. (c) Result of convolving the kernel with the image. (d) Result of using Eq. (10-15) was a single point (shown enlarged at the tip of the arrow). (Original image courtesy of X-TEK Systems, Ltd.)

Line Detection

Line detection:

Double lines



a b
c d

FIGURE 10.5 (a) Original image. (b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian. (c) Absolute value of the Laplacian. (d) Positive values of the Laplacian.

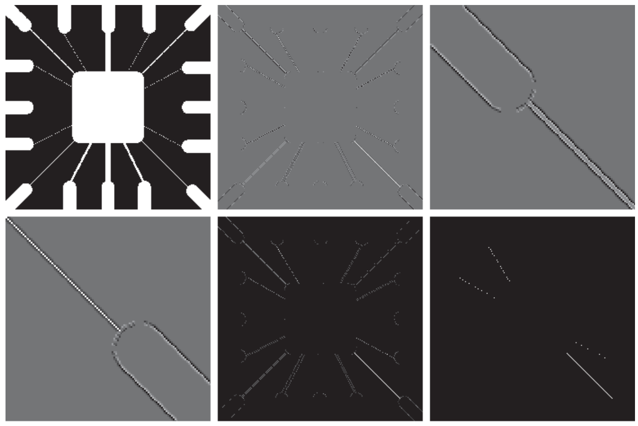
Line Detection Masks

−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°	

a b c d

FIGURE 10.6 Line detection kernels. Detection angles are with respect to the axis system in Fig. 2.19, with positive angles measured counterclockwise with respect to the (vertical) x-axis.

Line Detection



a b c
d e f

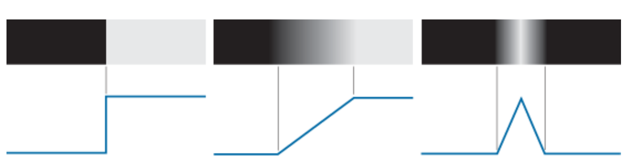
FIGURE 10.7 (a) Image of a wire-bond template. (b) Result of processing with the +45° line detector kernel in Fig. 10.6. (c) Zoomed view of the top left region of (b). (d) Zoomed view of the bottom right region of (b). (e) The image in (b) with all negative values set to zero. (f) All points (in white) whose values satisfied the condition $g > T$, where g is the image in (e) and $T = 254$ (the maximum pixel value in the image minus 1). (The points in (f) were enlarged to make them easier to see.)

Edge Models

- Edge models are classified based on the intensity profile.
- Step edge
- Ramp edge
- Roof edge

a b c

FIGURE 10.8 From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.



Edge Models

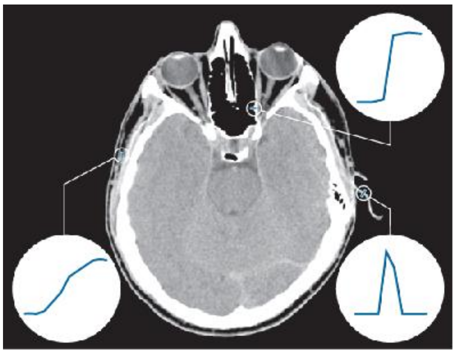
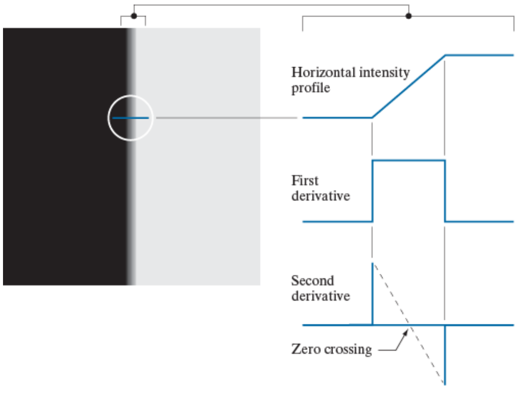


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas enclosed by the small circles. The ramp and step profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

Edge Models

The 1st derivative : $\nabla\mathbf{f} = [G_x, Gy] = [\partial f/\partial x, \partial f/\partial y]$
The 2nd derivative : $\nabla^2\mathbf{f} = [\partial^2 f/\partial x^2, \partial^2 f/\partial y^2]$

a b
FIGURE 10.10 (a) Two regions of constant intensity separated by an ideal ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, and its first and second derivatives.



Edge Models

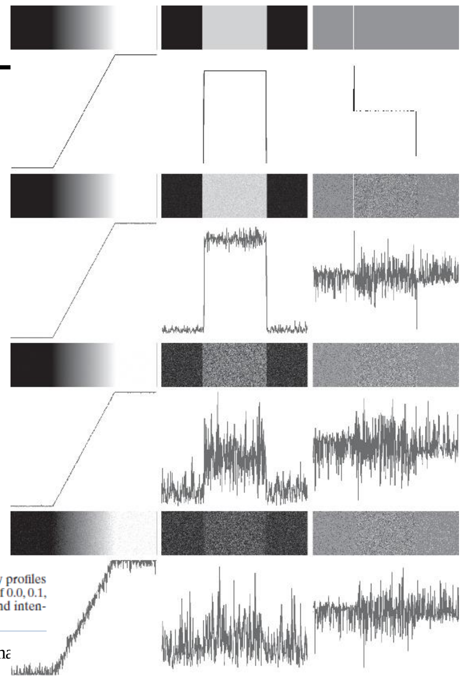
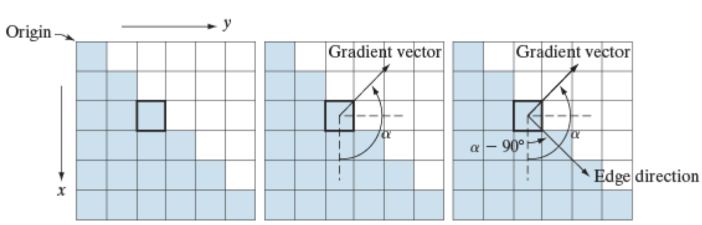


FIGURE 10.11 First column: 8-bit images with values in the range [0,255], and intensity profiles of a ramp edge corrupted by Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

Basic Edge Detection



a b c

FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge direction is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square represents one pixel. (Recall from Fig. 2.19 that the origin of our coordinate system is at the top, left.)

Edge Models

- Apply gradient operators on an image $f(x,y)$ at location (x,y) to obtain a 2-D gradient defined as:

$$\nabla \mathbf{f} = [G_x, Gy] = [\partial f/\partial x, \partial f/\partial y]$$

$$\partial f/\partial x = f(x+1,y) - f(x,y)$$

$$\partial f/\partial x = f(x,y+1) - f(x,y)$$

−1
1

−1	1

a b
FIGURE 10.13
1-D kernels used to implement Eqs. (10-19) and (10-20).

- The **magnitude** of this vector is

$$\nabla f = \mathrm{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}$$

- The **direction** is $\alpha(x,y) = \tan^{-1}(G_x/G_y)$

Basic Edge Detection

- Robert **cross-gradient** operators:

$$G_x = (z_9 - z_5) \text{ and } G_y = (z_8 - z_6)$$

- It does not have a clear center.
- Prewitt 3x3 operators

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \text{ and}$$
$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

- Weighted Prewitt (Sobel) 3x3 operators

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \text{ and}$$
$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

- The gradient is $\nabla f \cong |G_x| + |G_y|$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

Basic Edge Detection

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

Roberts

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

a
b c
d e
f g

FIGURE 10.14 A 3×3 region of an image (the z's are intensity values), and various kernels used to compute the gradient at the point labeled z_5.

Basic Edge Detection

−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

a b c d
e f g h

FIGURE 10.15 Kirsch compass kernels. The edge direction of strongest response of each kernel is labeled below it.

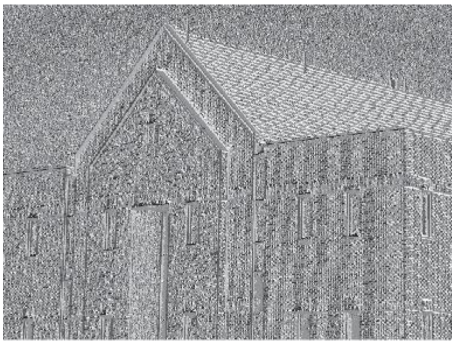
Basic Edge Detection

a b
c d

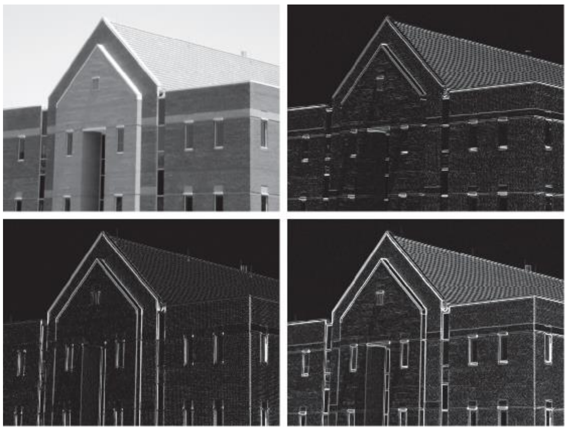
FIGURE 10.16
(a) Image of size 834 × 1114 pixels, with intensity values scaled to the range [0,1]. (b) $|g_x|$, the component of the gradient in the x-direction, obtained using the Sobel kernel in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the kernel in Fig. 10.14(g). (d) The gradient image, $|g_x| + |g_y|$.

Basic Edge Detection

FIGURE 10.17
Gradient angle image computed using Eq. (10-18). 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.



Basic Edge Detection



a b
c d

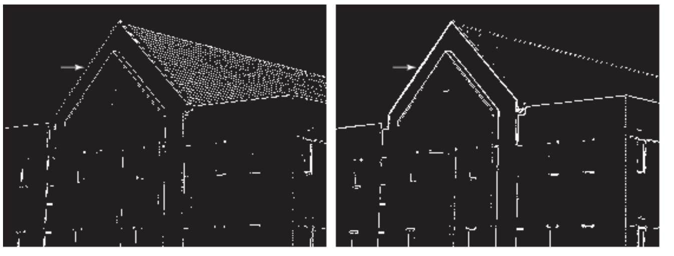
FIGURE 10.18 Same sequence as in Fig. 10.16, but with the original image smoothed using a 5×5 averaging kernel prior to edge detection.

Basic Edge Detection

a b

FIGURE 10.19 Diagonal edge detection. (a) Result of using the Kirsch kernel in Fig. 10.15(c). (b) Result of using the kernel in Fig. 10.15(d). The input image in both cases was Fig. 10.18(a).

Basic Edge Detection



a b

FIGURE 10.20
(a) Result of thresholding Fig. 10.16(d), the gradient of the original image.
(b) Result of thresholding Fig. 10.18(d), the gradient of the smoothed image.

More Advanced Edge Detection

- The Laplacian operator:
  $\nabla^2 f = [\partial^2 f / \partial x^2, \partial^2 f / \partial y^2]$
- Digital approximation for 3x3 region is
  $\nabla^2 f = 4z_5 - (z_2 + z_4 + z_6 + z_8)$
  or $\nabla^2 f = 8z_5 - (z_2 + z_4 + z_6 + z_8 + z_1 + z_3 + z_7 + z_9)$
- The Laplacian is very sensitive to noise

Laplacian of a Gaussian (LoG)

- **Laplacian** plus **Gaussian** (smoothing)

  Gaussian: $G(r) = e^{-\frac{r^2}{2\sigma^2}}$ where $r^2 = x^2 + y^2$
- Convolving the Gaussian with the image will blur the image.
- The Laplacian of $G(r)$ is

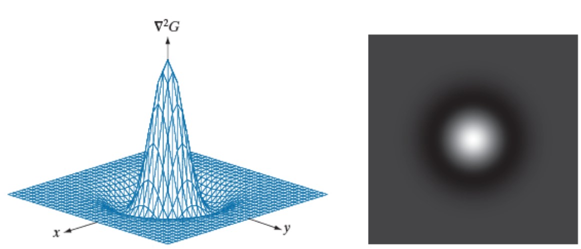
$$\nabla^2 G(r) = \left[\frac{r^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{r^2}{2\sigma^2}} = \left[\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

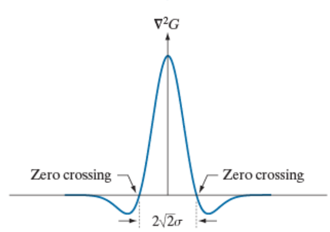
- It is called the Laplacian of a Gaussian (LoG) which is also called **Mexican hat function**

Laplacian of a Gaussian (LoG)

a b
c d

FIGURE 10.21
(a) 3-D plot of the *negative* of the LoG.
(b) Negative of the LoG displayed as an image.
(c) Cross section of (a) showing zero crossings.
(d) 5×5 kernel approximation to the shape in (a). The negative of this kernel would be used in practice.





0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

Laplacian of a Gaussian (LoG)

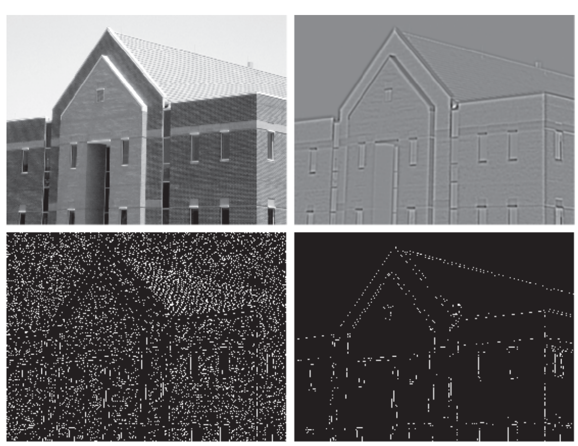
- **Marr-Hildreth edge detector**
- Convolving the image $f(x,y)$ with the LoG filter $\nabla^2 G(x,y)$ *as*
- $g(x,y) = \left[\nabla^2 G(x,y)\right] * f(x,y)$

  or $g(x,y) = \nabla^2\left[G(x,y) * f(x,y)\right]$

Laplacian of a Gaussian (LoG)

a b
c d

FIGURE 10.22 (a) Image of size 834×1114 pixels, with intensity values scaled to the range [0, 1]. (b) Result of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n = 25$. (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges). (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.



Difference of Gaussians (DoG)

- Approximate LoG filter with **difference of Gaussian** as

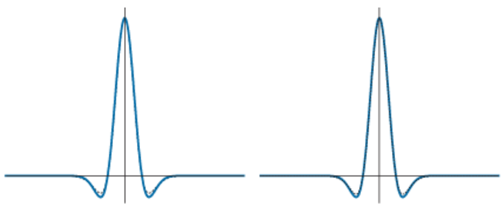
$$\mathrm{DoG}(x,y)=\frac{1}{2\pi\sigma_1^2}e^{-\frac{x^2+y^2}{2\sigma_1^2}}-\frac{1}{2\pi\sigma_2^2}e^{-\frac{x^2+y^2}{2\sigma_2^2}}$$

- LoG and DoG have the same zero-crossing if

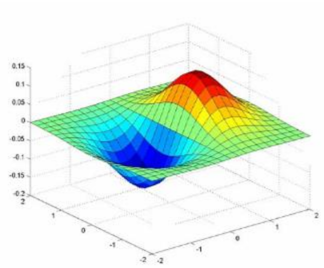
$$\sigma^2=\frac{\sigma_1^2\sigma_2^2}{\sigma_1^2-\sigma_2^2}\ln\left[\frac{\sigma_1^2}{\sigma_2^2}\right]$$

a b

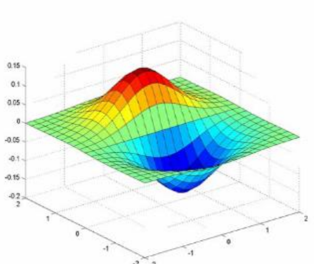
FIGURE 10.23 (a) Negatives of the LoG (solid) and DoG (dotted) profiles using a σ ratio of 1.75:1. (b) Profiles obtained using a ratio of 1.6:1.



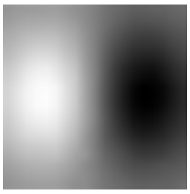
DoG Filters

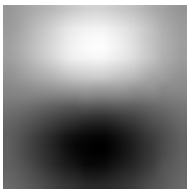


x-direction

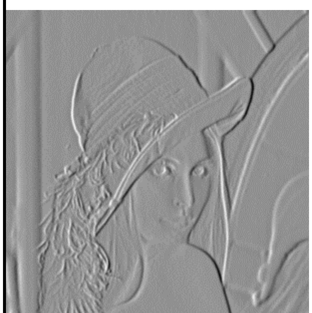


y-direction



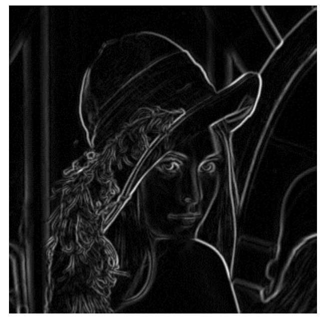


DoG Gradients



X-Derivative of Gaussian

Y-Derivative of Gaussian



Gradient Magnitude

Canny Edge Detector

- This is probably the most widely used edge detector in computer vision (**30,130 cites**)
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

A computational approach to edge detection
J Canny - IEEE Transactions on pattern analysis and machine ..., 1986 - ieeexplore.ieee.org
Abstract-This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the ...
Cited by 24439 Related articles All 24 versions Import into BibTeX Save More

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Canny Edge Detector

- **Canny Edge detector**
  - Low error rate
  - Edge point should be well localized
  - Single edge point response
- Smooth the image by circular 2-D Gaussian function as $f_s(x,y) = G(x,y) * f(x,y)$

  where $G(x,y) = e^{\frac{x^2+y^2}{2\sigma^2}}$
- Magnitude is $M(x,y) = \sqrt{g_x^2 + g_y^2}$
- Direction is $\alpha(x,y) = \tan^{-1}[g_y/g_x]$

Orientations of Pixels

- Threshold at minimum level
- Get orientation

theta = atan2(-gy, gx)

Non-Max Suppression

- Let $d_1, \ldots, d_4$ denote the four basic edge directions.
- The following **non-maxima suppression** scheme for a 3×3 region centered at (x, y) in $\alpha(x, y)$:

1) Find the direction d_k that is closest to $\alpha(x, y)$.
2) If $M(x, y)$ is less than at least one of its two neighbors along d_k let $g_N(x, y) = 0$ **(suppression)**; otherwise $g_N(x, y) = M(x, y)$, where $g_N(x, y)$ is the non-maxima-suppressed image.

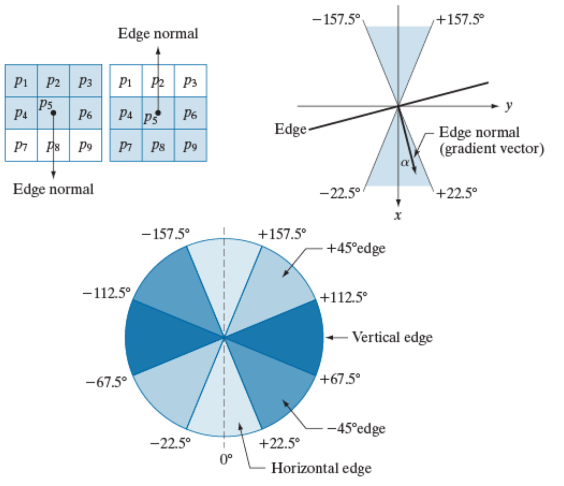
(Example in Fig. 10.24, center point p_5 and a horizontal edge through p_5. neighbors: p_2 and p_8)

3) Threshold $g_N(x, y)$ to reduce the false edge points.

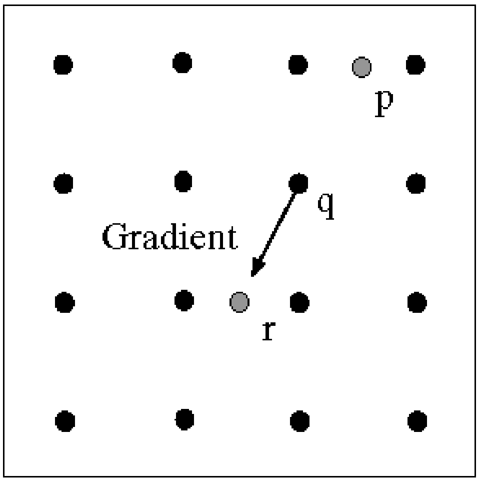
Non-Max Suppression

a b
c

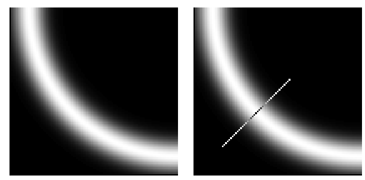
FIGURE 10.24 (a) Two possible orientations of a horizontal edge (shaded) in a 3×3 neighborhood. (b) Range of values (shaded) of α, the direction angle of the edge normal for a horizontal edge. (c) The angle ranges of the edge normals for the four types of edge directions in a 3×3 neighborhood. Each edge direction has two ranges, shown in corresponding shades.



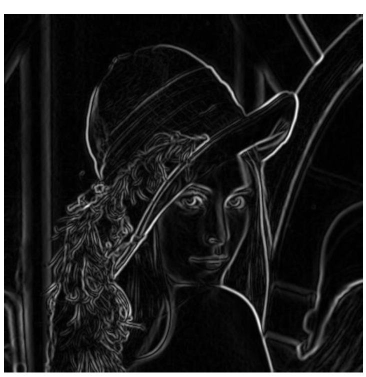
Non-Max Suppression



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Non-Max Suppression



Before

After

Hysteresis (Double) Thresholding

Thresholding:

- Threshold too low → false edges found (false positive).
- Threshold too high → valid edges eliminated (false negative).
- Canny algorithm using **hysteresis thresholding** by using two thresholds $T_L$andT_H, $\frac{T_H}{T_L} \cong 2 \sim 3$.
- Strong edges: $g_{NH}(x,y) = g_N(x,y) \geq T_H$
- Weak edges: $g_{NL}(x,y) = g_N(x,y) \geq T_L$

$$g_{NL}(x,y) = g_{NL}(x,y) - g_{NH}(x,y)$$

Hysteresis (Double) Thresholding

- **Longer edges** are formed using the following steps:

a) Locate the next un-visited edge pixel, p, in $g_{NH}(x,y)$.

b) Mark as **valid edge pixels** all the weak pixels in $g_{NL}(x,y)$ that are connected to p using 8-connectivity.

c) If all nonzero pixels in $g_{NH}(x,y)$ have been visited, continue, else return to Step (a).

d) Set to zero all pixels in $g_{NL}(x,y)$ that were not marked as valid edge pixels.

e) The final edge image output is formed by appending to $g_{NH}(x,y)$ all the nonzero pixel from $g_{NL}(x,y)$.

Hysteresis (Double) Thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Final Canny Edges

More Advanced Edge Detection

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

- MATLAB: edge(image, ‘canny’)

More Advanced Edge Detection

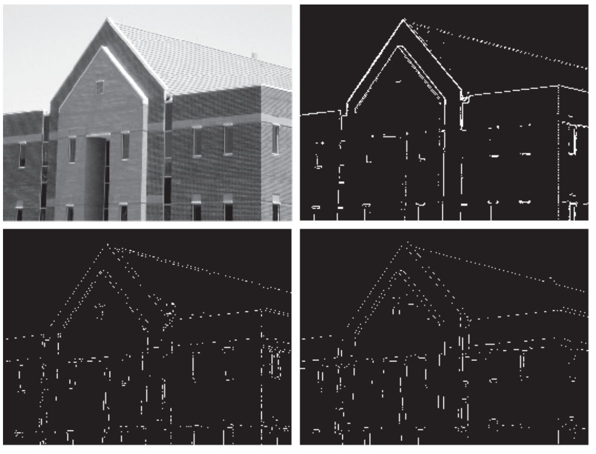
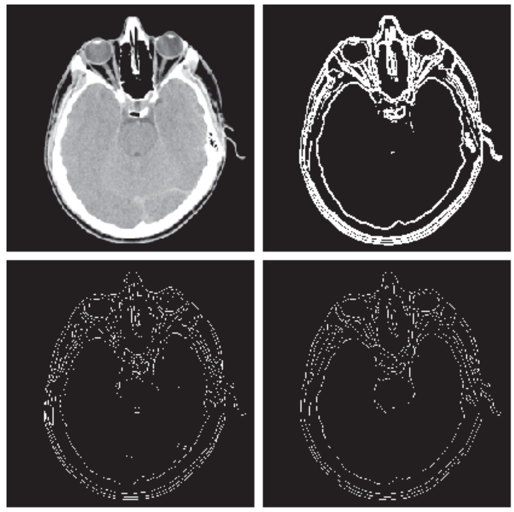


FIGURE 10.25 (a) Original image of size 834×1114 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of the smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

More Advanced Edge Detection

a b
c d

FIGURE 10.26 (a) Head CT image of size 512×512 pixels, with intensity values scaled to the range [0, 1]. (b) Thresholded gradient of the smoothed image. (c) Image obtained using the Marr-Hildreth algorithm. (d) Image obtained using the Canny algorithm. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)



Edge Linking and Boundary Detection

- **Edge detection**: true edge pixels and noise.
- **Edge linking**: assemble the edge pixels into meaningful edges.
- **Local processing**: edge points in a local region (3×3)
- **Regional processing**: edge points on a boundary of a region.
- **Global processing**: edge points on entire image.

Edge Linking and Boundary Detection

- **Local processing:** All points that are similar are linked.
- For pixels $(s,t) \in S_{xy}$, S_{xy} is the neighborhood of pixel (x,y), do the following **similarity measure**.
- **Similar in magnitude:**

$$|M(s,t) - M(x,y)| \leq E$$

- **Similar in direction**:

$$|\alpha(s,t) - \alpha(x,y)| \leq A$$

- Pixel (s,t) is linked to pixel (x,y) if the two conditions are satisfied.

Edge Linking and Boundary Detection

- **Simplified local processing:**

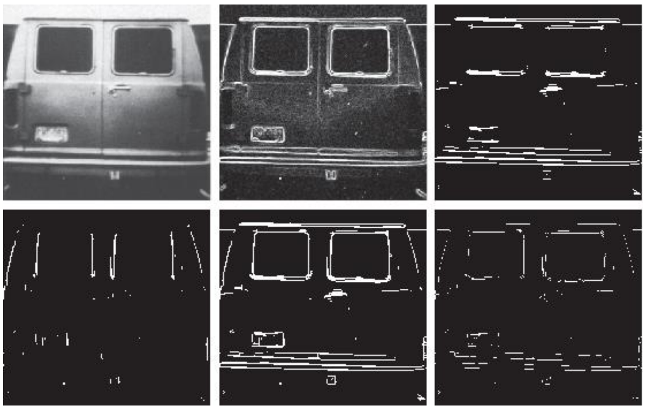
1) Compute the $M(x,y)$ and $\alpha(x,y)$ of the input image $f(x,y)$.
2) Form a binary image $g(x,y)$ as

$$g(x,y) = \begin{cases} 1 & \text{if } M(x,y) > T_M \text{ AND } \alpha(x,y) = A \pm T_A \\ 0 & \text{otherwise} \end{cases}$$

where T_M is a threshold, A is a specified direction, and $\pm T_A$ is a margin.

3) Scan the rows of $g(x,y)$ and fill all **gaps** in each row that do not exceed certain length K. The rows are processed individually.
4) To detect gaps in any other direction, θ, rotate $g(x,y)$ by this angle and apply the **gap filling** in step (3).

Edge Linking and Boundary Detection



a b c
d e f

FIGURE 10.27
(a) Image of the rear of a vehicle.
(b) Gradient magnitude image.
(c) Horizontally connected edge pixels.
(d) Vertically connected edge pixels.
(e) The logical OR of (c) and (d).
(f) Final result, using morphological thinning. (Original image courtesy of Perceptics Corporation.)

Global Processing via Hough Transform

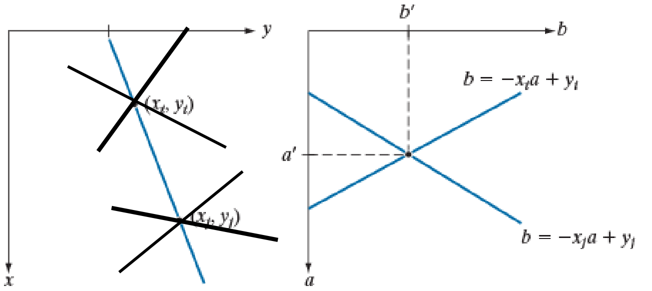
- Given a point (x_i, y_i) in **image space** — many lines pass through this point as $y_i = ax_i+b$ with different a and b.
- A point (x_i,y_i) in **image space** is mapped to many points $\{(a,b)\}$ in **parameter space** which are collinear

$$b = -ax_i+y_i.$$

- The **collinear points in parameter space** $\{(a_j,b_j)\}$ — many lines pass through this point (x_i,y_i) as $y_i = a_jx_i+b_j$ with different a_j and b_j.
- The **collinear points** in image space $\{(x_j,y_j)\}$ is mapped to many lines in parameter space as $b = -ax_j+y_j$.
- These lines intersect at the same point (a,b).

Global Processing via Hough Transform

These two lines in parameter space intersect at (a', b')



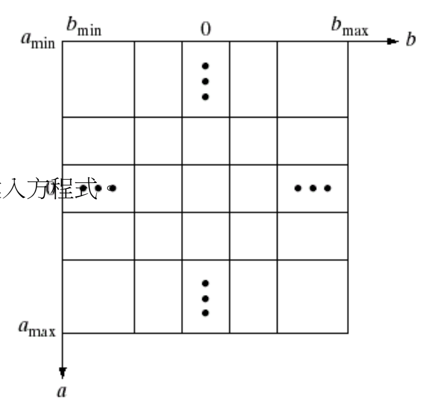
a b
FIGURE 10.28
(a) xy-plane.
(b) Parameter space.

a' is slope, b' is the intercept of the line passing through (x_i, y_i) and (x_j, y_j)

$(a', b') \in \{(a, b)\}$

Global Processing via Hough Transform

FIGURE 10.18
Subdivision of the parameter plane for use in the Hough transform.



Accumulator Cell $A(i, j)$: 在這裡鍵入方程式

$(x_i, y_i) \rightarrow \{(a, b)\}$

$a \rightarrow a_p$ and $b \rightarrow b_q$

$A(p, q) = A(p, q) + 1$

Problem: slope "a" may approach infinity for detecting a vertical line

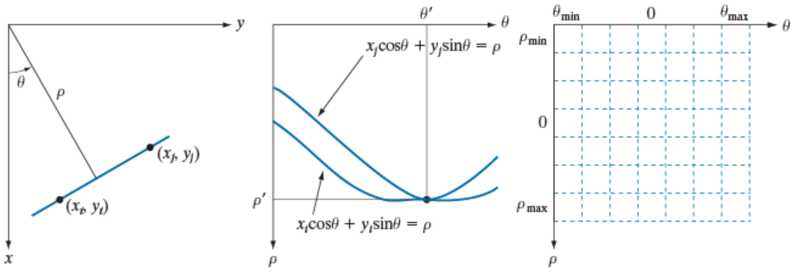
Global Processing via Hough Transform

- Given a point (x_i, y_i) → many lines pass through this point as $x_i \cos\theta + y_i \sin\theta = \rho$ with different θ and ρ
- A point (x_i, y_i) in image space is mapped a set of points $\{(\theta, \rho)\}$ in parameter space
- The point (x_j, y_j) → many lines pass through this point as $x_i \cos\theta + y_i \sin\theta = \rho$ with different θ and ρ.
- The collinear point (x_j, y_j) in image space is mapped to another set of points $\{(\theta, \rho)'\}$ in parameter space.
- These two sets in parameter space intersect at (θ_j, ρ_i)
- Collinear points $(x_i, y_i) \in \{(x, y)\}$ lies on a line:

$$x \cos\theta_j + y \sin\theta_j = \rho_i$$

Global Processing via Hough Transform

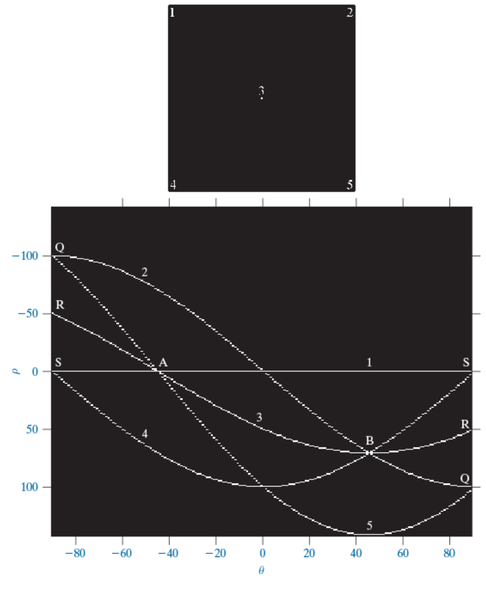
Representing a line as $x\cos\theta + y\sin\theta = \rho$



a b c

FIGURE 10.29 (a) (ρ,θ) parameterization of a line in the xy-plane. (b) Sinusoidal curves in the $\rho\theta$-plane;the point of intersection (ρ',θ') corresponds to the line passing through points (x_i, y_i) and (x_j, y_j) in the xy-plane. (c) Division of the $\rho\theta$-plane into accumulator cells.

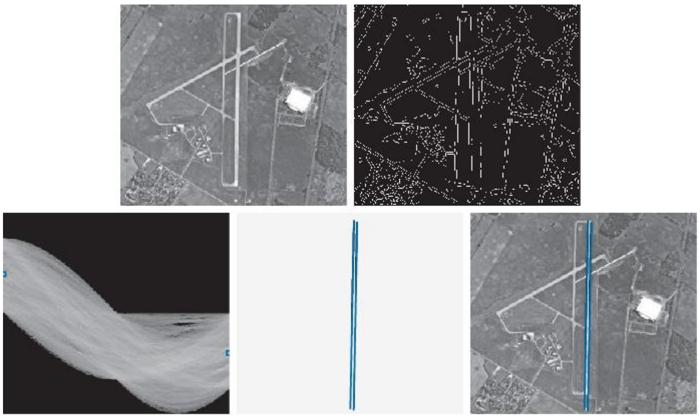
Global Processing via Hough Transform



a
b

FIGURE 10.30 (a) Image of size 101×101 pixels, containing five white points (four in the corners and one in the center). (b) Corresponding parameter space.

Global Processing via Hough Transform



a b
c d e

FIGURE 10.31 (a) A 502×564 aerial image of an airport. (b) Edge map obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes. (e) Lines superimposed on the original image.

Global Processing via Hough Transform

- The Hough transform can be applied to any function of the form $g(\mathbf{v}, \mathbf{c}) = 0$ where $\mathbf{v}$ is a vector of coordinate and $\mathbf{c}$ is a vector of coefficients.
- For example to detect a **circle**:

$$(x - c_1)^2 + (y - c_2)^2 = c_3^2$$

  where $\mathbf{v} = (x, y)$, $\mathbf{c} = (c_1, c_2, c_3)$
- Accumulator $A(i, j, k) = A(c_1, c_2, c_3)$

Global Processing via Hough Transform

- **Circle detection**

1. Let V_p denote the set of point $\{\mathbf{v} | p(\mathbf{v}) \neq 0\}$
2. For each image point $p(\mathbf{v})$, there is a set of circles passing through $\mathbf{v}$. Let C_v denote the set of circles.
3. Find the center (c_1, c_2) and the radius c_3 of each member in C_v under the constraint that $\mathbf{v} \in V_p$.
4. For each member of $\{\mathbf{c} | \mathbf{v} \in V_p\}$ an **accumulator** at (c_1, c_2, c_3) in $\mathbf{c}$ space is incremented by 1.

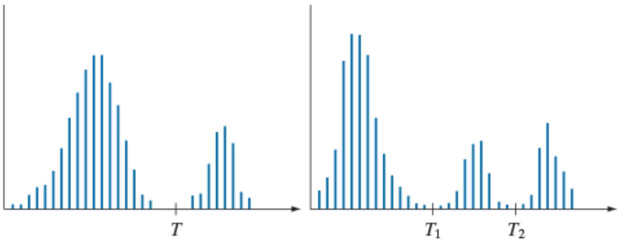
Thresholding

- To extract an object form the background is to select a threshold T that separate the object pixels from background pixels.
- Any point (x, y) with $f(x, y) > T$ is called an object point; otherwise, the point is called a background point.
- For multilevel thresholding classifies a point (x, y) as belongs to one object class if $T_1 < f(x, y) \leq T_2$, and to the other object class if $f(x, y) > T_2$.

Thresholding

a b

FIGURE 10.32 Intensity histograms that can be partitioned (a) by a single threshold, and (b) by dual thresholds.



Thresholding

- The threshold T is determined as
$$T = T[x, y, p(x, y), f(x, y)]$$
where $p(x, y)$ denotes some **local property** of this point (x, y), e.g., the average level of a neighborhood centered on (x, y).
- A thresholded image is defined as:
$$g(x, y) = 1 \text{ if } f(x, y) \geq T$$
$$g(x, y) = 0 \text{ if } f(x, y) < T$$
- If T does not depend on $p(x, y)$ then the threshold is called global threshold, otherwise it is called local or adaptive threshold.

Thresholding

Histogram distortion due to noise

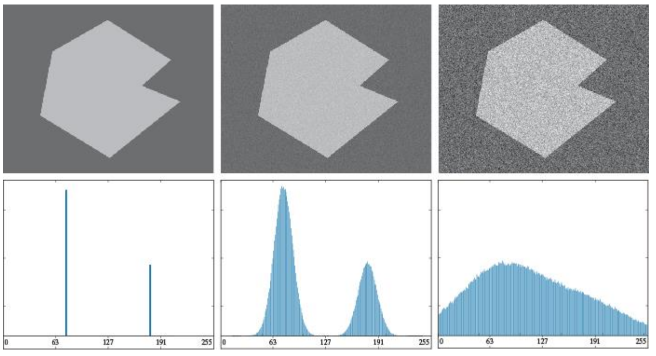
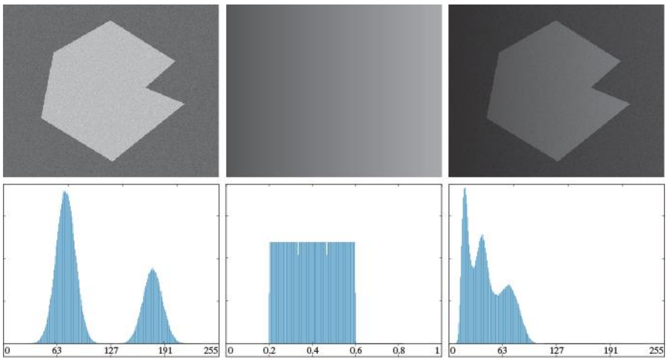


FIGURE 10.33 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d) through (f) Corresponding histograms.

Thresholding

Histogram distortion due to non-uniform illumination



a b c
d e f

FIGURE 10.34 (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d) through (f) Corresponding histograms.

Thresholding

- $f(x,y) = i(x,y)r(x,y)$
- $z(x,y) = \ln\{f(x,y)\} = \ln\{i(x,y)\} + \ln\{r(x,y)\}$
  $= i'(x,y) + r'(x,y)$
- If $i'(x,y)$ and $r'(x,y)$ are **independent random variables**, the histogram of $z(x,y)$ is given by the **convolution** of the histogram of $i'(x,y)$ and $r'(x,y)$.
- If $i(x,y)$ = constant and $i'(x,y)$ = constant (its histogram is an impulse), then the histogram of $z(x,y) \approx r'(x,y)$ is unchanged.
- If $i'(x,y)$ has a broader histogram (nonuniform illumination), the convolution process may smear the histogram of $r'(x,y)$ and the shape of the histogram of $z(x,y)$ will be quite different from $r'(x,y)$.
- The degree of distortion depends on the broadness of the histogram of $i'(x,y)$.

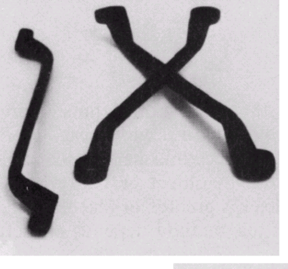
Thresholding

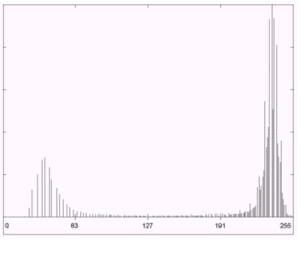
- If the illumination source $i(x, y)$ is available, compensating the non-uniformity by projecting the illumination on a **constant white reflective surface** (i.e., $r(x, y) = k$) . This yields a new image as $g(x, y) = ki(x, y)$ where k is a constant
- For any image $f(x, y) = i(x, y)r(x, y)$, we have a normalized image as $h(x, y) = f(x, y)/g(x, y) = r(x, y)/k$
- If $r(x, y)$ can be segmented by threshold T then $h(x, y)$ can be segmented by threshold T/k.

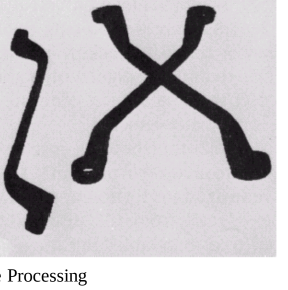
Basic Global Thresholding

1. Partition the image using a single threshold T

2. How to find the best T?





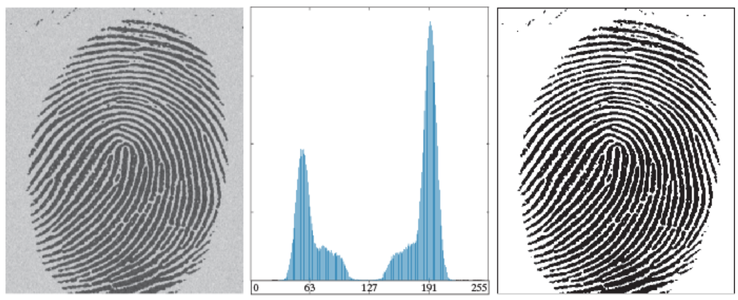


a
b c

(a) Original image. (b) Image histogram. (c) Result of global thresholding with T midway between the maximum and minimum gray levels.

Basic Global Thresholding

T = 125



a b c

FIGURE 10.35 (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (thin image border added for clarity). (Original image courtesy of the National Institute of Standards and Technology.).

Otsu Thresholding

- **Minimizing within group variance**
- Let $P(i)$ denote the **probability distribution** of the gray level $i = 1, \ldots, I$ of a picture.
- Let t be the **threshold** that separates the image pixels into two groups, $\{1, \ldots, t\}$ and $\{t+1, \ldots, I\}$
- $q_1(t)$ be the probability for group with $i \leq t$, i.e.,

$$q_1(t) = \sum_{i=1}^{t} P(i)$$

- $q_2(t)$ be the probability for group with $i > t$, i.e.,

$$q_2(t) = \sum_{i=t+1}^{I} P(i)$$

Otsu Thresholding

- Let $\mu_1(t)$ and $\mu_2(t)$ be the **mean** for group 1 and group 2 as $\mu_1(t) = \sum_{i=1}^{t} iP(i)/q_1(t)$ and $\mu_2(t) = \sum_{i=t+1}^{I} iP(i)/q_2(t)$
- Let $\sigma_1(t)$ and $\sigma_2(t)$ be the **variance** for group 1 and group 2 as $\sigma_1^2(t) = \sum_{i=1}^{t}[i-\mu_1(t)]^2P(i)/q_1(t)$

  and $\sigma_2^2(t) = \sum_{i=t+1}^{I}[i-\mu_2(t)]^2P(i)/q_2(t)$
- Let σ_w be the weighted sum of group variance (or **within group variance**), i.e.,

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

Otsu Thresholding

- The relationship between σ and σ_w

$$\sigma^2(t) = \sum_{i=1}^{I}[i-\mu]^2 P(i)$$

$$= \sum_{i=1}^{t}[i-\mu_1(t)+\mu_1(t)-\mu]^2 P(i) + \sum_{i=t+1}^{I}[i-\mu_2(t)+\mu_2(t)-\mu]^2 P(i)$$

$$= \sigma_w^2(t) + q_1(t)[1-q_1(t)][\mu_1(t)-\mu_2(t)]^2 = \sigma_w^2(t) + \sigma_B^2(t)$$

- σ_B is the **between group variance**
- Minimize σ_w = maximize σ_B
- There is a relationship between the value of computed t and that computed for next t: $t+1$.

  $q_1(t+1) = q_1(t) + P(t+1)$ and $q_2(t+1) = q_2(t) - P(t+1)$

Otsu Thresholding

- We can obtain the recursive relation

$$\mu_1(t+1)=\frac{\sum_{i=1}^{t} iP(i)+(t+1)P(t+1)}{q_1(t+1)}=\frac{q_1(t)\mu_1(t)+(t+1)P(t+1)}{q_1(t+1)}$$

$$\mu_2(t+1)=\frac{q_2(t)\mu_2(t)-(t+1)P(t+1)}{q_2(t+1)}=\frac{\mu-q_1(t)\mu_1(t)-(t+1)P(t+1)}{1-q_1(t+1)}$$

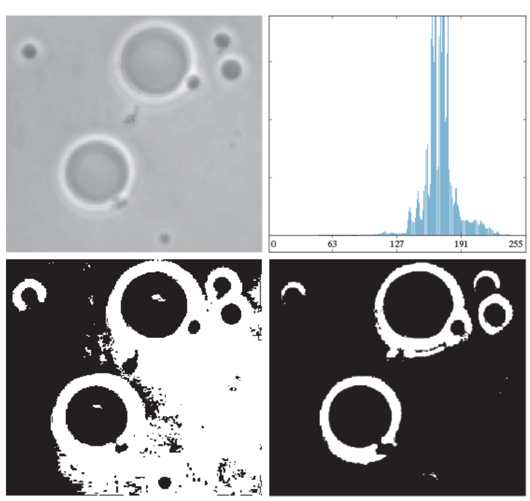
$$=\frac{\mu-q_1(t+1)\mu_1(t+1)}{1-q_1(t+1)}$$

Select t → $\mu_1(t)$ → $\mu_1(t+1)$ → $\mu_2(t+1)$ → maximize σ_B

Otsu Thresholding

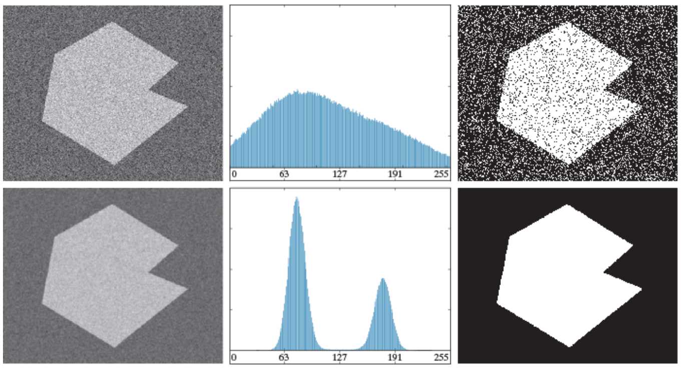
a b
c d

FIGURE 10.36
(a) Original image.
(b) Histogram (high peaks were clipped to highlight details in the lower values).
(c) Segmentation result using the basic global algorithm from Section 10.3.
(d) Result using Otsu's method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)



Using Image smoothing to Improve Global Thresholding

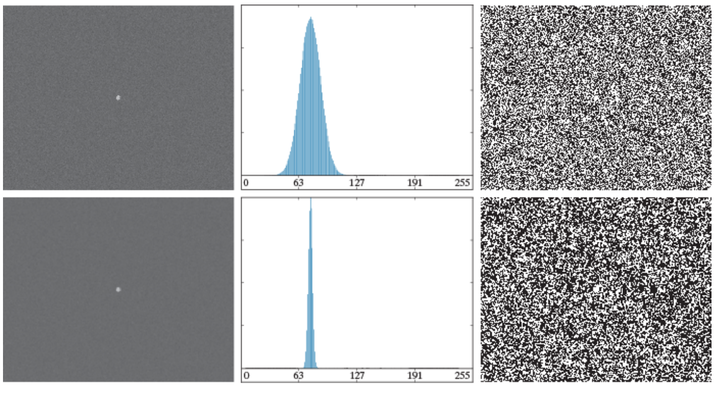
- Reduce the noise before thresholding



a b c
d e f

FIGURE 10.37 (a) Noisy image from Fig. 10.33(c) and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging kernel and (e) its histogram. (f) Result of thresholding using Otsu's method.

Using Image smoothing to Improve Global Thresholding



a b c
d e f

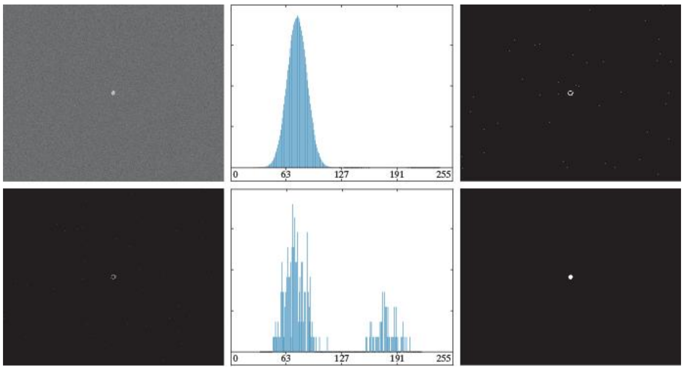
FIGURE 10.38 (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging kernel and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases to extract the object of interest. (See Fig. 10.39 for a better solution.)

Using Edges to Improve Global Thresholding

- Assume that the edges between objects and background are known.

1) Compute the edge image of the input image $f(x, y)$.
2) Specify a Threshold T
3) Threshold the edge image using threshold T and produce a binary image $g_T(x, y)$ as a **mask image**.
4) Compute a histogram using only the pixels in $f(x, y)$ that correspond to mask image $g_T(x, y)$.
5) Use the histogram from step (4) to segment $f(x, y)$ globally using Otsu's method.

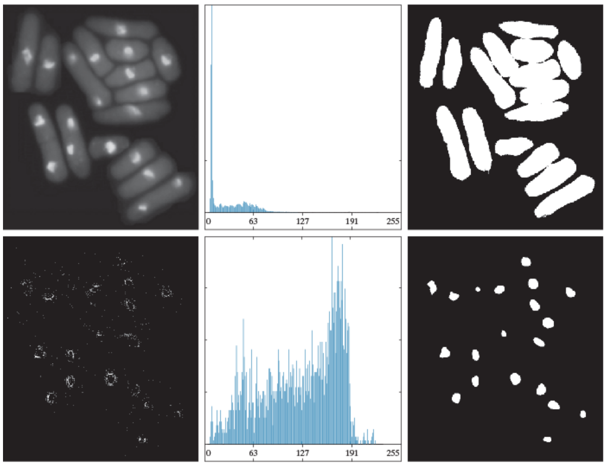
Using Edges to Improve Global Thresholding



a b c
d e f

FIGURE 10.39 (a) Noisy image from Fig. 10.38(a) and (b) its histogram. (c) Mask image formed as the gradient magnitude image thresholded at the 99.7 percentile. (d) Image formed as the product of (a) and (c). (e) Histogram of the nonzero pixels in the image in (d). (f) Result of segmenting image (a) with the Otsu threshold based on the histogram in (e). The threshold was 134, which is approximately midway between the peaks in this histogram.

Using Edges to Improve Global Thresholding



a b c
d e f

FIGURE 10.40 (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Mask image formed by thresholding the absolute Laplacian image. (e) Histogram of the nonzero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram in (e). (Original image courtesy of Professor Susan L. Forsburg, University of Southern California.)

Using Edges to Improve Global Thresholding

FIGURE 10.41 Image in Fig. 10.40(a) segmented using the same procedure as explained in Figs. 10.40(d) through (f), but using a lower value to threshold the absolute Laplacian image.

Multiple Thresholding

- In the cases of K classes, $C_1, C_2, \dots, C_K$, the between class variance are

$$\sigma_B = \sum_{k=1}^{K} P_k (m_k - m_G)^2$$

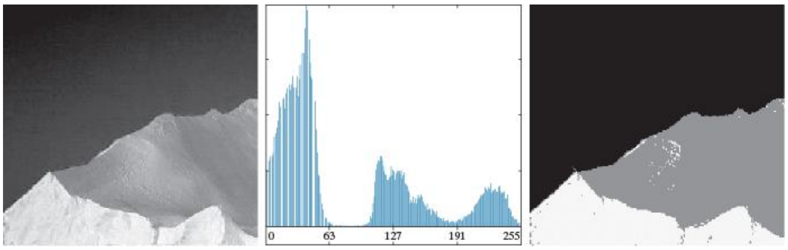
where m_G is the **global mean**, and m_k is the **local mean** of class k, $P_k = \sum_{i \in C_k} p_i$. ($i \in Ck$ indicates, $T_{k-1} < i \le T_k$)

- The K class are separated by $K-1$ thresholds, T_1^* , T_2^*,..., T_{K-1}^*, which are determined by maximize $\sigma_B(T_1, T_2, \dots, T_{k-1})$
- For three classes:

$$\sigma_B = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 + P_3(m_3 - m_G)^2$$

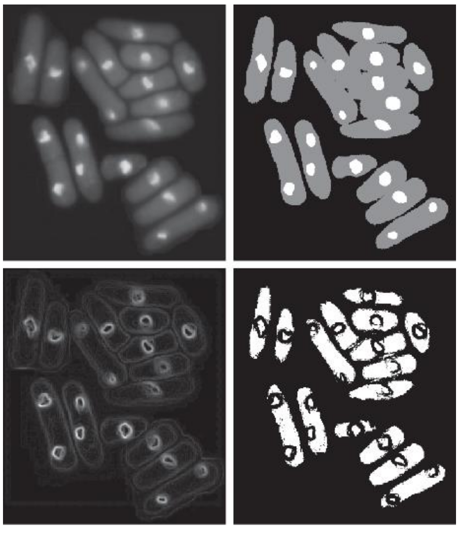
- Minimize $\sigma_B(T_1, T_2)$ to determine T_1^* and T_2^*

Multiple Thresholding



a b c
FIGURE 10.42 (a) Image of an iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

Variable Thresholding Based on Local Image Properties



a b
c d

FIGURE 10.43 (a) Image from Fig. 10.40. (b) Image segmented using the dual thresholding approach given by Eq. (10-76). (c) Image of local standard deviations. (d) Result obtained using local thresholding.

Variable Thresholding Using Moving Average

- Compute a moving average along scan lines.
- The scanning is carried out line-by-line in a zigzag pattern to reduce illumination bias.
- Let z_{k+1} denote the intensity of point encountered in the scanning sequence at step $k+1$
- The moving average at this new point is

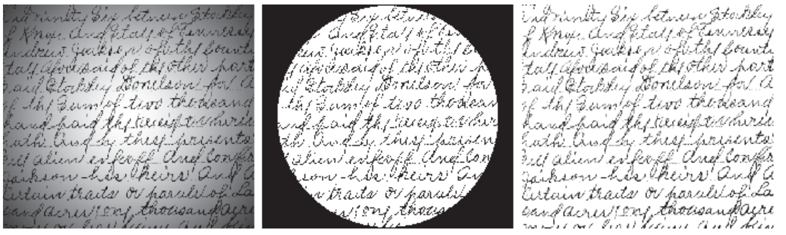
$$m(k+1) = \frac{1}{n}\sum_{i=k+2-n}^{k+1} z_i = m(k) + \frac{1}{n}(z_{k+1} - z_{k-n})$$

where n denotes the number of points used in computing, and $m(1) = z_1/n$.

- Segmentation is using $g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{if } f(x,y) \le T_{xy} \end{cases}$

with $T_{xy} = bm_{xy}$

Variable Thresholding Using Moving Average

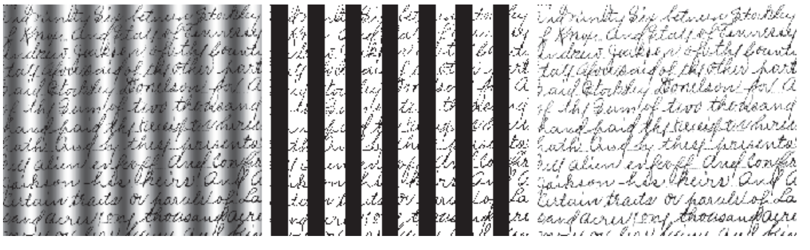


a b c

FIGURE 10.44 (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.

n = 5×(average stroke width) = 20

Variable Thresholding Using Moving Average



a b c

FIGURE 10.45 (a) Text image corrupted by sinusoidal shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages..

Multivariable Thresholding

- For color images with R, G, and B components
- Each pixel at (x, y) is characterized by three values and represented as 3-D vector, $\mathbf{z}(x, y) = (z_1, z_2, z_3)$
- Distance measure $D(\mathbf{z}, \mathbf{a})$ between an arbitrary color point $\mathbf{z}(x, y)$ and the average color $\mathbf{a}$.
- Segment the input image based on the distance measure

$$g(x, y) = \begin{cases} 1 & if\ D(\mathbf{z}(x, y), \mathbf{a}) > T \\ 0 & \text{otherwise} \end{cases}$$

- The **Euclidean distance** is defined as

$$D(\mathbf{z}, \mathbf{a}) = \|\mathbf{z} - \mathbf{a}\| = \left[(\mathbf{z} - \mathbf{a})^T (\mathbf{z} - \mathbf{a})\right]^{1/2}$$

- The **Mahalanobis distance** is defined as

$$D(\mathbf{z}, \mathbf{a}) = \|\mathbf{z} - \mathbf{a}\| = \left[(\mathbf{z} - \mathbf{a})^T C^{-1} (\mathbf{z} - \mathbf{a})\right]^{1/2}$$

Region-Based Segmentation

- Segmentation is accomplished by finding the region directly.
- Segmentation is to partition the image into **sub-regions**: $R_1, R_2, \ldots, R_n$, where

  (a) R_i is a connected region

  (b) $R_1 \cup R_2 \cdots \cup R_n = R$

  (c) $R_i \cap R_j = \emptyset$ for $i \neq j$

  (d) $Q(R_i) = \text{TRUE}$, all pixel in R_i have **the same gray level or texture.**

  (e) $Q(R_i \cup R_j) = \text{FALSE}$ for $i \neq j$

Region Growing

- Region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria.
- It starts with a set of "**seed**" points and from these grow regions by appending to each seed those neighboring pixels that have properties similar to the seed (such as specified gray-level or color)

Region Growing

- Let $f(x, y)$: input image, $S(x, y)$: a seed array, Q a **predicate** applied at location (x, y), f and S are of the same size.

1) Find all connected components in $S(x, y)$ and erode each component to a single pixel.
2) Form an image f_Q, such that $f_Q(x, y) = 1$ if predicate Q is satisfied at (x, y) else $f_Q(x, y) = 0$.
3) Let $g(x, y)$ be an image formed by appending each seed point in S all the 1-valued points in $f_Q(x, y)$ that are 8-connected to that seed point.
4) Label each connected component in g with different region label.

Region Growing

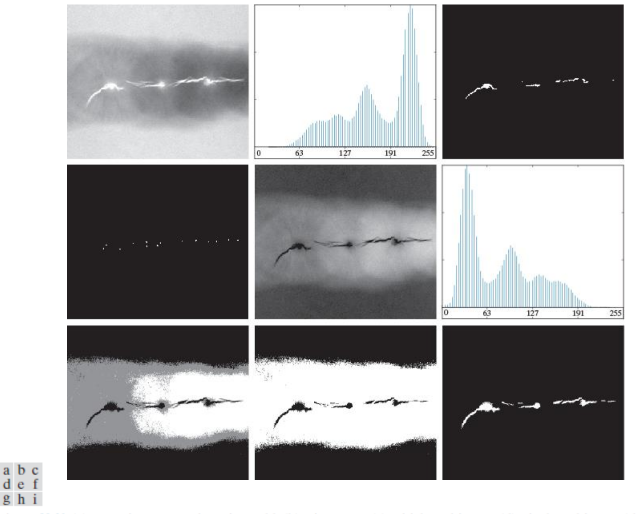


Figure 10.46 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between the seed value (255) and (a). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing. (Original image courtesy of X-TEK Systems, Ltd.)

Region Splitting and Merging

- Subdivide an image initially into a set of arbitrary disjointed regions and then merge and/or split the regions in an attempt to satisfy the conditions of regions
- Two adjacent regions R_i and R_j are **merged** only if $Q(R_i \cup R_j) = \text{TRUE}$

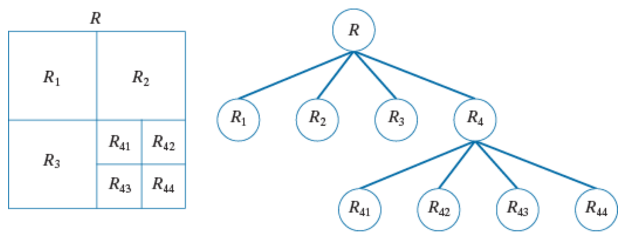
  The split and merge algorithm is mentioned as follows:

1. **Split** into four disjoint quadrants any region R_i for which $P(R_i) = \text{FALSE}$
2. **Merge** any adjacent regions R_i and R_j, for which $Q(R_i \cup R_j) = \text{TRUE}$
3. Stop when no further merging or splitting is possible.

Region Splitting and Merging

a b

FIGURE 10.47
(a) Partitioned image.
(b) Corresponding quadtree.
R represents the entire image region.

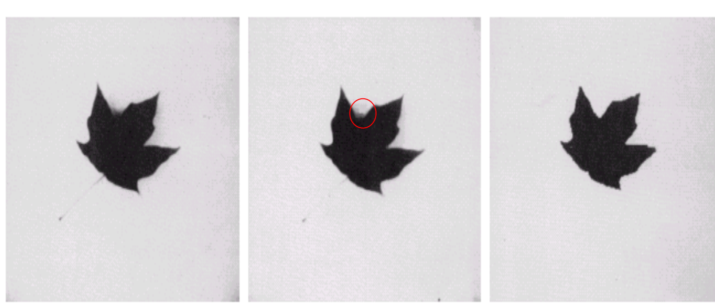


Region Splitting and Merging

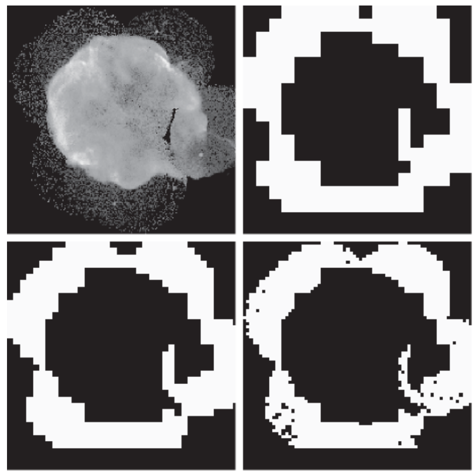
1. Define $Q(R_i) = \text{TRUE}$ if at least 80% of the pixel in R_i have the property $|z_i - mi| \leq 2\sigma_i$, m_i is the mean, σ_i is the standard deviation.
2. If $Q(R_i) = \text{TRUE}$, the value of all pixels in R_i are set to m_i the shaded area is erroneously removed.

a b c

(a) Original image. (b) Result of split and merge procedure.
(c) Result of thresholding (a).



Region Splitting and Merging



a b
c d

FIGURE 10.48 (a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope. (b) through (d) Results of limiting the smallest allowed quadregion to be of sizes of 32×32, 16×16, and 8×8 pixels, respectively. (Original image courtesy of NASA.)

Segmentation by Morphological Watersheds

- Segmentation based on (1) detection of discontinuity (2) thresholding, and (3) region processing.
- **Segmentation by Morphological Watersheds** embodies the concepts of the three approaches.
- Produce more stable segmentation results, *i.e.*, continuous segmentation boundary.
- Incorporate knowledge-based constraints in the segmentation process.

Segmentation by Morphological Watersheds

- Visual image in 3-D (coordinate and gray-level) and consider three types of points:

  1) points belong to regional minimum.

  2) points at which a drop of water, if placed at the location of any of these points, would fall with certainty to **a single minimum**. A set of such points is called **catchment basin** (盆地) or *watershed* of that minimum.

  3) points at which water would be equally likely to fall **more than one** such minimum. A set of such points is called **divide lines** or **watershed lines** (分水嶺).

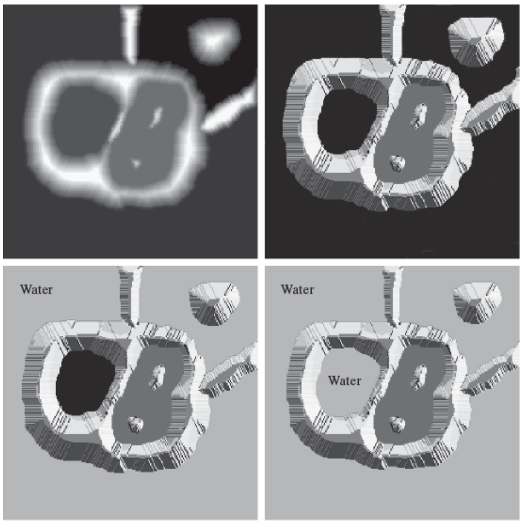
Segmentation by Morphological Watersheds

- **Segmentation** → to find the watershed lines.
- The entire topography is flooded from below by letting the water rise at a uniform rate.
- The rising water in distinct catchment basins is about to merge, a **dam** is built to prevent this merging.
- The flooding will reach a stage when only the tops of the dam are visible above the water line.
- The dam boundaries correspond to the divide lines of the watersheds.

Segmentation by Morphological Watersheds

a c
b d

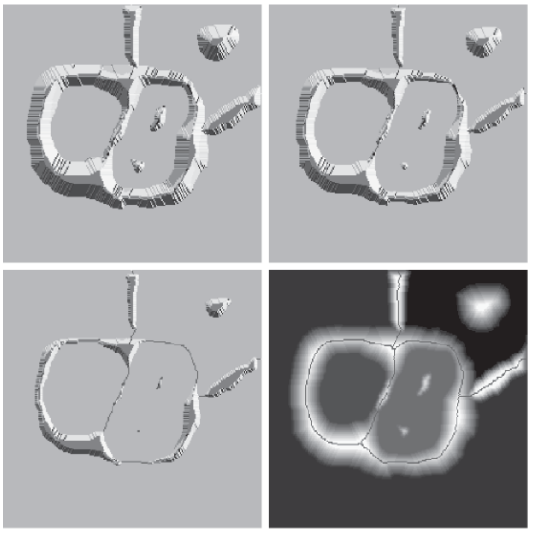
FIGURE 10.57
(a) Original image.
(b) Topographic view. Only the background is *black*. The basin on the left is slightly lighter than black.
(c) and (d) Two stages of flooding. All constant dark values of gray are intensities in the original image. Only constant *light gray* represents "water."
(Courtesy of Dr. S. Beucher, CMM/ Ecole des Mines de Paris.)
(*Continued on next page.*)



Segmentation by Morphological Watersheds

e f
g h

FIGURE 10.57
(*Continued*)
(e) Result of further flooding.
(f) Beginning of merging of water from two catchment basins (a short dam was built between them).
(g) Longer dams.
(h) Final watershed (segmentation) lines superimposed on the original image.
(Courtesy of Dr. S. Beucher, CMM/ Ecole des Mines de Paris.)



Dam Construction

- Use **morphological dilation** to construct **dam**.
- Let M_1 and M_2 denote the sets of coordinates of points in two regional minima.
- Let the set of coordinates of points in the *catchment basin* associated with the two minima at stage $n-1$ of flooding be denoted by $C_{n-1}(M_1)$ and $C_{n-1}(M_2)$.
- Let the union of the two sets be $C[n-1]$.
- The two components merge when the water between the two catchment basins has merged at the flooding step n
- Let this **connected component** (Figure 10.58(b)) be denoted as q.
- The two components from step $n-1$ can be extracted from q by the following AND operation: $q \cap C[n-1]$.

Dam Construction

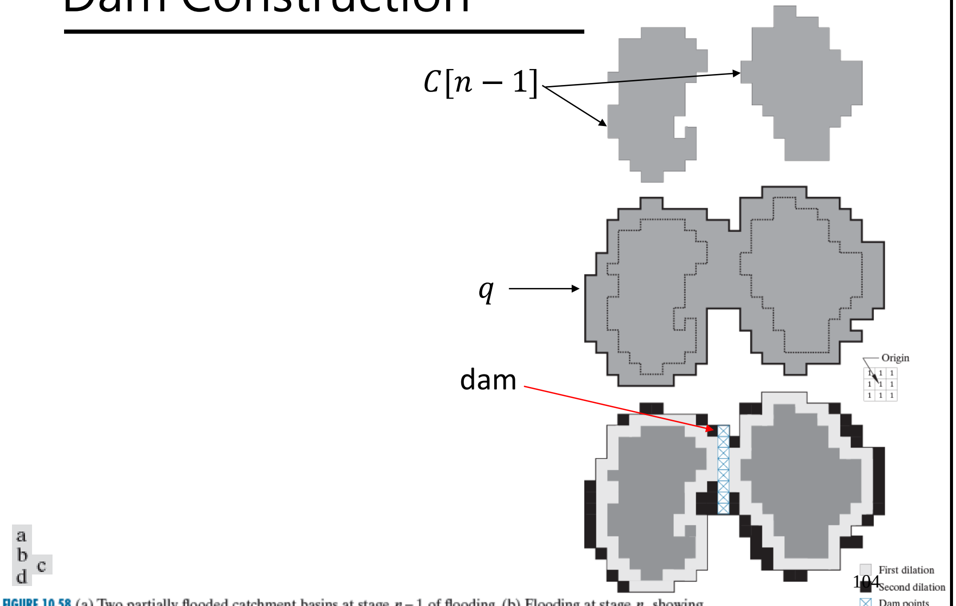


FIGURE 10.58 (a) Two partially flooded catchment basins at stage $n-1$ of flooding. (b) Flooding at stage n, showing that water has spilled between basins. (c) Structuring element used for dilation. (d) Result of dilation and dam construction.

Dam Construction

- Fig. 10.58(a) is dilated by the structure element in Fig. 10.58(c), subject to two conditions:

1) The dilation has to be constrained to q.
2) The dilation cannot be performed on points that would cause sets being dilated to merge.

- During the 1st dilation, cond. (1) is satisfied only.
- During the 2nd dilation, cond. (2) is considered only, it results in broken perimeter.
- The only points in q that satisfy the **two conditions** under consideration describe the one-pixel-thick connected path shown by **cross-hatched points.**
- The path constitutes the desired separation **dam** at stage n of flooding. (n : 水位)

Watershed Segmentation Algorithm

- Let $M_1, M_2, \ldots, M_R$ be sets denoting the coordinates of the points in the regional minima of an image $g(x, y)$.
- Let $C(M_i)$ be a set denoting the coordinates of the **points** in the *catchment basin* associated with **regional minimum** M_i .
- Let $T[n]$ represent the set of coordinates (s, t) for which $g(s, t) < n$. i.e., $T[n] = \{(s, t) | g(s, t) < n\}$
- The topology will be flooded in *integer* flood increments from $n = min + 1$ to $n = max + 1$ where min and max are the minimum and maximum value of $g(x, y)$.

Watershed Segmentation Algorithm

- Let $C_n(M_i)$ denote the set of **coordinates of points** in the catchment basin associated with minimum M_i that are flooded at stage n.
- $C_n(M_i)$ can be viewed as a **binary image** given by $C_n(M_i) = C(M_i) \cap T[n]$
- $C_n(M_i) = 1$ at location (x, y) if $(x, y) \in C(M_i)$ AND $(x, y) \in T[n]$; otherwise $C_n(M_i) = 0$
- Let $C[n]$ denote the union of the flooded catchment basins portion at stage n as

$$C[n] = \bigcup_{i=1}^{R} C_n(M_i)$$

Watershed Segmentation Algorithm

- Then $C[max + 1]$ is the union of all catchment basins as:

$$C[max + 1] = \bigcup_{i=1}^{R} C_n(M_i)$$

- $C[n-1]$ is subset of $C[n]$, $C[n]$ is a subset of $T[n] \rightarrow$ $C[n-1]$ is subset of $T[n]$.
- The algorithm for finding the watershed lines is initialized with $C[min + 1] = T[min + 1]$.
- The algorithm proceeds recursively assuming that at step n, $C[n-1]$ has been constructed.

Watershed Segmentation Algorithm

- Obtain $C[n]$ from $C[n-1]$ as follows:
  - Let Q denote the set of connected components in $T[n]$.
  - For each connected component $q \in Q[n]$: $q \cap C[n-1]$ may be

(a) **empty**: when a new minimum is encountered, where connected component q is incorporated into $C[n-1]$ to form $C[n]$

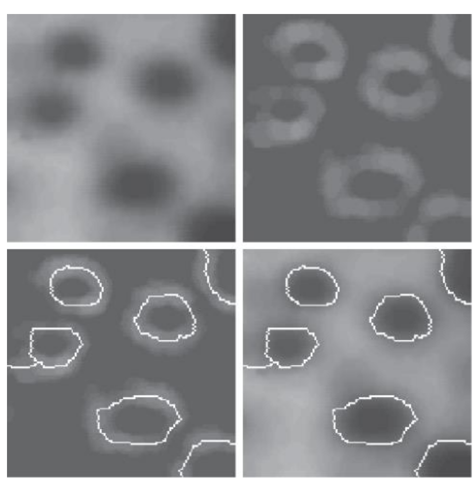
(b) **one connected component**: when q lies within the *catchment basin* of some regional minimum in which case q is incorporated into $C[n-1]$ to form $C[n]$

(c) **more than one component**: when all or part of a ridge separating two or more *catchment basins* is encountered. Further flooding would cause water level in these catchment basin to merge, therefore, a **dam** must built within q to prevent overflow between the catchment basins

Watershed Segmentation Algorithm

a b
c d

FIGURE 10.59
(a) Image of blobs.
(b) Image gradient.
(c) Watershed lines, superimposed on the gradient image.
(d) Watershed lines superimposed on the original image.
(Courtesy of Dr. S. Beucher, CMM/ Ecole des Mines de Paris.)



The Use of Markers

- Over-segmentation due to noise or other irregularities of the gradient - use the markers
- A **marker** is a connected component belonging to an image.
- Internal markers: object of interests.
- External markers: background.

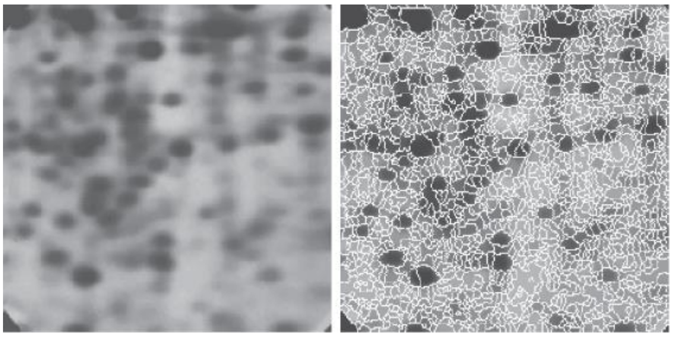
The Use of Markers

- **Marker selection:**
  - a) Preprocessing: Use smooth filtering to remove small spatial detail.
  - b) A set of criteria that markers must satisfy
    1) A region that is surrounded by points of higher “altitude”
    2) The points in the regions form a connected component.
    3) All the points in the connected region have the same gray-level value.
    4) After image is smoothed, the **internal markers** are shown as light gray, blob like region
    5) Watershed algorithm is applied and the resulting watershed lines are defined as the **external markers** (Figure 10.61(a)).

The Use of Markers

a b

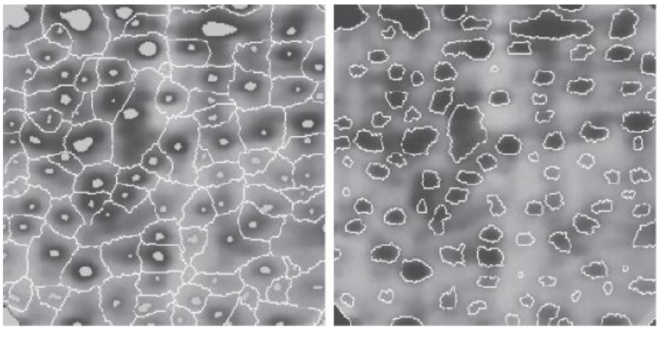
FIGURE 10.60
(a) Electrophoresis image.
(b) Result of applying the watershed segmentation algorithm to the gradient image. Over-segmentation is evident.
(Courtesy of Dr. S. Beucher, CMM/ Ecole des Mines de Paris.)



The Use of Markers

a b

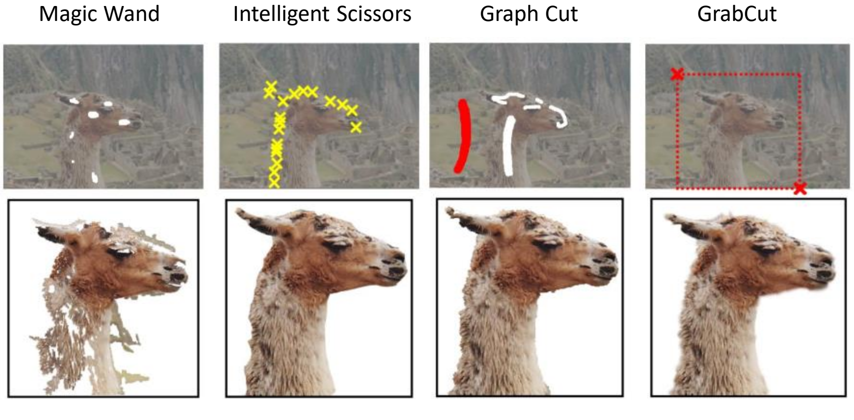
FIGURE 10.61
(a) Image showing internal markers (light gray regions) and external markers (watershed lines).
(b) Result of segmentation. Note the improvement over Fig. 10.60(b).
(Courtesy of Dr. S. Beucher, CMM/ Ecole des Mines de Paris.)



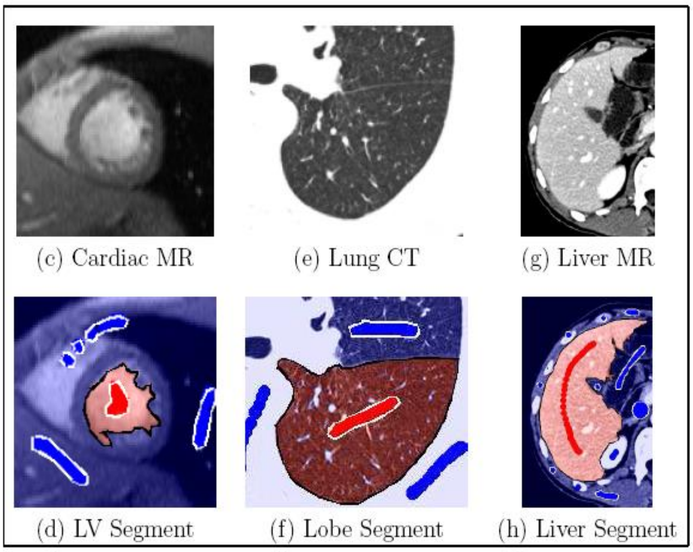
The Use of Markers

- In Fig. 10.58, the image is partitioned into regions, each containing a single internal marker and part of the background.
- Simplify the problem as partition each region into a **single object** and **its background**.
- Marker selection can be based on gray-level value and connectivity, and more complex description involving size, shape, location, texture content, and so on.
- Apply the watershed segmentation on each region with **internal marker**.

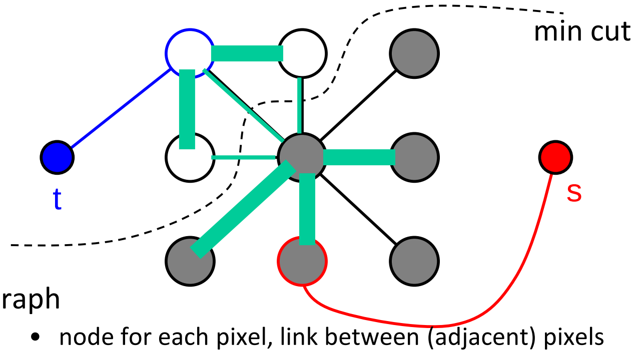
Interactive Segmentation



Graph Cut



Graph Cut-Based Segmentation



Graph
- node for each pixel, link between (adjacent) pixels
- specify a few pixels as foreground and background
  - create an infinite cost link from each bg pixel to the "t" node
  - create an infinite cost link from each fg pixel to the "s" node
- compute min cut that separates s from t
  - The min-cut max-flow theorem [Ford and Fulkerson 1956]
- how to define link cost between neighboring pixels?

Element of Graph Theory

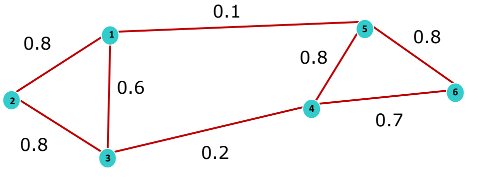
- A graph $G = (V, E)$ consists of a vertex set V and an edge set E
- If G is a directed graph, each edge is an ordered pair of vertices
- A bipartite graph is one in which the vertices can be divided into two groups, so that all edges join vertices in different groups

Similarity Graph

- Distance decrease similarity increase
- Represent dataset as a weighted graph $G(V, E)$

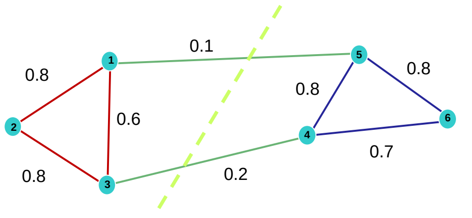
$V = \{x_i\}$ Set of n vertices representing data points

$E = \{W_{ij}\}$ Set of weighted edges indicating pair-wise similarity between points



Clustering Objectives

- Traditional definition of a "good" clustering:
  1. Points assigned to same cluster should be highly similar.
  2. Points assigned to different clusters should be highly dissimilar.
- Apply these objectives to our graph representation

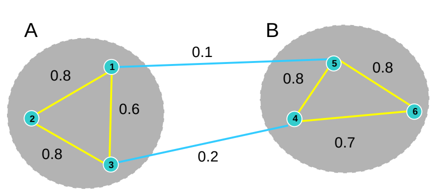


Minimize weight of between-group connections

Graph Cuts

- Express partitioning objectives as a function of the "edge cut" of the partition
- **Cut**: Set of edges with only one vertex in a group. We wants to find the minimal cut between groups. The groups that has the minimal cut would be the partition

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$



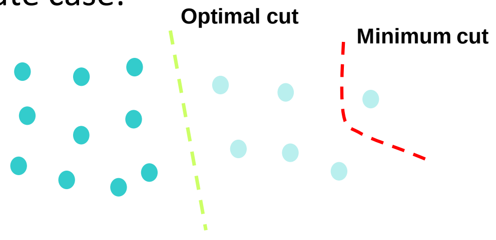
$$cut(A,B) = 0.3$$

Graph Cut Criteria

- Criterion: **Minimum-cut**
  - Minimise weight of connections between groups

$$\min cut(A,B)$$

- Degenerate case:



- Problem:
  - Only considers external cluster connections
  - Does not consider internal cluster density

Graph Cut Criteria

- Criterion: **Normalized-cut** (Shi & Malik,'97)
  - Consider the connectivity between groups relative to the density of each group.

$$\min_{A,B} N_{cut}(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

  - Normalize the association between groups by *volume*.
    - $vol(A)$: The total weight of the edges originating from group A.
    - Why use this criterion?
      - Minimizing the normalized cut is equivalent to maximizing normalized association
      - Produces more balanced partitions

Matrix Representations

- Possible approach
  - Represent a similarity graph as a matrix
  - Apply knowledge from Linear Algebra…
- The **eigenvalues** and **eigenvectors** of a matrix provide global information about its structure.

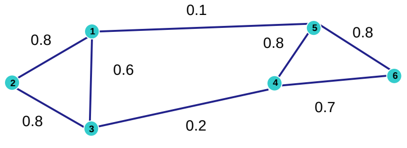
$$\begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

- **Spectral Graph Theory**
  - Analyze the “spectrum” of matrix representing a graph.
  - **Spectrum**: The eigenvectors of a graph, ordered by the magnitude (strength) of their corresponding eigenvalues.

$$\Lambda = \{\lambda_1, \lambda_2, \cdots, \lambda_n\}$$

Similarity Graph

- **Adjacency matrix (A)**
  - $n \times n$ matrix
  - $A = [w_{ij}]$: edge weight between vertex x_i and x_j



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

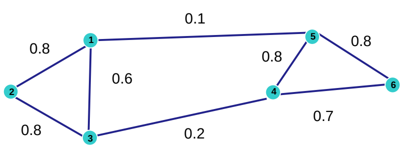
- Important properties:
  - Symmetric matrix

  $\Rightarrow$ Eigenvalues are real

  $\Rightarrow$ Eigenvector could span orthogonal base

Degree Matrix

- **Degree matrix (D)**
  - $n \times n$ diagonal matrix
  - $D(i,i) = \sum_j w_{ij}$: total weight of edges incident to vertex x_i



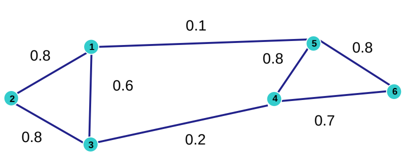
	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	0	0	0	0	0
x_2	0	1.6	0	0	0	0
x_3	0	0	1.6	0	0	0
x_4	0	0	0	1.7	0	0
x_5	0	0	0	0	1.7	0
x_6	0	0	0	0	0	1.5

- Important application:
  - Normalized adjacency matrix

Graph Laplacian Matrix

$$L = D - A$$

- **Laplacian matrix (L)**
  - $n \times n$ symmetric matrix



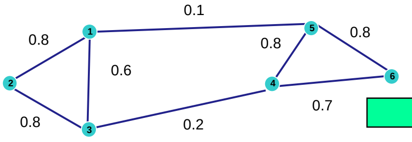
	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

- Important properties:
  - Eigenvalues are non-negative real numbers
  - Eigenvectors are real and orthogonal
  - Eigenvalues and eigenvectors provide an insight into the connectivity of the graph…

Normalized Laplacian

- **Laplacian matrix (L)**
  - $n \times n$ symmetric matrix

$$L = D^{-0.5}(D - A)D^{-0.5}$$

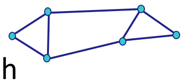


1.00	-0.52	-0.39	0.00	-0.06	0.00
-0.52	1.00	-0.50	0.00	0.00	0.00
-0.39	-0.50	1.00	-0.12	0.00	0.00
0.00	0.00	-0.12	1.00	0.47-	0.44-
-0.06	0.00	0.00	-0.47	1.00	0.50-
0.00	0.00	0.00	0.44-	0.50-	1.00

- Important properties:
  - Eigenvectors are real and normalized
  - Each A_{ij} (which i, j are not equal) = $\dfrac{-A_{ij}}{D_{ii}}$

Normalized Laplacian

1. Pre-processing
   - Build Laplacian matrix L of the graph



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	0	0	-0.2	1.7	-0.8	-0.7
x_5	-0.1	0	0	-0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

2. Decomposition
   - Find eigenvalues X and eigenvectors Λ of the matrix L
   - Map vertices to corresponding components of λ_2

$$\Lambda = \begin{matrix} 0.0 \\ 0.4 \\ 2.2 \\ 2.3 \\ 2.5 \\ 3.0 \end{matrix} \qquad X = \begin{matrix} 0.4 & 0.2 & 0.1 & 0.4 & -0.2 & -0.9 \\ 0.4 & 0.2 & 0.1 & -0. & 0.4 & 0.3 \\ 0.4 & 0.2 & -0.2 & 0.0 & -0.2 & 0.6 \\ 0.4 & -0.4 & 0.9 & 0.2 & -0.4 & -0.6 \\ 0.4 & -0.7 & -0.4 & -0.8 & -0.6 & -0.2 \\ 0.4 & -0.7 & -0.2 & 0.5 & 0.8 & 0.9 \end{matrix}$$

x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7

K-Eeigenvector Clustering

- ***K*-eigenvector Algorithm** (Ng et al.,'01)

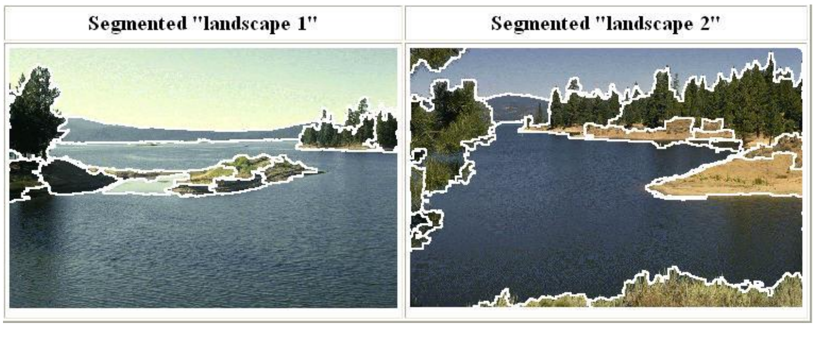
1. Pre-processing
   - Construct the **scaled adjacency matrix**

$$A' = D^{-1/2} A D^{-1/2}$$

2. Decomposition
   - Find the eigenvalues and eigenvectors of A'
   - Build embedded space from the eigenvectors corresponding to the k largest eigenvalues
3. Grouping
   - Apply k-means to the reduced $n \times k$ space to produce k clusters.

Mean-Shift Segmentation

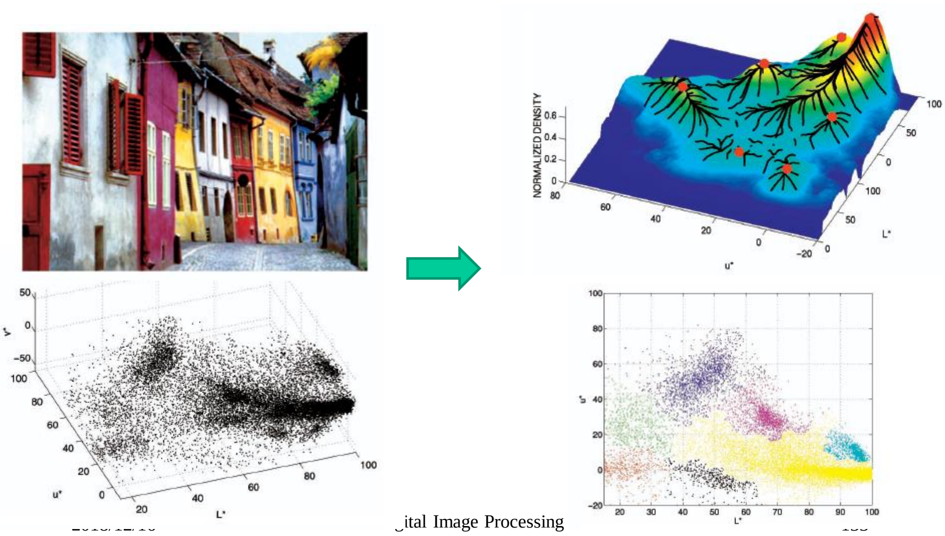
Versatile technique for clustering-based segmentation



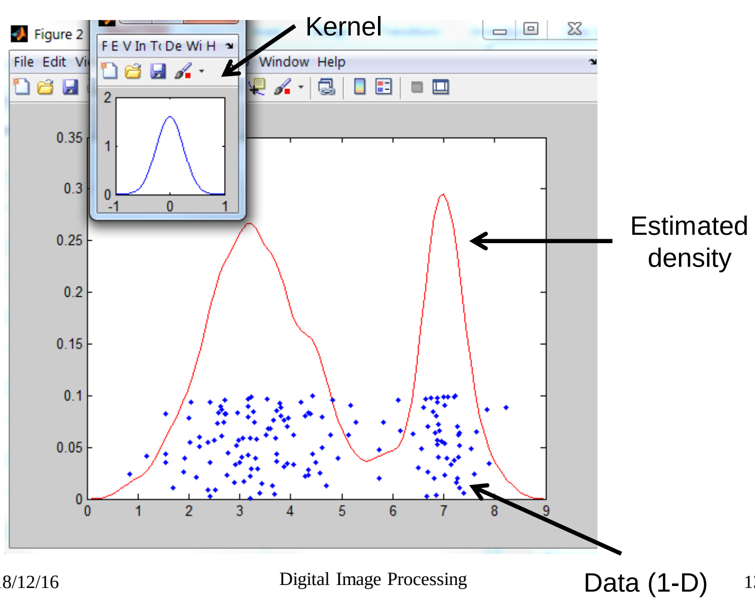
D. Comaniciu and P. Meer, Mean Shift: A Robust Approach toward Feature Space Analysis, IEEE T-PAMI 2002.

Mean-Shift Segmentation

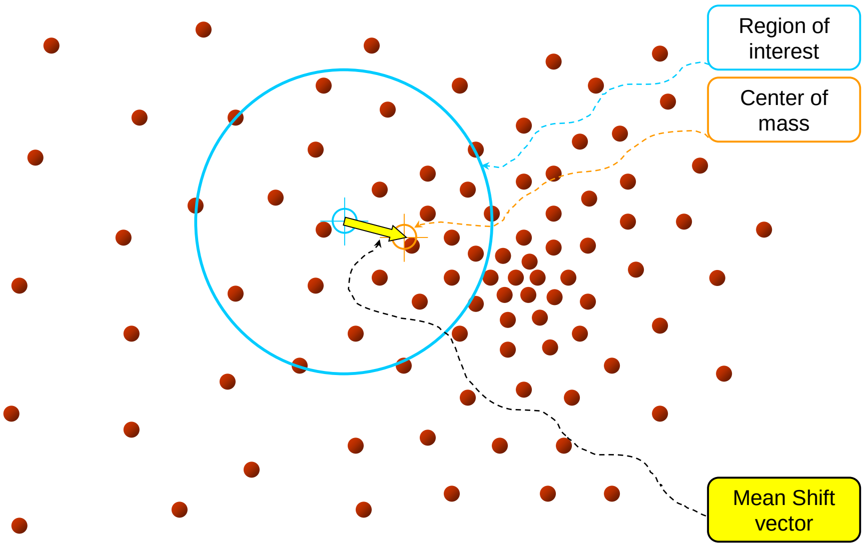
Try to find *modes* of this non-parametric density



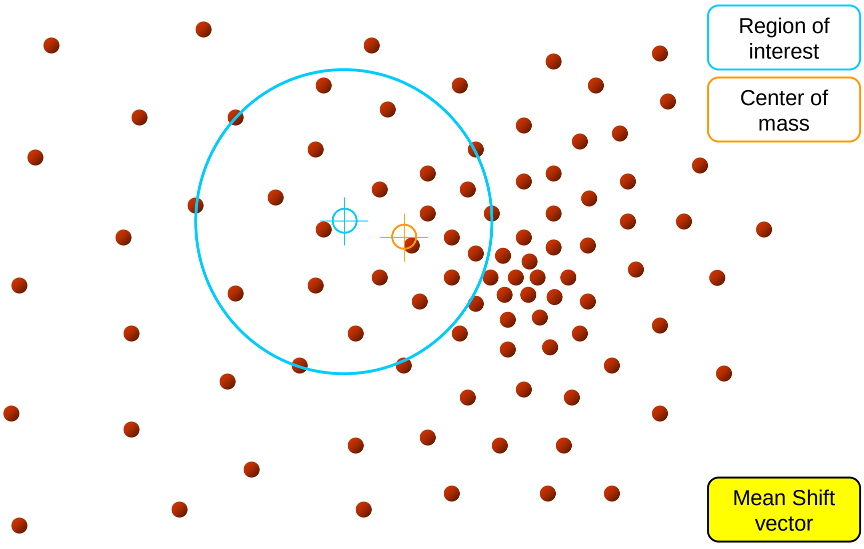
Kernel Density Estimation



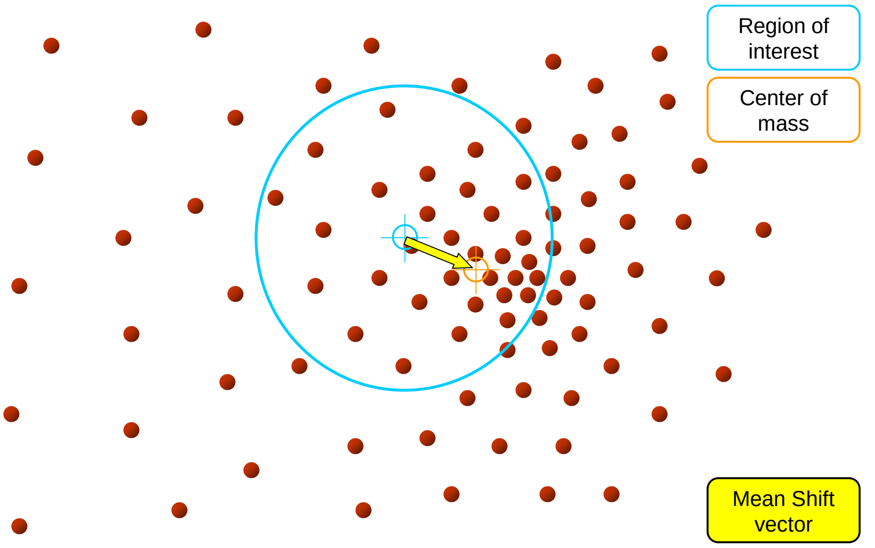
Mean Shift



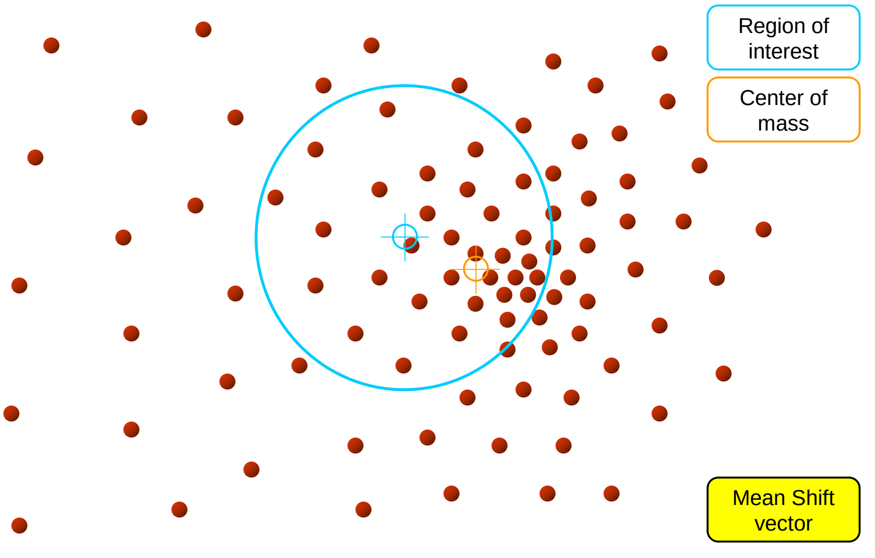
Mean Shift



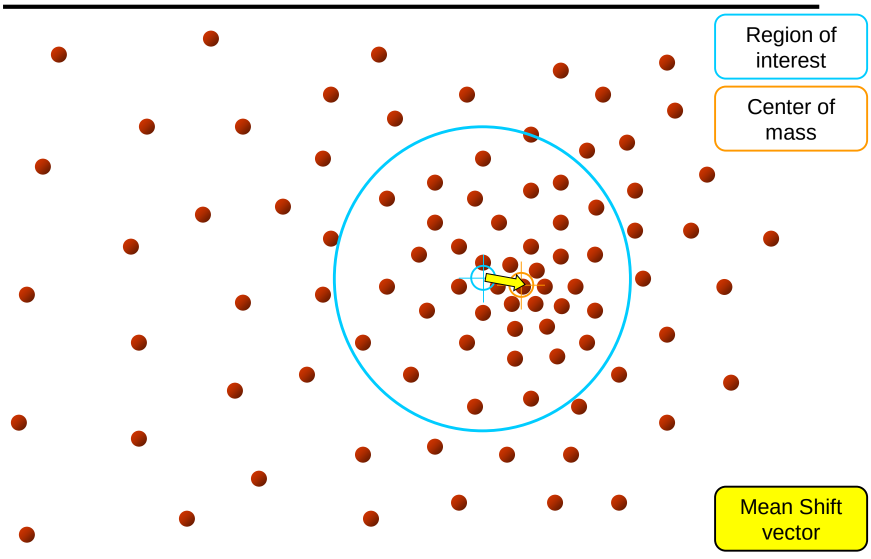
Mean Shift



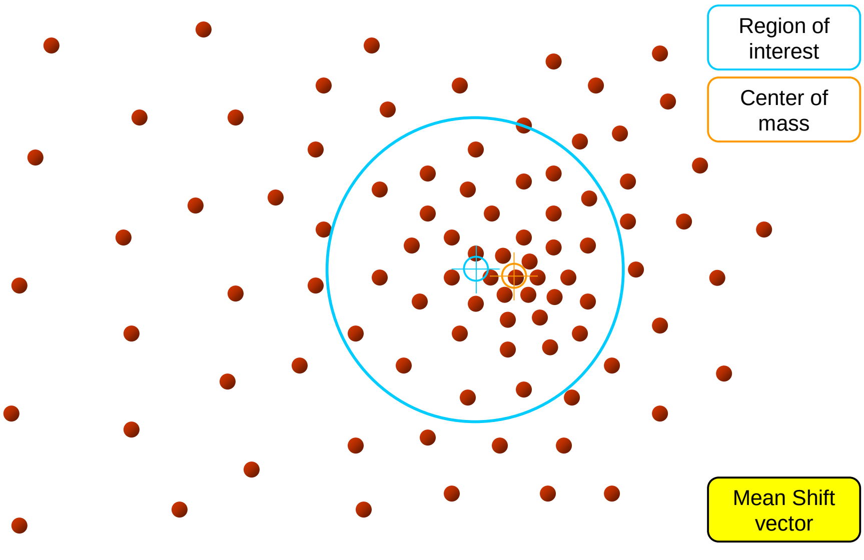
Mean Shift



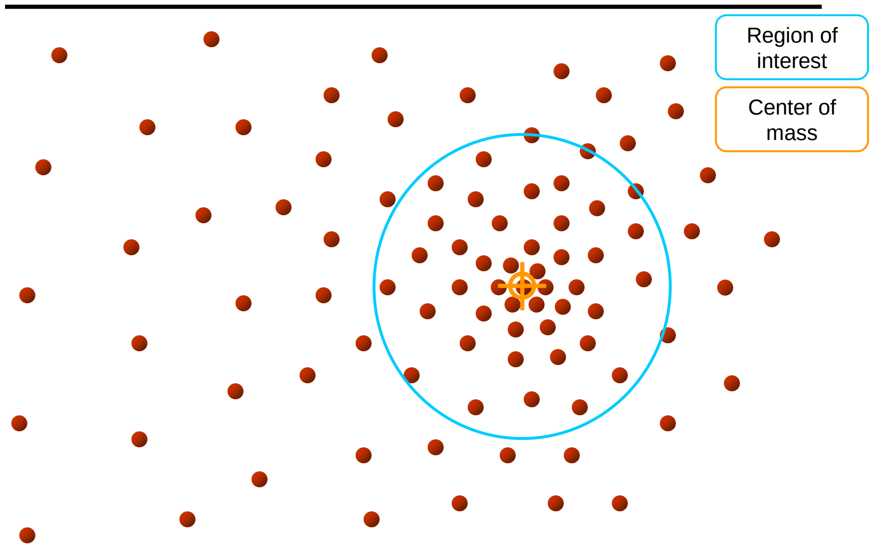
Mean Shift



Mean Shift



Mean Shift

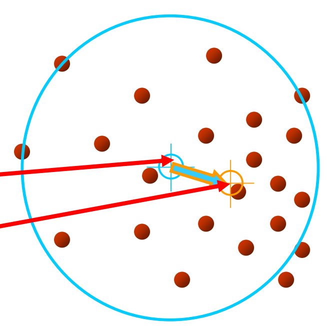


Mean Shift

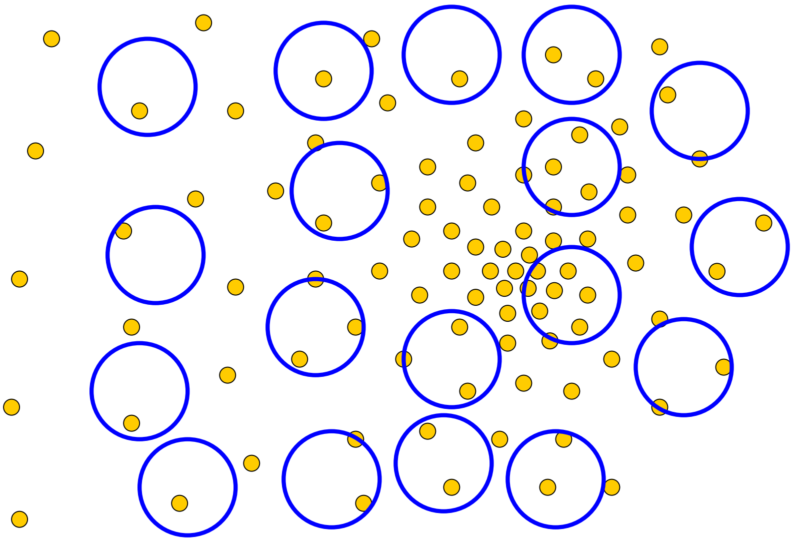
Simple Mean Shift procedure:

- Compute mean shift vector
- Translate the Kernel window by $\mathbf{m}(\mathbf{x})$

$$\mathbf{m}(\mathbf{x}) = \left[\frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{\left\| \mathbf{x} - \mathbf{x}_i \right\|^2}{h} \right)}{\sum_{i=1}^{n} g\left(\frac{\left\| \mathbf{x} - \mathbf{x}_i \right\|^2}{h} \right)} - \boxed{\mathbf{x}} \right]$$

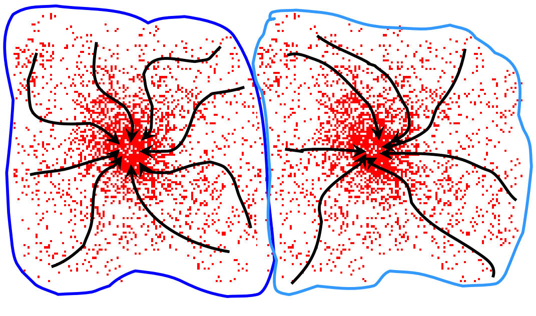


Real Modality Analysis

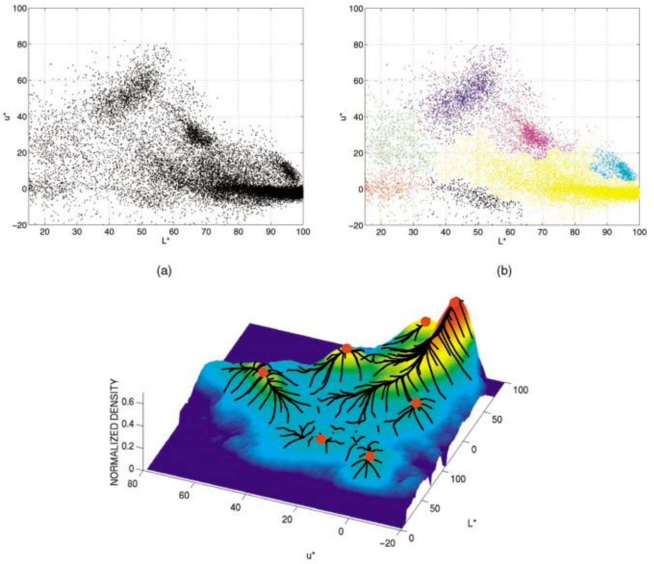


Attraction Basin

- **Attraction basin:** the region for which all trajectories lead to the same mode
- **Cluster:** all data points in the attraction basin of a mode



Attraction Basin

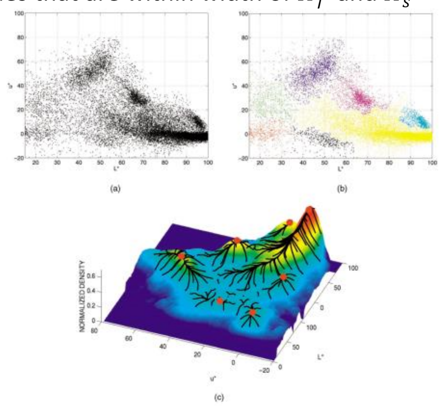


Mean-Shift Clustering

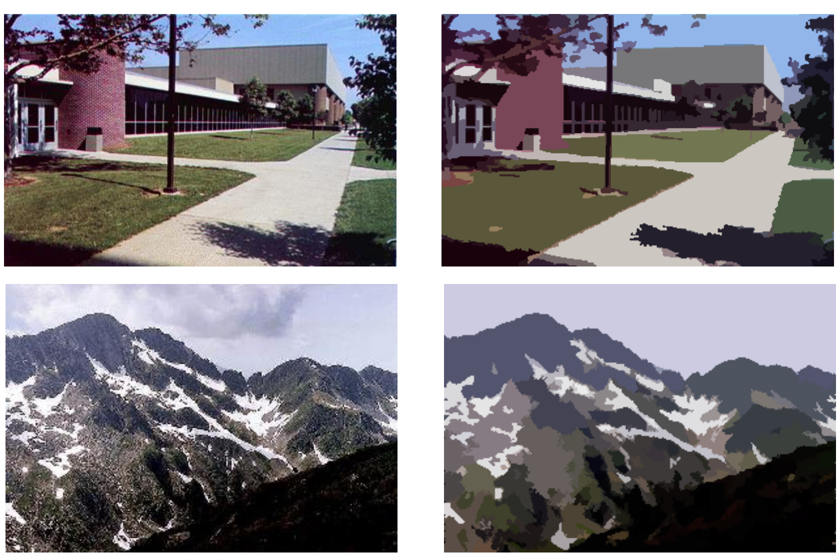
- The mean shift algorithm seeks *modes* of the given set of points
  1. Choose kernel and bandwidth
  2. For each point:
     a) Center a window on that point
     b) Compute the mean of the data in the search window
     c) Center the search window at the new mean location
     d) Repeat (b,c) until convergence
  3. Assign points that lead to nearby modes to the same cluster

Mean-Shift Segmentation

- Compute features for each pixel (color, gradients, texture, etc); also store each pixel's position
- Set kernel size for features K_f and position K_s
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge modes that are within width of K_f and K_s



Mean-Shift Segmentation



Mean-Shift Segmentation

Mean-Shift Pros & Cons

- Pros
  - Good general-purpose segmentation
  - Flexible in number and shape of regions
  - Robust to outliers
  - General mode-finding algorithm (useful for other problems such as finding most common surface normals)
- Cons
  - Have to choose kernel size in advance
  - Not suitable for high-dimensional features
- When to use it
  - Oversegmentation
  - Multiple segmentations
  - Tracking, clustering, filtering applications
    - D. Comaniciu, V. Ramesh, P. Meer: *Real-Time Tracking of Non-Rigid Objects using Mean Shift*, *Best Paper Award*, IEEE Conf. Computer Vision and Pattern Recognition (CVPR'00), Hilton Head Island, South Carolina, Vol. 2, 142-149, 2000