Image Processing
Segmentation
Fundamental steps in problem solving using digital image analysis

1. Problem
2. Image Acquisition
3. Preprocessing
4. Segmentation
5. Representation and Description
6. Classification, Recognition, interpretation
7. Solution
Image Segmentation

There are many definitions:

- Segmentation **subdivides** an image into its constituent regions and/or objects (Gonzales, pp567)
- Segmentation is a process of **grouping** together pixels that have similar attributes (Efford, pp250)
- Image Segmentation is the process of **partitioning** an image into non-intersecting regions such that each region is homogeneous and the union of no two adjacent regions is homogeneous (Pal, pp1277)
What is Image Segmentation?

- To be able to extract information from an image it is common to subdivide it into background and objects. This is called segmentation.

- Segmentation:
  - Split or separate an image into regions
  - To facilitate recognition, understanding, and region of interests (ROI) processing
What is Image Segmentation?

Divide the image contents into its constituent regions or objects

- **Full segmentation**: Individual objects are separated from the background and given individual ID numbers (labels).
- **Partial segmentation**: The amount of data is reduced (usually by separating objects from the background) to speed up further processing.

Segmentation ...

... is often the most difficult problem to solve in image analysis; there is **no universal solution**!

- The problem can be made much easier if solved in cooperation with the constructor of the imaging system (choice of sensors, illumination, background, etc.)
Region definition

Divide the image contents into its constituent regions

- Define $P$ as a function operating on a region. For a pixel $x$, $P(x) =$ true if $x$ satisfies a specific property.
- After applying the function, the image becomes a binary image. Using pixel connectivity definitions, a region can be defined.
Object definition

An object is a set of pixels that have similar relations

- A certain intensity
- Situated in an area delimited by a discontinuity of intensity
- An intensity that has a statistical relation to it’s surrounding
Basic Principle - Segmentation

- Segmentation algorithms are often based on one of the following two basic properties of intensity values:
  - Similarity
    - Partitioning an image into regions that are similar according to a set of predefined criteria.
  - Discontinuity
    - Detecting boundaries of regions based on local discontinuity in intensity.
Similarity vs Discontinuity (1)

Vertebra (CT)  
Finding a region (complete vertebra)  
Finding a boundary (complete vertebra)
Similarity vs Discontinuity (2)

Features
• intensity
• texture
• edge sharpness
• any other relevant feature(s)

FIGURE 10.1 (a) Image containing a region of constant intensity. (b) Image showing the boundary of the inner region, obtained from intensity discontinuities. (c) Result of segmenting the image into two regions. (d) Image containing a textured region. (e) Result of edge computations. Note the large number of small edges that are connected to the original boundary, making it difficult to find a unique boundary using only edge information. (f) Result of segmentation based on region properties.
Human visual system – an excellent segmentation system
We are very insensitive to all types of distortion!

We can even “see” what is not there!
Image Segmentation Methods

- **Edge based segmentation** [Discontinuity]
  - Finding boundary between adjacent regions
    - i.e. Detecting edges that separate regions from each other.

- **Threshold based segmentation** [Similarity]
  - Finding regions by grouping pixels with similar intensities
    - i.e. Based on pixel intensities (shape of histogram is often used for automation).

- **Region based segmentation** [Similarity]
  - Finding regions directly using growing or splitting
    - i.e. Grouping similar pixels (with e.g. region growing or merge & split).

- **Motion (Match) based segmentation** [Similarity]
  - Finding regions by comparing successive frames of a video sequence to identify regions that correspond to moving objects
    - i.e. Comparison to a given template.
Segmentation Using Discontinuity
Point, Line & Edge Detection
using Image Intensity Derivatives
Discontinuities

- **Point**
  An area with high frequency information in both $x$ and $y$ directions

- **Line**
  An area with high frequency information in one direction and low frequency in the other. Intensity is the same on both sides of the line

- **Edge**
  Separates two areas with different intensities
Point, line & edge detection

- Basic discontinuities in images
- Detection from the 1st and 2nd order derivatives of the intensity profile
- Sensitive to image noise
  -> noise filtering needed

**FIGURE 10.2** (a) Image. (b) Horizontal intensity profile through the center of the image, including the isolated noise point. (c) Simplified profile (the points are joined by dashes for clarity). The image strip corresponds to the intensity profile, and the numbers in the boxes are the intensity values of the dots shown in the profile. The derivatives were obtained using Eqs. (10.2-1) and (10.2-2).
Background

Derivatives of intensity profile $f(x)$

- First-order derivative

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

→ spatial filter with $[-1 \ 1]$ kernel

- Second-order derivative

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

→ filter with kernel $[1 \ 2 \ 1]$

Derivatives can be approximated by simple spatial filters!
Characteristics of First and Second Order Derivatives

- **1st order derivatives** generally produce **thicker edges** in image.
- **2nd-order derivatives** have a stronger response to **fine detail**, such as thin lines, isolated points, and noise.
- Second-order derivatives produce a double-edge response at ramp and step transition in intensity.
- The **sign of the second derivative** can be used to determine whether a transition into an edge is from light to dark or dark to light.
Detection of Isolated Points

- The Laplacian
  \[ \nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2} \]
  \[ = f(x + 1, y) + f(x - 1, y) + f(x, y + 1) + f(x, y - 1) - 4f(x, y) \]

- Filter with 3x3 kernel (mask)
  \[
  \begin{bmatrix}
    0 & 1 & 0 \\
    1 & -4 & 1 \\
    0 & 1 & 0 
  \end{bmatrix}
  \]

- Filtering using 3x3 mask
  \[ R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k \]

where \( z \) is the pixel intensity at the location of weight \( w \).
Detection of Isolated Points

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>8</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
</tr>
</tbody>
</table>

**FIGURE**

(a) Point detection mask.
(b) X-ray image of a turbine blade with a porosity.
(c) Result of point detection.
(d) Result of using Eq. (10.1-2).
(Original image courtesy of X-TEK Systems Ltd.)
Point detection in MATLAB

\[ g = \text{abs}(\text{imfilter}(\text{double}(f), w)) \geq T; \]

- where
  - \( f \) the input image
  - \( g \) the output image
  - \( w \) the filter kernel
  - \( T \) a given (known) threshold

**What if the threshold \( T \) is unknown?**

\[ w = [-1 -1 -1; -1 8 -1; -1 -1 -1]; \]
\[ g = \text{abs}(\text{imfilter}(\text{double}(f), w)); \]
\[ T = \max(g(:)); \]
\[ g = g \geq T; \]

- i.e. \( T \) is derived from the image data itself!
Line detection with the Laplacian

**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 …

- Detection of lines at specific angles
  - Tune the mask to the desired angle

This mask is isotropic, i.e. its response is independent of direction.

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & -8 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc}
-1 & -1 & -1 \\
2 & 2 & 2 \\
-1 & -1 & -1 \\
\end{array}
\quad
\begin{array}{ccc}
2 & -1 & -1 \\
-1 & 2 & -1 \\
-1 & -1 & 2 \\
\end{array}
\quad
\begin{array}{ccc}
-1 & 2 & -1 \\
-1 & 2 & -1 \\
-1 & 2 & -1 \\
\end{array}
\quad
\begin{array}{ccc}
-1 & -1 & 2 \\
-1 & 2 & -1 \\
2 & -1 & -1 \\
\end{array}
\]

Horizontal \quad +45^\circ \quad Vertical \quad -45^\circ

**Figure 10.6** Line detection masks. Angles are with respect to the axis system in Fig. 2.18(b).
FIGURE 10.7
(a) Image of a wire-bond template. (b) Result of processing with the $-45^\circ$ line detector mask 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 $g \geq T$.
where $g$ is the image in (e). (The points in (f) were enlarged to make them easier to see.)
What is an edge?

- Edges are places in an image that corresponds to object boundaries.

- Edges are pixels where image brightness changes abruptly.

- An edge is a property attached to an individual pixel and is calculated from the image function behavior in a neighborhood of the pixel.

- It is a vector variable (magnitude of the gradient, direction of an edge).

Brightness vs. Spatial Coordinates
Type of edges

- Physical Edges
  - Different objects in physical contact
  - Spatial change in material properties
  - Abrupt change in surface orientation

- Image Edges
  - In general: Boundary between contrasting regions in image
  - Specifically: Abrupt local change in brightness
Edge Model

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

**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 indicated by the short line segments shown in 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 Detection in 1-D

- Edges can be characterized as either:
  - local extrema of $f'(x)$
  - zero-crossings of $f''(x)$
Edges and derivatives

The magnitude of the first derivative can be used for edge detection.

The zero crossing indicates the middle of the edge.
Noise

- Noise is always a factor in images
- Derivative operators are high-pass filters
- High-pass filters boost noise
- Noise create false edges

Key concept:
- Build filters to respond to edges and suppress noise
Edges and Derivatives

- However …
  The first derivative is rather sensitive and the second derivative is very sensitive to noise!

- Therefore …
  - Image smoothing for noise reduction
  - Detection of individual edge points
  - Edge localization, i.e. combination of individual points into edge(s)

These are fundamental steps in edge detection.

**FIGURE 10.11** First column: Images and intensity profiles of a ramp edge corrupted by random 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.
Edge Detection

- Edge information in an image is found by looking at the relationship a pixel has with its neighborhoods.
- If a pixel’s gray-level value is similar to those around it, there is probably not an edge at that point.
- If a pixel’s has neighbors with widely varying gray levels, it may present an edge point.
Edge Detection Methods

- Many are implemented with convolution masks and based on discrete approximations to differential operators.
- Differential operations measure the rate of change in the image brightness function.
- Some operators return orientation information. Other only return information about the existence of an edge at each point.
Edge Detection

- Finding the edge strength and direction

Basic Edge Detection by Using First-Order Derivative

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

Gradient magnitude (strength): \[ M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \]

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

The direction of the edge: \[ \phi = \alpha - 90^\circ \]

**FIGURE 10.12** Using the gradient to determine edge strength and direction at a point. Note that the edge is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square in the figure represents one pixel.
Edge Detection

Finding the edge strength and direction

Approximations of the magnitude

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

\[ M(x, y) \approx g_x^2 + g_y^2 \]

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

Determination of the partial derivatives

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

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

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

**FIGURE 10.13**
One-dimensional masks used to implement Eqs. (10.2-12) and (10.2-13).
Directional Edge Detection

\[ \frac{\partial f(x, y)}{\partial x} \geq T \]  
Horizontal operator  
(finds vertical edges)

\[ \frac{\partial f(x, y)}{\partial y} \geq T \]  
Vertical operator  
(finds horizontal edges)

\[ \frac{\partial f(x, y)}{\partial x} \cos \theta + \frac{\partial f(x, y)}{\partial y} \sin \theta \geq T \]  
finds edges perpendicular to the \( \theta \) direction
Designing an edge detector

- Criteria for an “optimal” edge detector:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge

![Diagram showing edge detection metrics: True edge, Poor robustness to noise, Poor localization, Too many responses](image)
A $3 \times 3$ region of an image (the $z$’s are intensity values) and various masks used to compute the gradient at the point labeled $z_5$. 

**FIGURE 10.14**

A $3 \times 3$ region of an image (the $z$’s are intensity values) and various masks used to compute the gradient at the point labeled $z_5$. 
Prewitt and Sobel masks for detecting diagonal edges.

**FIGURE 10.15**
Sobel

**FIGURE 10.16**
(a) Original image of size $834 \times 1114$ 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 mask in Fig. 10.14(f) to filter the image. (c) $|g_y|$, obtained using the mask in Fig. 10.14(g). (d) The gradient image, $|g_x| - |g_y|$. 
The effect of filtering

**FIGURE 10.18**
Same sequence as in Fig. 10.16, but with the original image smoothed using a $5 \times 5$ averaging filter prior to edge detection.
Laplacian of Gaussian (LoG)

- Approximation of differential operator
- Tunable to the “scale” of the edge
- Based on 2D Gaussian function
- Sometimes called “Mexican hat operator”

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

\[
\nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} = \left[ \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right] e^{-\frac{x^2+y^2}{2\sigma^2}}
\]
Edge detection with the LoG

- Convolve image $f$ with LoG filter
  $$g(x, y) = \left[\nabla^2 G(x, y)\right] * f(x, y) = \nabla^2 \left[ G(x, y) * f(x, y) \right]$$
- Find zero crossings in the result $g$
- Equal to (Marr-Hildreth):
  1. Filter (convolve) $f$ with Gaussian low-pass filter $G$
  2. Compute the Laplacian $\nabla^2$ of the resulting image
  3. Find zero crossings in the image resulting from step 2
Edge detection with the LoG

**FIGURE 10.22**
(a) Original image of size $834 \times 1114$ pixels, with intensity values scaled to the range $[0, 1]$. (b) Results 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.
The Canny Edge Detector

- Optimal for step edges corrupted by white noise.
- The Objective
  - Low error rate
    The edges detected must be as close as possible to the true edge
  - Edge points should be well localized
    The edges located must be as close as possible to the true edges
  - Single edge point response
    The number of local maxima around the true edge should be minimum
The Canny Edge Detector: Algorithm

- Let $f(x, y)$ denote the input image and $G(x, y)$ denote the Gaussian function:
  \[ G(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \]

- We form a smoothed image, $f_s(x, y)$ by convolving $G$ and $f$.
  \[ f_s(x, y) = G(x, y) * f(x, y) \]

- Compute the gradient magnitude and direction (angle):
  \[ M(x, y) = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2} \]
  \[ \alpha(x, y) = \tan^{-1} \left[ \frac{g_y}{g_x} \right] \]

- The gradient $M(x, y)$ typically contains wide ridge around local maxima. Next step is to thin those ridges.
The Canny Edge Detector: Algorithm

- Nonmaxima suppression:

Let \( d_1, d_2, d_3, \) and \( d_4 \) denote the four basic edge directions for a 3 x 3 region:
  - horizontal,
  - -45 degree,
  - vertical,
  - +45 degree respectively.

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 \( d_k \), let \( g_N(x,y) = 0 \) (suppression); otherwise, let \( g_N(x,y) = M(x,y) \).
The Canny Edge Detector: Algorithm

(a) Two possible orientations of a horizontal edge (in gray) in a $3 \times 3$ neighborhood. (b) Range of values (in gray) of $\alpha$, 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 $3 \times 3$ neighborhood. Each edge direction has two ranges, shown in corresponding shades of gray.
The Canny Edge Detector: Algorithm

- The final operation is to threshold $g_N(x, y) = g_N(x, y)$ to reduce false edge points.
- Hysteresis thresholding:

  \[
  g_{NH}(x, y) = g_N(x, y) \geq T_H \\
  g_{NL}(x, y) = g_N(x, y) \geq T_L
  \]

And

\[
 g_{NL}(x, y) = g_{NL}(x, y) - g_{NH}(x, y)
\]
The Canny Edge Detector: Algorithm

- Depending on the value of $T_H$, the edges in $g_{NH}(x,y)$ typically have gaps. Longer edges are formed using the following procedure:

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

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

  (c). If all nonzero pixel in $g_{NH}(x,y)$ have been visited go to step (d), else return to (a).

  (d). Set to zero all pixels in $g_{NH}(x,y)$ that were not marked as valid edge pixels.
The Canny Edge Detection: Summary

- Smooth the input image with a Gaussian filter
- Compute the gradient magnitude and angle images
- Apply nonmaxima suppression to the gradient magnitude image
- Use double thresholding and connectivity analysis to detect and link edges
Edge detection with the LoG and Canny algorithm

*FIGURE 10.25*
(a) Original image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1].
(b) Thresholded gradient of 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.
Edge detection with the LoG and Canny algorithm

**FIGURE 10.26**
(a) Original head CT image of size $512 \times 512$ pixels, with intensity values scaled to the range $[0, 1]$.
(b) Thresholded gradient of 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.)

$T_L = 0.05; T_H = 0.15; \sigma = 2$ and a mask of size $13 \times 13$
Edge detection in MatLab

\[
\gg [g,t] = \text{edge}(f, \text{`method'}, \text{params});
\]

where

\( f \) the input image

\( \text{method} \) the edge detection method; \( \text{params} \) a set of parameters; \( t \) the threshold used by edge (optional)

Edge detection in MatLab – Sobel

\[
\gg [g,t] = \text{edge}(f, \text{`sobel'}, T, \text{dir});
\]

where

\( \text{`sobel'} \) the Sobel edge detection method; \( T \) a specified threshold; \( \text{dir} \) preferred edge detection (‘horizontal’, ‘vertical’, ‘both’)

<table>
<thead>
<tr>
<th>Edge Detector</th>
<th>Basic Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sobel</td>
<td>Finds edges using the Sobel approximation to the derivatives</td>
</tr>
<tr>
<td>Prewitt</td>
<td>Finds edges using the Prewitt approximation to the derivatives</td>
</tr>
<tr>
<td>Roberts</td>
<td>Finds edges using the Roberts approximation to the derivatives</td>
</tr>
<tr>
<td>Laplacian of a Gaussian (LoG)</td>
<td>Finds edges by looking for zero crossings after filtering ( f(x, y) ) with a Gaussian filter.</td>
</tr>
<tr>
<td>Zero crossings</td>
<td>Finds edges by looking for zero crossings after filtering ( f(x, y) ) with a user-specified filter.</td>
</tr>
<tr>
<td>Canny</td>
<td>Finds edges by looking for local maxima of the gradient of ( f(x, y) ). The gradient is calculated using the derivative of a Gaussian filter. The method uses two thresholds to detect strong and weak edges, and includes the weak edges in the output only if they are connected to strong edges. Therefore, this method is more likely to detect true weak edges.</td>
</tr>
</tbody>
</table>
Segmentation Using Discontinuity

Line Detection with the Hough Transform
The Hough Transform

- **Application**
  Detecting lines in an image (irrespective of their angles)

- **Basics**
  Point \((x_i, y_i)\) in the \((x,y)\) image plane
  Lines through \((x_i, y_i)\): \(y_i = ax_i + b\)
  Two points \((x_i, y_i)\) and \((x_j, y_j)\) on the same line:
  \(y_i = ax_i + b\) and \(y_j = ax_j + b\) -> same coefficients \((a,b)\)
The Hough Transform

- The Hough Transform from \((x, y)\) to \((a, b)\)

\[
y_i = ax_i + b \quad \Rightarrow \quad b = -x_i a + y_i
\]
\[
y_j = ax_j + b \quad \Rightarrow \quad b = -x_j a + y_j
\]

2 equations, two unknowns \(a, b\)

FIGURE 10.31
(a) \(xy\)-plane.
(b) Parameter space.

\((a', b')\) are the coefficients of the line through \((x_i, y_i)\) and \((x_j, y_j)\)
The Hough Transform

- Detecting lines with the Hough Transform
  - Convert the image to a binary image $B$ (e.g., thresholded edge image to enhance lines)
  - Transform all points $(x_i, y_i)$ to their corresponding lines in the Hough space
  - Detect crossings of lines
  - Transform the coordinates $(a_i, b_i)$ of these crossings to lines in the $(x_i, y_i)$ image domain
The Hough Transform

- What about vertical lines?

\[ y_i = y_j = b \]

i.e. \( a \) is undefined
The Hough Transform

- The Hough Transform – Another parametrization

\[ x \cos(\theta) + y \sin(\theta) = \rho \]

Description of line in \( xy \)-plane as \((\rho, \theta)\)

**Figure 10.32** (a) \((\rho, \theta)\) parameterization of line in the \(xy\)-plane. (b) Sinusoidal curves in the \(\rho\theta\)-plane; the point of intersection \((\rho', \theta')\) 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.
The Hough Transform

FIGURE 10.33
(a) Image of size 101 × 101 pixels, containing five points.
(b) Corresponding parameter space. (The points in (a) were enlarged to make them easier to see.)
The Hough Transform Example
The Hough Transform

**FIGURE 10.34** (a) A 502 × 564 aerial image of an airport. (b) Edge image 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.
Segmentation Using Similarity

Image Intensity Thresholding
Thresholding

- The derivative method will fail if the hills in the histogram are situated too closely.
- The possible reasons:
  - The image is very noisy.
  - The difference in intensity between object and background is low.
Global Thresholding

Derive binary image \( g(x, y) \) from image \( f(x, y) \) by

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

- Global thresholding in MATLAB
  \[
  \text{out} = \text{in} > T;
  \]
  - where
    \( \text{in} \) the input image; \( T \) a specified threshold; \( \text{out} \) the binary output image
The area of the segmented object is less sensitive to errors in threshold level if we chose $T$ to the value where the histogram has a local minimum.

The local min values can be found by setting the derivative of the histogram equal to zero.
Global Thresholding

- Performance and Multiple?
- Works well for
  - bi-modal histograms
  - images with low noise level
  - images without intensity inhomogeneity (transients)
- May need morphological post-processing
  - to remove small objects
  - to close holes

Multiple (global) thresholding
Can be used to select pixels within a certain intensity range

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

- The influence of noise

**FIGURE 10.36** (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)–(f) Corresponding histograms.
Global Thresholding

- The influence of intensity transients

**FIGURE 10.37** (a) Noisy image. (b) Intensity ramp in the range [0.2, 0.6]. (c) Product of (a) and (b). (d)–(f) Corresponding histograms.
Adaptive Thresholding

- Images having uneven illumination makes it difficult to segment using histogram
- This approach is to divide the original image into sub images and use the thresholding process to each subimage
Adaptive Global Thresholding

1. Select initial global threshold $T$ (e.g. mean of all pixels)
2. Segment image into two groups
   - All values > $T$: group $G_1$ with mean $m_1$
   - All values ≤ $T$: group $G_2$ with mean $m_2$
3. Compute new threshold $T = \frac{1}{2} (m_1 + m_2)$
4. Repeat steps 2 and 3 until $T$ changes less than predefined threshold $T$ (or % of $T$)

**Figure 10.38** (a) Noisy fingerprint. (b) Histogram. (c) Segmented result using a global threshold (the border was added for clarity). (Original courtesy of the National Institute of Standards and Technology.)
Adaptive Global Thresholding

● Method 1 in MatLab

```matlab
>> T = 0.5*(double(min(f(:)))+double(max(f(:))));
>> done = false;
>> while ~done
    g = f >= T;
    Tnext = 0.5*(mean(f(g)) + mean(f(~g)));
    done = abs(T - Tnext) < 0.5;
    T = Tnext;
end

● $T$ the resulting threshold
Adaptive Global Thresholding, Otsu

Image with \( n \) pixels and \( L \) intensity levels \( q = 0, 1, 2, \ldots, L - 1 \)

Normalized histogram is: \( p_q = \frac{n_q}{n} \), \( q = 0, 1, 2, \ldots, L - 1 \)

Threshold \( k \) that divides image in two sets (classes):
- \( C_0 \): pixels with levels in \([0, 1, \ldots, k - 1]\)
- \( C_1 \): pixels with levels in \([k, k + 1, \ldots, L - 1]\)

Choose \( k \) that maximizes the between-class variance

\[
\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2
\]
Maximize \[ \sigma_B^2 = \omega_0 \left( \mu_0 - \mu_T \right)^2 + \omega_1 \left( \mu_1 - \mu_T \right)^2 \]

where

\[ \omega_0 = \sum_{q=0}^{k-1} p_q \] is probability of \( C_0 \)

\[ \omega_1 = \sum_{q=k}^{L-1} p_q \] is probability of \( C_1 \)

\[ \mu_0 = \frac{1}{\omega_0} \sum_{q=0}^{k-1} q p_q \] mean of pixels in \( C_0 \)

\[ \mu_1 = \frac{1}{\omega_1} \sum_{q=k}^{L-1} q p_q \] mean of pixels in \( C_1 \)

\[ \mu_T = \sum_{q=0}^{L-1} q p_q \] mean of all pixels in image
Adaptive Global Thresholding, Otsu

- Maximize

$$\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2$$

- Probability of $C_0$
- Probability of $C_1$
- Variance in $C_0$
- Variance in $C_1$

Histogram
Adaptive Global Thresholding, Otsu

**FIGURE 10.39**
(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 10.3.2.
(d) Result obtained using Otsu’s method.
(Original image courtesy of Professor Daniel A. Hammer, the University of Pennsylvania.)
Adaptive Global Thresholding, Otsu

- Smoothing may be required

**FIGURE 10.40** (a) Noisy image from Fig. 10.36 and (b) its histogram. (c) Result obtained using Otsu’s method. (d) Noisy image smoothed using a $5 \times 5$ averaging mask and (e) its histogram. (f) Result of thresholding using Otsu’s method.
but doesn’t help !!!
Adaptive Global Thresholding, Otsu

- Multiple global thresholds

**FIGURE 10.45** (a) Image of iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)
Otsu’s thresholding method in MatLab

```matlab
>> T = graythresh(in);
```

where

- `in` the input image
- `T` the Otsu threshold (number in range $[0,1]$)

Note: $T$ has to be scaled to the proper range, e.g. $[0,255]$ for `uint8`
Local Thresholding

- Local thresholding (by image partitioning)

**FIGURE 10.46** (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu’s method. (e) Image subdivided into six subimages. (f) Result of applying Otsu’s method to each subimage individually.
Segmentation Using Similarity
Region Growing, Region Splitting & Merging
Region Growing

Region-oriented segmentation

Segment the image in different regions $R_i$

- The regions cover the whole image
- Two regions do not have the same elements
- A region fulfills some property $P$
- The union of two regions does not satisfy $P$
Region Growing

1. Define seed point
2. Add n-neighbors to list L
3. Get and remove top of L
4. Test n-neighbors p
   if p not treated:
   if $P(p, R) = \text{True}$
   then $p \rightarrow L$ and add p to region
   else p marked boundary
5. Go to 2 until L is empty

- Two Regions $R$ and $\neg R$
Region Growing

**FIGURE 10.51** (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 (a) and (c). (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 & Merging

1. Split any region $R_i$ into 4 quadrants if $Q(R_i) = \text{FALSE}$ until no further splitting is possible

2. Merge any adjacent $R_i$ and $R_j$ for which $Q(R_i \cup R_j) = \text{TRUE}$ until no further merging is possible

where $R$ is entire image, $R_i$ a region in image, $Q$ a predicate
Region Splitting & Merging

**Figure 10.52**
(a) Partitioned image.
(b) Corresponding quadtree. $R$ represents the entire image region.
Region Splitting & Merging

\[ Q = \begin{cases} 
\text{TRUE} & \text{if } \sigma > a \ \text{AND} \ 0 < m < b \\
\text{FALSE} & \text{otherwise}
\end{cases} \]

\( \sigma = \text{standard deviation} \)

\( m = \text{mean} \)

in a region

FIGURE 10.53
(a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope.
(b)-(d) Results of limiting the smallest allowed quadregion to sizes of 32 \( \times \) 32, 16 \( \times \) 16, and 8 \( \times \) 8 pixels, respectively.
(Original image courtesy of NASA.)
Segmentation Using Similarity

Watershedding

A kind of region growing
Watershedding

- Think of the gray level image as a Landscape.
- Let water rise from the bottom of each valley (the water from the valley it given its own label).
- As soon as the water from two valleys meet, build a dam, or a watershed.
- These watersheds will define the borders between different regions in the image.

The watershed algorithm can be used directly on the image, on an edge enhanced image or on a distance transformed image.
Watershedding

- Example of watershed applied to gray image
Watershedding

**FIGURE 10.54** (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.
(Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)

**FIGURE 10.54**
(a) Original image. (b) Topographic view. (c)–(d) Two stages of flooding.
Watershed segmentation

- A watershed is a ridge dividing areas drained by different rivers
- Pixel intensities are considered as heights
  - A watershed line divides areas into parts which would collect water that falls on the area

http://cmm.ensmp.fr/~beucher/wtshed.html
Watershedding

**Figure 5.47** One-dimensional example of watershed segmentation. (a) Gray level profile of image data. (b) Watershed segmentation – local minima of gray level (altitude) yield catchment basins, local maxima define the watershed lines.

http://www.engineering.uiowa.edu/~dip/LECTURE/Segmentation3.html#watershed
Watershedding

Procedure

- Insert water at constant rate starting from the minimums
- Regions/basins get flooded
- When distinct basins are about to merge: build a dam
- The dams define the boundaries between regions
- Often applied to the gradient image instead of the original
Watershed segmentation

- Distance transform
  - Calculate the geometrical distance to a height

- Gradients
  - Strong gradients can be watershed lines

- Markers
  - Internal & External
  - All gradients can create too many regions
  - Only use some gradients
Watershedding

Figure 10.55 (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.
Example of application

Original image

Gradient image

Watershed segmentation

After region merging
Match based segmentation

Compare a template to the underlying image to find objects with a certain intensity distribution or shape.

* Pictures from Mean Shift: A Robust Approach toward Feature Space Analysis, by D. Comaniciu and P. Meer http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html