Image Segmentation

- Transitioning to algorithms with input as images and output as attributes extracted from those images
- Partition images into regions that are similar based on some criteria
- Thresholding, region growing, and region splitting and merging

Segmentation

- Subdivide an image into regions or objects
  - Group pixels in an image based on their low-level, local properties such as color or texture
    - Characterized by unsupervised learning
  - Convert an image into a representation that is more meaningful and easier to analyze
    - Extract image attributes, based on discontinuity and similarity of pixels
  - Level of detail dependent on problem being solved
    - Segmentation achieved when the objects or ROIs are detected
  - Assign a label to every pixel such that pixels with the same label share certain visual characteristics
  - One of the most difficult tasks in image processing
    - May be simplified by use of sensors for specific applications
    - IR sensor to detect heat signatures of objects
- Applications include
  - Medical imaging (locating tumors, studying anatomical structures)
  - Region classification in satellite images
  - Detection of disease in crops
  - Machine vision for industrial inspection
- Algorithms based on basic properties of intensity values: discontinuity and similarity
  - Discontinuity
    - Partition an image based on abrupt changes in intensity values, such as edges
    - Boundaries of regions are sufficiently different from each other and from the background
  - Similarity
    - Partition an image into regions that are similar according to some predefined criteria

Fundamentals

- Let R be the spatial region occupied by an image
- Image segmentation is the process that partitions R into n subregions $R_1, R_2, \ldots, R_n$ such that

1. $\bigcup_{i=1}^{n} R_i = R$
   - Union must be *complete*; every pixel must be in a region
2. R_i is a connected set, $i = 1, 2, \ldots, n$
   - Points in a region must be connected (8-connected)

3. $R_i \cap R_j = \emptyset \; \forall i, j | i \neq j$
   - Regions must be disjoint
4. $Q(R_i) = \texttt{true}$ for $i = 1, 2, \ldots, n$
   - $Q(R_k)$ is a logical predicate defined over the points in set R_k
5. $Q(R_i \cup R_j) = \texttt{false}$ for any adjacent region R_i and R_j
   - Adjacent regions must be different

- Two regions R_i and R_j are said to be adjacent if their union forms a connected set
- Regions are disjoint if their union is not connected

- Edge-based segmentation
  - Used when boundaries of regions are sufficiently different from each other and from the background to allow detection based on discontinuities
- Region-based segmentation
  - Partition an image into regions based on similarity of pixels
- Figure 10.1

Thresholding

- Simplest method for image segmentation
- Based on a threshold value to turn a grayscale image into a binary image
  - Easy if the histogram is characterized by two peaks corresponding to foreground and background regions that are separated by a valley
- Highly dependent on selection of the threshold value
- May be better to preprocess the image before applying thresholding
  - Smooth the image by convolving with a Guassian, to make the histogram peaks more distinctive
  - Compute the intensity edges and apply algorithms only to pixels near the edges to reduce the asymmetry in the size of histogram peaks when foreground objects are small with respect to background

Point, line, and edge detection

- Detection of sharp, local changes in intensity
  - Interested in isolated points, lines, and edges
  - Edge pixels
    - Pixels at which the intensity of an image function changes abruptly
  - Edges
    - Sets of connected edge pixels
  - Edge detectors
    - Local image processing methods designed to detect edge pixels
  - Line

    - An edge segment in which intensity of the background on either side of the line is much higher or much lower than the intensity of line pixels
  - Isolated point
    - A line with length and width equal to one pixel

- Background
  - Local averaging smooths an image
    - Since averaging is same as integration, abrupt changes can be detected by differentiation, using first- and second-order derivatives
  - Derivatives of a digital function
    - Computed in terms of differences in neighboring pixels
    - Approximation used for a first derivative
      1. Must be zero in areas of constant intensity
      2. Must be nonzero at the onset of an intensity step or ramp
      3. Must be nonzero at points along an intensity ramp
    - Approximation used for a second derivative
      1. Must be zero in areas of constant intensity
      2. Must be nonzero at the onset *and* end of an intensity step or ramp
      3. Must be zero along intensity ramps
    - The maximum possible intensity change for digital [finite] quantities is finite
    - Shortest distance over which a change can occur is adjacent pixels
  - Approximation to the first-order derivative at point x of a 1D function $f(x)$
    - Expand the function $f(x + \Delta x)$ into a Taylor series about x
    - Let $\Delta x = 1$
    - Keep only the digital terms, we get the *forward difference* as

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

    - *Backward difference* is computed as

$$\frac{\partial f}{\partial x} = f'(x) = f(x) - f(x-1)$$

    - *Central difference* is computed by

$$\frac{\partial f}{\partial x} = f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

  - Second derivative is computed by differentiating the first derivative

$$\begin{aligned}\frac{\partial^2 f}{\partial x^2} &= \frac{\partial f'(x)}{\partial x} \\ &= f'(x+1) - f'(x) \\ &= (f(x+2) - f(x+1)) - (f(x+1) - f(x)) \\ &= f(x+2) - 2f(x+1) + f(x)\end{aligned}$$

    - This expression is about point $x + 1$
    - Translate it to point x to get

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

  - Verify that the expressions for first- and second-order derivatives satisfy the conditions outlined
  - Figure 10.2
    * Ramp edges, step edges, roof edges
    * First-order derivative is nonzero at the onset and along the entire intensity ramp, producing thick edges
    * Second-order derivative is nonzero only at the onset and end of ramp, producing finer edges
    * For isolated noise point, second-order derivative has a much stronger response compared to the first-order derivative
      · Second-order derivative is much more aggressive than first-order derivative in enhancing sharp changes
      · Second-order derivative enhances fine detail (including noise) much more than the first-order derivative
    * In both ramp and step edges, second-order derivative has opposite signs as it transitions into and out of an edge
      · The *double-edge* effect is used to locate edges
  - Summary of observations
    1. First-order derivatives produce thicker edges in an image
    2. Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise
    3. Second-order derivatives produce a double-edged response at ramp and step transitions in intensity
    4. Sign of second derivative can be used to determine whether the transition into an edge is from light to dark or dark to light
  - Use spatial filters [convolution kernels] to compute first and second derivatives at every pixel location
    * Response of filter at center point of a 3×3 region is

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 \\ &= \sum_{k=1}^{9} w_k z_k \end{aligned}$$

    * Spatial filter mask given by

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

- Detection of isolated points
  - Based on second derivative using the Laplacian

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

  - Partials are given by

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

  - Laplacian is given by

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

  - Implemented by a 4-neighborhood mask, or can be extended to an 8-neighborhood mask

1	1	1
1	-8	1
1	1	1

  - Figure 10.4
    * Select a point in the output image if its response to the mask exceeds a specified threshold; deselect others

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

    * T selected as 90% of highest absolute pixel value in Figure 10.4c
  - Note that in constant intensity areas, the mask response will be zero

- Line detection
  - Second derivative results in a stronger response and produces thinner lines compared to first derivative
    * Use Laplacian mask (Figure 10.4a) for line detection
    * Handle double-line effect of the second derivative properly
  - Example – Figure 10.5
    * Scale the image to handle negative values (Fig 10.5b)
      · Mid-gray represents zero
      · Darker shades represent negative values
      · Lighter shades represent positive values
      · Shows double line effect in the magnified region
    * Using absolute values doubles the thickness of lines (Fig 10.5c)
    * Use only positive values of Laplacian (Figure 10.5d)
    * Thicker lines separated by a *valley*
  - Laplacian detector is isotropic; response independent of direction
  - Lines in specific directions can be detected by other masks (Fig 10.6)
    * Different masks give different response depending on the direction of lines
    * Example – Figure 10.7

- Edge models
  - Edge detection used to segment images based on abrupt (local) change in intensity
  - Edge models – classified by their intensity profiles

    Step edge Figure 10.8a
      * Transition between two intensity levels occurring over a distance of single pixel
      * Occur in computer-generated images for solid modeling and animation
      * Used frequently as edge models in algorithm development (Canny edge detector)

    Ramp edge Figure 10.8b
      * Practical images have blurry and noisy edges
        · Blurring depends on optical focusing
        · Noise depends on electronic components
      * Slope of ramp is inversely proportional to the degree of blur
      * Edge point is not well defined but contained in any point in the ramp

    Roof edge Figure 10.8c
      * Models of lines through a region
      * Base of roof edge determined by the thickness and sharpness of line
      * Found in range images when thin objects (pipes) are closer to the sensor than their equidistant background (walls)
      * Also found in digitization of line drawings and satellite imagery (roads)
  - Edge models allow us to write mathematical expressions for edges to develop image processing algorithms

  - Figure 10.9 to show all three edge types
  - Figure 10.10
    * Intensity profile and its first and second derivatives
      · First derivative positive at the onset of ramp and at points on the ramp; zero in the areas of constant intensity
      · Second derivative positive at the onset of ramp, negative at the end of ramp, and zero at points on ramp as well as constant intensity points
    * Zero crossing of the second derivative
      · Intersection between the intensity axis and a line extending between the extrema of the second derivative
    * Use magnitude of the first derivative to detect the presence of an edge
    * Sign of second derivative indicates whether an edge pixel lies on the dark or bright side of the edge
    * Additional properties of second derivative
      1. Produces two values for every edge in an image (undesirable feature)
      2. Zero crossing can be used to locate center of thick edges
  - Example 10.4 – Effect of noise
    * Figure 10.11
    * Four ramp edges that transition from black region to white region
      · Top image – noise free
      · Next three images corrupted by additive Gaussian noise with zero mean and standard deviation of 0.1, 1.0, and 10.0 intensity levels
      · Graph below each image shows a horizontal intensity profile passing through the center of the image
      · Middle column shows the first derivative while the right column shows the second derivative
      · As you move own the center column, the derivatives become increasingly different from the noiseless case
      · Difficult to associate the last profile in the center column with the first derivative of a ramp edge
      · Second derivative is even more sensitive to noise
    * This shows the importance of image smoothing prior to edge detection
  - Three fundamental steps in edge detection
    1. Image smoothing for noise reduction
    2. Detection of edge points
    3. Edge localization

- Basic edge detection
  - Image gradient and its properties
    * Find edge strength and direction at location (x, y) in image f
    * Use the gradient ∇f, defined as a vector

$$\nabla f = \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

      · Vector ∇f points in the direction of greatest rate of change of f at (x, y)
    * Magnitude of vector $M(x, y)$ gives the rate of change in the direction of the gradient vector

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

    * g_x, g_y, and $M(x, y)$ are images of the same size as original image
      · $M(x, y)$ known as gradient image, or gradient
    * Direction of the gradient vector with respect to x axis is given by

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

- $\alpha(x, y)$ is also an image of the same size as original array

* Figure 10.12
  - Compute partial derivatives over 3×3 neighborhood

    Partial derivative along x axis is computed by subtracting pixels in left row from corresponding pixels in right row

    Partial derivative along y axis is computed by subtracting pixels in upper row from corresponding pixels in lower row
  - Using the differences to implement partial derivatives, we get $\partial f/\partial x = -2$ and $\partial f/\partial y = 2$

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

  - This gives $M(x, y) = 2\sqrt{2}$
  - Similarly, $\alpha(x, y) = \tan^{-1}(g_y/g_x) = -45°$, or $135°$ in counterclockwise direction with respect to x-axis
  - Direction of the edge at a point is orthogonal to the gradient vector at that point
  - Therefore, direction of edge in the example is $135° - 90° = 45°$
* Gradient vector may be called *edge normal*
* If gradient vector is normalized to unit length, the resulting vector may be called *edge unit normal*

– Gradient operators

* Computing gradient requires the computation of $\partial f/\partial x$ and $\partial f/\partial y$ at every pixel location
* Digital approximation of partial derivatives over a neighborhood is given by

$$\begin{aligned} g_x = \frac{\partial f(x,y)}{\partial x} &= f(x+1, y) - f(x, y) \\ g_y = \frac{\partial f(x,y)}{\partial y} &= f(x, y+1) - f(x, y) \end{aligned}$$

  - Implemented using 1D masks of Figure 10.13
* Diagonal edge direction computed by 2D mask
* Robert cross-gradient operator in 3×3 neighborhood using Figure 10.14

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

$$\begin{aligned} g_x = \frac{\partial f}{\partial x} &= z_9 - z_5 \\ g_y = \frac{\partial f}{\partial y} &= z_8 - z_6 \end{aligned}$$

* Simplest digital approximations to the partial derivatives using 3×3 masks are given by Prewitt operators as

$$\begin{aligned} g_x = \frac{\partial f}{\partial x} &= (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3) \\ g_y = \frac{\partial f}{\partial y} &= (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7) \end{aligned}$$

* Sobel's operator uses a 2 in the center location to provide image smoothing (noise suppression)

$$\begin{aligned} g_x = \frac{\partial f}{\partial x} &= (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \\ g_y = \frac{\partial f}{\partial y} &= (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \end{aligned}$$

    - Coefficients of all masks in Figure 10.14 add to zero, giving zero response in areas of constant intensity
    - Operator masks give the gradient components g_x and g_y at every pixel location in the image
        - Used to estimate edge strength and direction
        - Magnitude may be approximated by
        $$M(x, y) \approx |g_x| + |g_y|$$
        - Resulting filters may not be isotropic (invariant to rotation)
    - Figure 10.15
        - Kirsch compass kernels
        - Modification of Prewitt and Sobel masks to get maximum response along the diagonal directions
        - Detect edge magnitude as the response of the kernel with the strongest response at that point, out of eight kernels, representing eight compass directions
        - Assign direction based on the selected kernel

          Differentiate between north and south instead of just a vertical edge as in Sobel kernel
        - Consider the edge given by

·	·	·	·	·
·	1	1	0	·
·	1	1	0	·
·	1	1	0	·
·	·	·	·	·

        - Application of kernels on center pixel gives the response as

          N = -15

          NW = -7

          W = 1

          SW = 9

          S = 9

          SE = 9

          E = 1

          NE = -7
    - Example: Figure 10.16
        - Directionality of horizontal and vertical component
    - Figure 10.17: Gradient angle image
        - Not very useful, but plays a key role in Canny edge detector
    - Figure 10.18: Smoothed using a 5×5 averaging filter prior to edge detection (Remove brick boundaries)
    - Figure 10.19: Diagonal edge detection
  - Combining the gradient with thresholding
    - Edge detection can be made more selective by smoothing the image before computing gradient
    - Can also threshold the gradient image
    - Figure 10.20: Thresholded version of gradient image

- More advanced techniques for edge detection
  - Account for image noise and nature of edges
  - Marr-Hildreth edge detector
    - Arguments
      1. Intensity changes are not independent of image scale and so, their detection requires the use of operators of different sizes
      2. A sudden intensity change will give rise to a peak or trough in the first derivative, or to a zero crossing in second derivative

* Desired features of edge operators
  1. It should be a differential operator capable of computing a digital approximation of the first or second derivative at every point in the image
  2. It should be capable of being tuned to act at any desired scale
     - Large operators can be used to detect any blurry edges
     - Small operators can detect sharply focused fine detail
* Most satisfactory operator for those conditions is $\nabla^2 G$, called *Laplacian of a Gaussian* or LoG

$$\begin{aligned} \nabla^2 &= \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \\ G(x,y) &= e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned}$$

with standard deviation σ

$$\begin{aligned} \nabla^2 G(x,y) &= \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} \\ &= \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}\right] \\ &= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \\ &= \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \end{aligned}$$

* Figure 10.21
  - Zero crossing of the LoG occurs at $x^2 + y^2 = 2\sigma^2$ which defines a circle of radius $\sqrt{2}\sigma$ centered on the origin
  - LoG is also known as the *Mexican hat* operator
  - 5×5 mask
  - Positive central term surrounded by an adjacent negative region with decreasing values away from the center
* Generate masks of arbitrary size by sampling and making sure that the coefficients sum to zero
* LoG operator
  1. Gaussian part blurs the image
     - Reduces the intensity of structures (including noise) at scales much smaller than σ
     - Smooth in both spatial and frequency domains; less likely to introduce artifacts (ringing)
  2. Laplacian (∇^2)
     - First derivatives are directional operators
     - Laplacian is isotropic
     - No need for multiple kernel to find the strongest response at any point
* Algorithm convolves the LoG filter with an input image $f(x,y)$

$$g(x,y) = [\nabla^2 G(x,y)] \bigstar f(x,y)$$

  - Find zero crossings in $g(x,y)$ to determine the edge locations
  - Can also write the equation as (for linear processes)

$$g(x,y) = \nabla^2 [G(x,y) \bigstar f(x,y)]$$

  - Smooth the image first with a Gaussian and then, apply Laplacian
* Summary of algorithm
  1. Filter the input image with $n \times n$ Gaussian lowpass filter

2. Compute the Laplacian of the resulting image
3. Find the zero crossings

* Size of the Gaussian kernel
  - Values of a Gaussian function at a distance larger than 3σ from the mean are small enough so that they can be ignored
  - Use a kernel of size $\lceil 6\sigma \times 6\sigma \rceil$
  - Use the smallest odd integer $> 6\sigma$
* Finding zero crossings
  - Use a 3×3 neighborhood centered on pixel
  - Zero crossing implies that signs of two of the opposing neighboring pixels must differ
  - Four cases: left/right, up/down, and two diagonals
  - If using a threshold, the absolute value of the difference should exceed the threshold
* Example: Figure 10.22
* Possible to approximate the LoG filter by a difference of Gaussians (DoG)

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

  with $\sigma_1 > \sigma_2$
* To make meaningful comparisons between LoG and DoG, σ for LoG must be selected so that LoG and DoG have the same zero crossings

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

– Canny edge detector
* Superior performance compared to other edge detectors
* Three objectives
  1. Low error rate
     - No spurious responses
     - All edges should be found
  2. Edge points should be well localized
     - As close as possible to true edges
     - The distance between a point marked as edge and the center of true edge should be minimum
  3. Single edge point response
     - Only a single point for each true edge point
     - Number of local maxima around the true edge should be minimum
     - Detector should not identify multiple edge pixels where only a single edge point exists
* Good approximation to the above in 1D by first derivative of a Gaussian

$$\frac{d}{dx} e^{-\frac{x^2}{2\sigma^2}} = \frac{-x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}}$$

* In 2D, apply 1D approach in the direction of edge normal
  - Direction of edge normal is unknown
  - So, apply it in all possible directions
  - Approximate by first smoothing with a circular 2D Gaussian function, compute the gradient of the result, and then use the gradient magnitude and direction to estimate edge strength and direction at every point
  - For input image $f(x,y)$, denote the Gaussian function by

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Form smoothed image $f_s(x, y)$ by convolving G and f

$$f_s(x, y) = G(x, y) \bigstar f(x, y)$$

- Compute the gradient magnitude and direction

$$\begin{aligned} M(x, y) &= ||\nabla f_s(x, y)|| \\ &= \sqrt{g_x^2 + g_y^2} \\ \alpha(x, y) &= \tan^{-1}\left[\frac{g_y}{g_x}\right] \end{aligned}$$

  where $g_x = \partial f_s/\partial x$ and $g_y = \partial f_s/\partial y$
- $M(x, y)$ typically contains wide ridges around local maxima
- Thin those ridges using *nonmaxima suppression*
- Specify a number of discrete orientations of the edge normal (gradient vector)
- In a 3×3 region, define four orientations as horizontal, vertical, and $\pm 45^\circ$ (Figure 10.24a)
- Define a range of directions over which an edge is considered to be horizontal (Figure 10.24b)
- Figure 10.24c: Angle ranges corresponding to four directions under consideration
- Denote four basic directions by: $d_1 =$ horizontal, $d_2 = -45^\circ$, $d_3 =$ vertical, and $d_4 = +45^\circ$
- Formulate the nonmaxima suppression scheme for a 3×3 region centered at every point (x, y) in $\alpha(x, y)$
  1. Find the direction d_k that is closest to $\alpha(x, y)$
  2. If the value of $M(x, y)$ is less than at least one of its two neighbors along d_k, $g_N(x, y) = 0$ (suppression); otherwise, $g_N(x, y) = M(x, y)$
- Above steps yield a nonmaxima suppressed image g_N
- Threshold $g_N(x, y)$ to reduce false edge points
- With single thresholding
  1. Threshold too low implies false positives
  2. Threshold too high eliminates valid edge points (false negatives)
- Hysteresis thresholding using two thresholds – low threshold T_L and high threshold T_H; $T_H : T_L$ should be $3 : 1$ or $2 : 1$
- Thresholding creates two additional images

$$\begin{aligned} g_{\text{NH}}(x, y) &= g_N(x, y) \geq T_H \\ g_{\text{NL}}(x, y) &= g_N(x, y) \geq T_L \end{aligned}$$

- All the nonzero pixels in g_{NH} are contained in g_{NL} as the latter image is created with a lower threshold
- Eliminate all the nonzero pixels contained in g_{NH} from g_{NL}

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

- Now, the nonzero pixels in $g_{NH}(x, y)$ and $g_{NL}(x, y)$ are considered *strong* and *weak* edge pixels
- All strong edge pixels are marked as valid pixels
- Edges in $g_{NH}(x, y)$ may have gaps that are filled by
  1. Locate the next unvisited pixel p in $g_{NH}(x, y)$
  2. Mark as valid edge pixels all the weak pixels in $g_{NL}(x, y)$ that are connected to p using 8-connectivity
  3. If all nonzero pixels in $g_{NH}(x, y)$ have been visited, go to step 4, otherwise go to step 1
  4. Set to zero all pixels in $g_{NL}(x, y)$ that did not get marked as valid edge pixels
- Form the output by adding $g_{NH}(x, y)$ and $g_{NL}(x, y)$

∗ Summary of Canny edge detector

1. Smooth the input image by a Gaussian filter

2. Compute the gradient magnitude and angle images
3. Apply nonmaxima suppression to gradient magnitude image
4. Use double thresholding and connectivity analysis to detect and link edges

* Typically, follow the detector by an edge-thinning algorithm
* Figure 10.25
* Example 10.9: Figure 10.26

– Quality vs speed comparisons
* Complex implementation
* Real-time applications may require higher speed than possible with Canny
* Good choice when the quality of edges is important

- Edge linking and boundary detection
– Pixels may not characterize edges completely because of noise, breaks in edges due to nonuniform illumination, and other spurious discontinuities in intensity values
– Follow edge detection by linking algorithms to assemble edge pixels into meaningful edges or region boundaries
– Local processing
* Requires knowledge of edge points in a local region such as 3×3 neighborhood
* Analyze characteristics of pixels in the neighborhood about every edge point pixel (x, y)
* Link all pixels that are similar based on a predefined criteria
* Establish similarity by
1. Edge strength/magnitude
· Let S_{xy} be the coordinates of a neighborhood centered at (x, y)
· Edge pixel at $(s, t) \in S_{xy}$ is similar in magnitude to pixel at (x, y) if

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

where E is a positive threshold
2. Direction of gradient vector
· Edge pixel at $(s, t) \in S_{xy}$ has an angle similar to pixel at (x, y) if

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

where A is a positive angle threshold
* Pixel at (s, t) is linked to pixel at (x, y) if both magnitude and direction conditions are satisfied
* Computationally expensive to examine each pixel with each of its neighbors
* Simplified method (good for real-time applications)
1. Compute gradient magnitude and angle arrays $M(x, y)$ and $\alpha(x, y)$ of input image $f(x, y)$
2. Compute binary image $g(x, y)$ as

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

3. Scan the rows of g and set to 1 all gaps (0's) in each rows that do not exceed a specified length L
4. Detect gaps in other directions θ by rotating g by θ and apply horizontal scanning from last step.
* Example 10.10 – Edge linking using local processing – Fig 10.27
– Global processing using Hough transform
* Good to work in unstructured environments
* All pixels are candidates for linking and are accepted/eliminated based on predefined global properties, such as their location on curves of specified shapes

- Curves form the edges or boundaries of regions of interest

* Given n points, find subset of points hat lie on straight lines
  - Find all lines determined by every pair of points
  - Find all subsets of points that are close to particular lines
  - All pair of points implies finding $n(n-1)/2-n^2$ lines; then performing $n(n(n-1)/2-n^2)$comparisons of every point to all lines

* Hough transform
  - Let (x, y) be a point in the xy-plane
  - General equation of a straight line in slope-intercept form is $y_i = ax_i + b$
  - There are infinite number of lines passing through (x_i, y_i) satisfying the above equation with different values of a and b
  - Rewrite the equation as $b = -x_i a + y_i$
  - Considering the ab-plane (parameter space) yields the equation of a single line for a fixed point (x_i, y_i)
  - A second point (x_j, y_j) also has a single line in parameter space, intersecting the first line with (x_i, y_i) at some point (a', b') in parameter space
  - a' is the slope and b' is the intercept of the line containing both (x_i, y_i) and (x_j, y_j) in the xy-plane (assume that the lines are not parallel)
  - All points on this line have lines in parameter space that intersect at (a', b')
  - Figure 10.28
  - Parameter space lines corresponding to all points (x_k, y_k) can be plotted
  - Principle lines can be found by identifying points in parameter space where large number of parameter-space lines intersect
  - Problem: a approaches infinity for almost vertical lines
  - Solved by using normal representation of a line

$$x\cos\theta + y\sin\theta = \rho$$

  - Figure 10.29: Geometrical interpretation of parameters ρ and θ
  - A horizontal line $(\theta = 0)$ has ρ as positive x-intercept
  - A vertical line $(\theta = \frac{\pi}{2})$ has ρ as positive y-intercept, or $(\theta = -\frac{\pi}{2})$ has ρ as negative intercept $[-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}]$
  - Each sinusoidal curve in Fig. 10.29b represents family of lines passing through a particular point (x_k, y_k) in the xy-plane
  - Intersection point (ρ', θ') in Fig. 10.29b gives the line passing through both (x_i, y_i) and (x_j, y_j) in Fig. 10.29a
  - Hough transform divides $\rho\theta$ parameter space into *accumulator cells* (Fig. 10.29c) where $(\rho_{\min}, \rho_{\max})$ and $(\theta_{\min}, \theta_{\max})$ are expected ranges of parameter values, $-\frac{\pi}{2} \leq \theta \leq \frac{\pi}{2}$ and $-D \leq \rho \leq D$ where D is the maximum distance between opposite corners in an image
  - Cell at coordinates (i, j) with accumulator value $A(i, j)$ corresponds to the square associated with parameter-space coordinates (ρ_i, θ_j)
  - Initialize all cells to zero
  - For every non-background point (x_k, y_k) in the xy-plane, let θ equal each of the allowed subdivision values on the θ-axis and solve for corresponding ρ using

$$\rho = x_k\cos\theta + y_k\sin\theta$$

  - Round resulting ρ values to the nearest allowed cell value along the ρ axis
  - If a choice of θ_q gives the solution ρ_p, increment $A(p, q)$ by 1
  - At the end, a value K in a cell $A(i, j)$ means that there are K points in the xy-plane on the line $x\cos\theta_j + y\sin\theta_j = \rho_i$
  - Number of subdivisions in the $\rho\theta$-plane determines the accuracy of the colinearity of those points

        - Number of computations is linear with respect to n, the number of non-background points in the xy-plane
    - Example 10.11: Some basic properties of Hough transform
        - Figure 10.30
        - Image of size 101×101 with five labeled white points
        - Points mapped onto $\rho\theta$ plane using subdivisions of one unit for ρ and θ axes
        - Range of θ: $\pm\frac{\pi}{2}$
        - Range of ρ: $\pm\sqrt{2} \cdot 101$
        - Horizontal line resulting from mapping of point 1 is sinusoid of zero amplitude
        - Points labeled A and B show the colinearity detection property of Hough transform
        - Point B marks the intersection of curves corresponding to points 2, 3, 4 in xy-plane
        - Location of point A indicates that these points lie on a straight line passing through the origin ($\rho = 0$) and oriented at $-\frac{\pi}{4}$
        - Curves intersecting at point B in parameter space indicate that points 2, 3, 4 are on a straight line oriented at $\frac{\pi}{4}$ with distance from origin $\rho = 71$
    - Example 10.12: Using Hough transform for edge linking
        - Figure 10.31

Thresholding

- Central to image segmentation
- Foundation
  - Partition images directly into regions based on intensity values and/or properties of these values
  - Basics of intensity thresholding
    - Figure 10.32a: Intensity histogram
    - Light objects on a dark background
    - Intensity values grouped into two distinct modes
    - Select a threshold T that separates these modes
    - The segmented image $g(x, y)$ is given by

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

      where the points labeled 1 are called *object points* and other points are called *background points*
    - If T is constant in the above expression, the thresholding is called *global thresholding*
    - When the value of T changes over an image, the thresholding is called *variable thresholding*
    - Local or regional thresholding
        - Variable thresholding in which the value of T at any point depends on its neighborhood
        - If T depends on the location itself, the thresholding is called *dynamic* or *adaptive* thresholding
    - Figure 10.32b
        - Histogram with three dominant modes
        - Possibly two types of light objects on a dark background
        - Use multiple thresholding to classify some points to the background and other points to two separate objects

$$g(x,y) = \begin{cases} a & \text{if } f(x,y) > T_2 \\ b & \text{if } T_1 < f(x,y) \leq T_2 \\ c & \text{if } f(x,y) \leq T_1 \end{cases}$$

    - Success of thresholding is directly related to the width and depth of valleys separating the histogram modes

    - Key factors affecting the valley properties
        1. Separation between peaks
        2. Noise content of the image
        3. Relative sizes of objects and background
        4. Uniformity of illumination source
        5. Uniformity of reflectance properties of the image
  - Role of noise in image thresholding
    - Figure 10.33
    - No noise yields two spikes in the histogram; trivial to segment
    - Gaussian noise of zero mean and σ of 10; broader modes but good separation
    - Gaussian noise of zero mean and σ of 50; impossible to separate modes based on histogram
  - Role of illumination and reflectance
    - Figure 10.34
    - Noisy image of Figure 10.33b
    - Multiply this image by a nonuniform intensity function, such as intensity ramp in the middle
    - Deep valley between peaks gets corrupted and modes cannot be easily separated
    - Nonuniform reflectance in the objects will have similar effect
  - Noise and illumination may not be easily controllable
    - Correct the shading pattern directly
        - Nonuniform (but fixed) illumination corrected by multiplying the image by the inverse of the pattern, obtained by imaging a flat surface of constant intensity
    - Attempt to correct global shading pattern via processing
    - Work around nonuniformities using variable thresholding

- Basic global thresholding
  - Automatic estimation of global threshold value
    1. Select an initial estimate for global threshold T
    2. Segment the image using T, giving G_1 as group of object pixels and G_2 as group of background pixels ($G_1 > T; G_2 \leq T$)
    3. Compute average intensity m_1 and m_2 corresponding to G_1 and G_2, respectively
    4. Compute a new threshold value
    $$T = \frac{1}{2}(m_1 + m_2)$$
    5. Repeat steps 2 to 4 until the difference between values of T in successive iterations is smaller than a predefined parameter ΔT
  - Process efficiency can be improved by modifying steps 2-4 to work with histogram instead of the whole image
  - Works well when there is a reasonably clear valley between the modes of the histogram corresponding to objects and background
  - ΔT is used to control the number of iterations
  - Initial threshold value should be somewhere between the minimum and maximum intensity values in the image
    - Average intensity of the image is a good initial value
  - Figure 10.35

- Optimum global thresholding using Otsu's method
  - Thresholding viewed as a statistical-decision theory problem whose objective is to minimize average error in assigning pixels to two or more classes

- Closed form solution known as Bayes decision function
- Solution based on two parameters
  1. Probability density function of intensity levels in each class
  2. Probability that each class occurs in a given application
- Estimating PDFs not trivial; simplified by making workable assumptions
- Otsu's method
  - Maximizes the between-class variance
  - Basic idea: Properly thresholded classes should be distinct with respect to intensity values of their pixels
    - A threshold giving the best separation between classes would be the optimum threshold
    - Method based entirely on the histogram of the image
  - L integer intensity levels $\{0, 1, 2, \ldots, L-1\}$
  - Image size $M \times N$ pixels
  - Number of pixels at intensity level i denoted by n_i

$$MN = n_0 + n_1 + \cdots n_{L-1}$$

  - Normalized histogram has components $p_i = n_i/MN$

$$\sum_{i=0}^{L-1} p_i = 1 \qquad\qquad p_i \geq 0$$

  - Select a threshold $T(k) = k, 0 < k < L-1$ and threshold input image into two classes c_1 (intensity values in $[0, k]$) and c_2 (intensity values in $[k+1, L-1]$)
  - Probability $P_1(k)$ that pixel is assigned to c_1 is given by

$$P_1(k) = \sum_{i=0}^{k} p_i$$

    - If $k = 0$, the probability of any pixel assigned to class c_1 is 0
  - Probability of class c_2

$$P_2(k) = \sum_{i=k+1}^{L-1} p_i = 1 - P_1(k)$$

  - Mean intensity of pixels in c_1

$$\begin{aligned} m_1(k) &= \sum_{i=0}^{k} iP(i|c_1) \\ &= \sum_{i=0}^{k} \frac{iP(c_1|i)P(i)}{P(c_1)} \\ &= \frac{1}{P_1(k)} \sum_{i=0}^{k} ip_i \qquad \text{Since } i \text{ comes from class } c_1 \end{aligned}$$

  - Bayes' formula

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

  - Similarly, the mean intensity value of pixels assigned to class c_2 is

$$\begin{aligned} m_2(k) &= \sum_{i=k+1}^{L-1} iP(i|c_2) \\ &= \frac{1}{P_2(k)} \sum_{i=k+1}^{L-1} ip_i \end{aligned}$$

* Cumulative mean or average intensity up to level k is

$$m(k) = \sum_{i=0}^{k} ip_i$$

* Global mean or average intensity of the entire image is

$$m_G = \sum_{i=0}^{L-1} ip_i$$

* Also note that

$$\begin{aligned} P_1 m_1 + P_2 m_2 &= m_G \\ P_1 + P_2 &= 1 \end{aligned}$$

* Effectiveness of threshold at level k is evaluated by normalized, dimensionless measure

$$\eta = \frac{\sigma_B^2}{\sigma_G^2}$$

where σ_G^2 is global variance of all pixels in the image

$$\eta_G^2 = \sum_{i=0}^{L-1} (i - m_G)^2 p_i$$

and σ_B^2 is the between-class variance

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

* This expression can be written as

$$\begin{aligned} \sigma_B^2 &= P_1 P_2 (m_1 - m_2)^2 \\ &= \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)} \end{aligned}$$

  - Only two parameters m_1 and P_1 need to be computed for any value of k
  - Farther the two means m_1 and m_2 from each other, larger the σ_B^2, implying that between-class variance is a measure of separability between classes
  - σ_G^2 is constant implying that the effectiveness of threshold η is also a measure of separability, and maximizing η is same as maximizing σ_B^2
  - Now, objective is to find threshold k that maximizes σ_B^2
  - Computation of η implicitly assumes that $\sigma_G^2 > 0$
  - The variance can be zero only if all intensity levels in image are the same, or only one class of pixels in the image
  - $\eta = 0$ for a constant intensity image; separability of a single class from itself is zero

* Final results

$$\begin{aligned} \eta(k) &= \frac{\sigma_B^2(k)}{\sigma_G^2} \\ \sigma_B^2(k) &= \frac{[m_G P_1(k) - m(k)]^2}{P_1(k)[1 - P_1(k)]} \end{aligned}$$

* Optimum threshold is given by k^* that maximizes $\sigma_B^2(k)$

$$\sigma_B^2(k^*) = \max_{0 \le k \le L-1} \sigma_B^2(k)$$

    - k^* is computed by evaluating the above expression for all integer values of k
    - The input image can now be segmented using k^* as threshold
    - Summary of Otsu's method
        1. Compute the normalized histogram of input image p_i, $i = 0, 1, \ldots, L-1$
        2. Compute cumulative sums $P_1(k)$, for $k = 0, 1, \ldots, L-1$
        3. Compute cumulative means m_k, for $k = 0, 1, \ldots, L-1$
        4. Compute global mean m_G
        5. Compute between-class variance term $\sigma_B^2(k)$ for $k = 0, 1, \ldots, L-1$
        6. Obtain Otsu threshold k^* as the value of k for which $\sigma_B^2(k)$ is maximum
        7. Compute global variance σ_G^2 and obtain the separability measure η^*
  - Example 10.14: Optimum global thresholding using Otsu's method
    - Figure 10.36

- Image smoothing to improve global thresholding
  - Figure 10.37
  - Figure 10.38
    - A small foreground with respect to the background
    - Additive Gaussian noise with zero mean and standard deviation of 10 intensity levels
    - Segmentation failed even after smoothing
    - Foreground region is so small that its contribution to histogram is insignificant compared to intensity spread due to noise

- Edges to improve global thresholding
  - Chances of finding a good threshold are enhanced consideraly if the histogram peaks are tall, narrow, symmetric, and separated by deep valleys
  - Improve the shape of histogram by considering only the pixels on or near the edges between objects and background
  - Using pixels on or near the edges between objects and background to compute histogram gives peaks with approximately the same height and separated by a valley
  - Above approach assumes that the edges between objects and background are known
  - The fact that a pixel is on a boundary can be indicated by computing its gradient or Laplacian
  - Example 10.15: Using edge information based on the gradient to improve global thresholding

- Variable thresholding
  - Image partitioning
    - Subdivide the image into nonoverlapping rectangles
      - Compensates for non-uniformities in illumination and/or reflectance
      - Rectangles should be small enough so that illumination of each is approximately uniform
    - Subdivision works as long as objects of interest and background are of reasonably comparable area/size
      - If size is not similar, the method fails because the subdivision may contain only the foreground or background
  - Variable thresholding based on local image properties
    - Based on specific properties computed in a neighborhood of every point
    - Mean m_{xy} and standard deviation σ_{xy} of pixels in the neighborhood S_{xy} of every point (x, y) in the image
      - Descriptors of average intensity and local contrast

* Common forms of variable, local thresholds

$$\begin{aligned} T_{xy} &= a\sigma_{xy} + bm_{xy} \\ T_{xy} &= a\sigma_{xy} + bm_G \end{aligned}$$

where a and b are nonnegative constants and m_G is the global image mean

* Segmented image is computed as

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{otherwise} \end{cases}$$

* Can also use a predicate based on parameters in the pixel neighborhood

$$g(x,y) = \begin{cases} 1 & \text{if } Q\text{(local parameters) is true} \\ 0 & \text{otherwise} \end{cases}$$

* Consider the following predicate $Q(\sigma_{xy}, m_{xy})$ based on local mean and standard deviation

$$Q(\sigma_{xy}, m_{xy}) = \begin{cases} \text{true} & \text{if } f(x,y) > a\sigma_{xy} \ \ \&\& \ \ f(x,y) > bm_{xy} \\ \text{false} & \text{otherwise} \end{cases}$$

* Example 10.18: Variable thresholding based on local image properties
  - Figure 10.43: Yeast image
  - Choosing global mean gives better result when the background is nearly constant and all object intensities are above or below the background intensity
  - Choose $a = 30$ and $b = 1.5$, experimentally determined

– Variable thresholding based on moving averages

* Based on computing moving average along scan lines of an image
* Useful in document processing applications
* Scanning line-by-line in a zigzag pattern to reduce illumination bias
* Let z_{k+1} be the intensity of a point encountered in the scanning sequence at step $k+1$
* The moving average (mean intensity) at new point is given by

$$\begin{aligned} m(k+1) &= \frac{1}{n}\sum_{i=k+2-n}^{k+1} z_i && \text{for } k \geq n-1 \\ &= m(k) + \frac{1}{n}(z_{k+1} - z_{k-n}) && \text{for } k \geq n+1 \end{aligned}$$

where n is the number of points used in computing the average, and $m(1) = z_1, k > 0$

* If m_{xy} is the moving average at point (x,y), the threshold at the point T_{xy} is given by cm_{xy} where c is a positive scalar
* Example 10.19: Document thresholding using moving averages
  - Figure 10.44
  - Intensity shading from spot illumination such as photo flash
  - Figure 10.45
  - Corrupted by a sinusoidal intensity variation

Segmentation by region growing and by region splitting and merging

- Region growing

– Group pixels or subregions into larger regions based on predefined criteria for growth

* Start with a set of *seed points* and grow regions by appending neighboring pixels to the seed with predefined properties (range of intensities or color)

    - Compute the same set of properties at every pixel to assign pixels to regions
    - If results show clusters of values, pixels close to the cluster centroids can be used as seeds
  - Selection of similarity criteria
    - Depends on type of available image data
    - Color in land-use satellite imagery
    - Intensity levels and spatial properties (moments/texture) for monochrome images
  - Use of connectivity to grow regions
  - Stopping rule
    - Stop region growth when no more pixels satisfy the criteria for inclusion in the region
    - Criteria such as intensity values, texture, and color are local and may not take into account the *history* of region of growth
    - Additional criteria may use the concept of size
      - Comparison of intensity of candidate pixel to the average intensity of the region
      - Shape of the region being grown
  - Basic region-growth algorithm based on 8-connectivity
    - $f(x, y)$ – Input image
    - $S(x, y)$ – Seed array containing 1's at the location of seed points and 0 elsewhere; same size as f
    - Q – Predicate to be applied at each location (x, y)

    1. Find all connected components in $S(x, y)$ and reduce connected component to one pixel; label all such pixels 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 pixels in f_Q that are 8-connected to that seed point
    4. Label each connected component in g with a different region label (integer or letter); this is the segmented image obtained by region growing

  - Example 10.20: Segmentation by region growing
    - Figure 10.46

- Region splitting and merging
  - Alternative to growing regions by seed points
  - Subdivide an image into a set of disjoint regions and then, merge and/or split the regions to satisfy the condition of segmentation
  - R – Entire image region
  - Q – Predicate
  - Subdivide R into smaller and smaller quadrants so that for any region R_j, $Q(R_j)$ = `true`
  - Splitting represented by a quadtree (Figure 10.47)
    - Root of quadtree corresponds to the entire image
  - If only splitting is used, final partition contains adjacent regions with identical properties
    - Remedied by merging as well as splitting
    - Merge adjacent regions whose combined pixels satisfy predicate Q
      - Merge two adjacent regions R_j and R_k if $Q(R_j \cup R_k) =$ `true`
  - Algorithm summary

```
algorithm split_merge ( R )
    for each region Ri in R
        if Q(Ri) is false
            split Ri into four quadrants { Ri1, Ri2, Ri3, Ri4 }
            for each quadrant Rj in { Ri1, Ri2, Ri3, Ri4 }
                split_merge ( Rj )
        for each adjacent Ri and Rj in R
            if Q( R_i union Rj ) == true
                merge ( Ri, Rj )
```

- Example 10.21: Segmenting by region splitting and merging
  - Figure 10.48
  - Extracting ring of less dense matter surrounding the dense inner region
  - Predicate

$$Q(R) = \begin{cases} \texttt{true} & \text{if } \sigma_R > a \ \ \texttt{\&\&} \ \ 0 < m_R < b \\ \texttt{false} & \text{otherwise} \end{cases}$$

Region segmentation using clustering and superpixels

- k-means clustering
  - Partition a set of observations Q into k clusters
  - Each observation assigned to the cluster with the nearest mean
  - Each mean is called the *prototype* of its cluster
- A set of vector observations $\{\mathbf{z_1}, \mathbf{z_2}, \ldots, \mathbf{z_Q}\}$
  - Each vector is of the form $\mathbf{z} = [z_1, z_2, \ldots, z_n]^T$
  - Each component of vector $\mathbf{z}$ represents a numerical pixel attribute: grayscale value or a 3D vector for RGB images
  - Objective of k-means clustering is to partition the set Q into k ($k \leq Q$) disjoint cluster sets $C = \{C_1, C - 2, \ldots, C_k\}$ so that the following criterion for optimality is satisfied

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

  $\mathbf{m}_i$ is the mean vector or centroid of cluster C_i
    - Find the sets $C = \{C_1, C_2, \ldots, C_k\}$ such that the sum of distances from each point in a set to the entroid of that set is minimized
    - NP-hard problem with no practical solution
  - Algorithm for k-means
    1. Initialize: Specify an initial set of means $\mathbf{m}_i(1), i = 1, 2, \ldots, k$
    2. Assign samples to clusters: Each sample assigned to cluster with closest centroid

$$\mathbf{z}_q \rightarrow C_i \text{ if } ||\mathbf{z}_q - \mathbf{m}_i||^2 < ||\mathbf{z}_q - \mathbf{m}_j||^2 \qquad j = 1, 2, \ldots, k(j \neq i); q = 1, 2, \ldots, Q$$

    3. Update cluster centroid

$$\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \qquad i = 1, 2, \ldots, k$$

    4. Test for completion
       - Compute Euclidean norms of differences between the centroid in the current and previous steps
       - Compute residual error E as the sum of k norms

    - If $E \leq T$ stop else go to step 2 (T is a prespecified nonnegative threshold)
  - Algorithm converges in a finite number of iterations when $T = 0$
  - Not guaranteed to yield global minima
  - Result depends on initial values chosen for $\mathbf{m}_i$
  - Choice of k gives you the number of segmented regions
  - Example 10.22: Using k-means clustering for segmentation
    - Figure 10.49

- Region segmentation using superpixels
  - Replace pixel grid by grouping pixels into primitive regions that are perceptually meaningful compared to individual pixels
    - Improves the performance of segmentation algorithms by reducing irrelevant details
    - Figure 10.50
    - May lose fine image detail compared to individual pixels
  - Adherence to boundaries
    - Boundaries between ROIs must be preserved in a superpixel image
    - Topological properties must be preserved with good computational efficiency
    - Figure 10.51
    - Figure 10.52
      - Decreasing the number of superpixels
      - Preserves boundaries between principal regions and basic topology
  - SLIC superpixel algorithm
    - SLIC – Simple Linear Iterative Clustering
    - Modification of the k-means algorithms
    - Observations typically use 5D vectors for three color components and two spatial coordinates

$$\mathbf{z} = [r\ g\ b\ x\ y]^T$$

    - Let n_{sp} be the desired number of superpixels
    - Let n_{tp} be the total number of pixels
    - Obtain initial superpixel centers $\mathbf{m}_i = [r_i\ g_i\ b_i\ x_i\ y_i]^T$ by sampling the image on a regular grid spaced s units apart where

$$s = \sqrt{n_{\text{tp}}/n_{\text{sp}}}$$

      - Helps with generating superpixels appromixately equal in size/area
    - Initial cluster centers are moved to the lowest gradient position in the 3×3 neighborhood about each center
      - Prevents centering a superpixel on image edge
    - Algorithm steps
      1. Initialization: Compute initial superpixel cluster centers

$$\mathbf{m}_i = [r_i\ g_i\ b_i\ x_i\ y_i]^T \qquad i = 1, 2, \ldots, n_{\text{sp}}$$

      2. Assign samples to cluster centers: For each cluster center $\mathbf{m}_i, i = 1, 2, \ldots, n_{\text{sp}}$, compute the distance $D_i(p)$ between $\mathbf{m}_i$ and each pixel p in a $2s \times 2s$ neighborhood about $\mathbf{m}_i$
         - For each p and $i = 1, 2, \ldots, n_{\text{sp}}$, if $D_i < d(p)$, let $d(p) = D_i$ and label $L(p) = i$
      3. Update the cluster centers: Let C_i denote the set of pixels with label $L(p) = i$; Update $\mathbf{m}_i$

$$\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \qquad i = 1, 2, \ldots, n_{\text{sp}}$$

4. Test for convergence
   - Compute Euclidean norm of the difference between the mean vectors in the current and previous steps
   - Compute residual error E as the sum of n_{sp} norms
   - If $E < T$ go to step 5; else go to step 2 (T is a prespecified nonnegative threshold)
5. Post-process the superpixel regions: Replace all superpixels in each region C_i by their average value $\mathbf{m}_i$

* Difference from k-means
  - Distances D_i are not specified as Euclidean distances
  - Distances are computed for regions of size $2s \times 2s$ rather than for all the pixels in the image, reducing computation time significantly

– Specifying the distance measure
  * SLIC superpixels described as colors and spatial variables
  * Not recommended to use a single Euclidean distance
    - Scales in the axes of coordinate system are different and unrelated (color and spatial components)
    - Normalize the distance of various components and combine them into a single measure
    - Distance between color and spatial components denoted by d_c and d_s

$$\begin{aligned} d_c &= \sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2} \\ d_s &= \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \end{aligned}$$

    - Composite distance D is given by

$$D = \left[\left(\frac{d_c}{d_{cm}} \right)^2 + \left(\frac{d_s}{d_{sm}} \right)^2 \right]^{1/2}$$

      where d_{cm} and d_{sm} are maximum expected value of d_c and d_s; maximum spatial distance should correspond to sampling interval, that is

$$d_{sm} = s = \sqrt{n_{\text{tp}}/n_{\text{sp}}}$$

    - Not straightforward to compute this distance; may vary significantly from cluster to cluster and from image to image
    - May set d_{cm} to a constant c such that

$$D = \left[\left(\frac{d_c}{c} \right)^2 + \left(\frac{d_s}{s} \right)^2 \right]^{1/2}$$

    - Rewrite the equation

$$D = \left[d_c^2 + \left(\frac{d_s}{s} \right)^2 c^2 \right]^{1/2}$$

    - c used to weigh the relative improtance between color similarity and spatial proximity
    - Large c implies spatial proximity is more important with compact superpixels
    - Small c implies superpixels adhere more tightly to image boundaries, with less regular size and shape
  * Example 10.23: Using superpixels for image segmentation
    - Figure 10.53

Region segmentation using graph cuts

- Consider pixels as nodes of graphs and find an optimum partition (cut) into group of nodes

- Images as graphs
  - A graph is defined as
    $$G = (V, E)$$
    where V is a set and
    $$E \subseteq V \times V$$
  - If $(u, v) \in E \Rightarrow (v, u) \in E$, the graph is undirected; otherwise it is directed
  - Consider undirected graphs with weighted edges characterized by a matrix $\mathbf{W}$ whose elements $w(i, j)$ indicate the weight of the edge connecting nodes i and j
    - $\mathbf{W}$ is symmetric since $w(i, j) = w(j, i)$
    - Graph with weights associated with edges is called weighted graph
  - Represent an image to be segmented as undirected weighted graph with nodes as pixels and an edge between every pair of nodes
    - $w(i, j)$ indicates similarity between i and j
    - Partition the graph into disjoint subsets $V_1, V_2, \ldots, V_k$ such that the similarity between nodes within a subset is high compared to similarity of nodes across the subsets
    - Nodes of partitioned subsets form the regions
  - Set V is partitioned into subsets by cutting the graph
    - Cut defined as partition of V into two subsets A and B such that
      $$A \cup B = V \text{ and } A \cap B = \emptyset$$
    - Cut implemented by removing the edges connecting subgraphs A and B
  - Key elements of using graph cuts for image segmentation
    1. Associate a graph with an image
    2. Cut the graph to partition image into background and foreground
  - Figure 10.55
    - Edges only between adjacent 4-connected pixels
      - In general, there should be edges between every pair of pixels
    - Weight of edge a function of spatial distance and intensity measure
      - Similarity between two pixels as inverse of difference in their intensity
      - For two nodes n_i and n_j
        $$w(i, j) = \frac{1}{|I(n_i) - I(n_j)| + c}$$
        where $I(n_i)$ and $I(n_j)$ are the intensities of two pixels and c is a constant added to prevent division by zero
    - Strong vs weak edges
      - Image segmented by cutting the graph along weak edges
  - Figure 10.56
    - Source and sink terminal nodes connected to all nodes in the graph via unidirectional links called *t-links*
    - Terminal nodes associate a probability to each pixel that it is a foreground or background pixel
      - Attached as weights to the t-links
- Minimum graph cuts
  - Interpret the graph as a flow network and obtain *minimum graph cut*
  - Max-Flow, Min-Cut theorem
    - In a flow network, the maximum amount of flow passing from source to sink is equal to the minimum cut

  - Minimum cut is the smallest total weight of the edges that, if removed, would disconnect sink from source

$$\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

  - Optimum partition of a graph minimizes this cut value
- May lead to groupings that favor cutting small sets of isolated nodes in a graph, leading to improper segmentations
  - Figure 10.57
    - Two ROIs characterized by the tightness of pixel groupings
    - Meaningful edge weights inversely proportional to the distance between pairs of points
    - Leads to weights that would be smaller for isolated points showing the min cut in the figure; must be avoided
  - Work with a measure of *disassociation* that computes the cost as a fraction of total edge connections to all nodes in the graph, leading to *normalized cut* or Ncut

$$\text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}$$

$$\text{assoc}(A, V) = \sum_{u \in A, z \in V} w(u, z)$$

$$\text{assoc}(B, V) = \sum_{v \in A, z \in V} w(v, z)$$

    assoc is the sum of weights of all edges from the nodes of subgraph to the nodes of the entire graph
  - Replacing $\text{cut}(A, B)$ with $\text{Ncut}(A, B)$ removes the small values for cut that partitions isolated points
  - Define a measure for total *normalized association* within graph partitions as

$$\text{Nassoc}(A, B) = \frac{\text{assoc}(A, A)}{\text{assoc}(A, V)} + \frac{\text{assoc}(B, B)}{\text{assoc}(B, V)}$$

    where $\text{assoc}(A, A)$ gives the total weights connecting the nodes within A

$$\text{Ncut}(A, B) = 2 - \text{Nassoc}(A, B)$$

    implying that minimizing $\text{Ncut}(A, B)$ simultaneously maximizes $\text{Nassoc}(A, B)$
  - Segmentation based on finding a partition that minimizes $\text{Ncut}(A, B)$
    - NP-complete task
    - Approximate discrete solution by formulating minimization as a generalized eigenvalue problem

- Computing minimal graph cuts
  - Let K be the number of nodes in V
  - Define a K-dimensional indicator vector $\mathbf{x}$ such that

$$x_i = \begin{cases} 1 & \text{if node } n_i \in A \\ -1 & \text{if node } n_i \in B \end{cases}$$

  - Let the sum of weights from node n_i to all other nodes in V be given by

$$d_i = \sum_j w(i, j)$$

  - Then, we define

$$\begin{aligned} \text{Ncut}(A, B) &= \frac{\text{cut}(A, B)}{\text{cut}(A, V)} + \frac{\text{cut}(A, B)}{\text{cut}(B, V)} \\ &= \frac{\sum_{x_i > 0, x_j < 0} -w(i, j) x_i x_j}{\sum_{x_i > 0} d_i} + \frac{\sum_{x_i < 0, x_j > 0} -w(i, j) x_i x_j}{\sum_{x_i < 0} d_i} \end{aligned}$$

- Objective is to find a vector $\mathbf{x}$ that minimizes Ncut(A, B)
- Solution found by solving the generalized eigen-system expression

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$

  where $\mathbf{D}$ is a $K \times K$ diagonal matrix with main-diagonal elements $d_i, i = 1, 2, \ldots, K$ and $\mathbf{W}$ is a $K \times K$ weight matrix with elements $w(i, j)$
- Solving this equation gives K eigenvalues and K eigenvectors, each corresponding to one eigenvalue; solution is the eigenvector corresponding to the *second smallest* eigenvalue
- Write the above equation as

$$\mathbf{A}\mathbf{z} = \lambda \mathbf{z}$$

  where

$$\mathbf{A} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-\frac{1}{2}}$$

  and

$$\mathbf{z} = \mathbf{D}^{\frac{1}{2}}\mathbf{y} \Rightarrow \mathbf{y} = \mathbf{D}^{-\frac{1}{2}}\mathbf{z}$$