Chapter 6

Segmentation: Thresholding and Region-based method

Image Processing and Computer Vision

LE Thanh Sach

Faculty of Computer Science and Engineering

Ho Chi Minh University of Technology, VNU-HCM

Overview

1. **Thresholding**

2. **Region-Based Segmentation**
   - Region Growing
   - Region Splitting and Merging

Thresholding: Definition

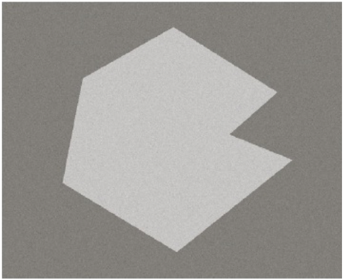
What is thresholding?

Thresholding is a process to **label** each pixel in the input image with its **class** using some threshold value, T

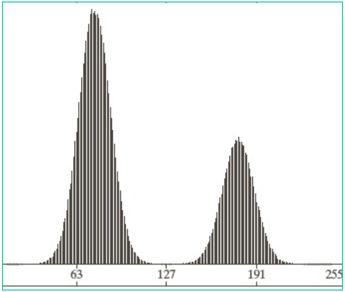
$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \leq T \end{cases}$$

- **labeling** : the process of assigning labels to pixels
- **class** : category of objects, e.g., background, object 1, object 2, etc

Thresholding: Illustration



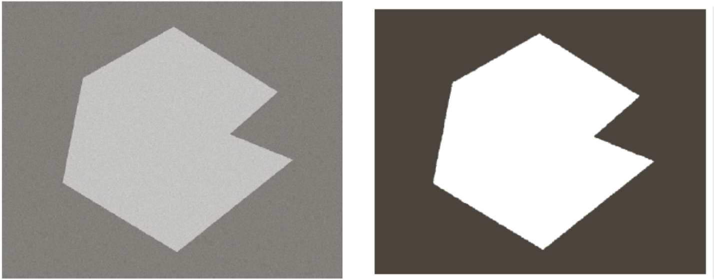
(a)



(b)

Figure: Thresholding Illustration: (a) An image contains a light object in dark background, (b) Histogram of (a), the mode on the right represents object, the mode on the left represents the background

Thresholding: Illustration



(a) (b)

Figure: Thresholding Illustration: (a) Input image, (b) Thresholded image

- T : middle of the left and the right mode $\Rightarrow$ labeling the background as 0 (black) and object as 1 (white)

Thresholding: Concepts

Concepts

- **Global Thresholding**: T is applicable to the entire image.
- **Variable Thresholding**: T changes over an image.
- **Local or Regional Thresholding**: is **Variable Thresholding**, where, T at (x, y) is a function of neighborhood of $f(x, y)$.
- **Dynamic or Adaptive Thresholding** T at (x, y) depends on coordinates (x, y) themselves.

Thresholding: Concepts

Concepts

- **Multiple Thresholding**: Labeling process uses more than one thresholds, e.g., T_1 and T_2

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

- a, b, c are three labels or classes.

Thresholding: Difficulty

Input images are corrupted by noises

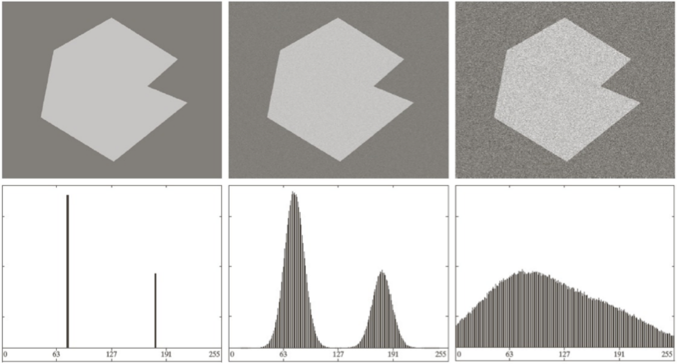


Figure: Left to right: without noise, Gaussian noise ($\mu = 0, \sigma = 10$), Gaussian noise ($\mu = 0, \sigma = 50$)

- **How do you do thresholding in the rightmost case?**

Thresholding: Difficulty

Input images are acquired with variable illumination

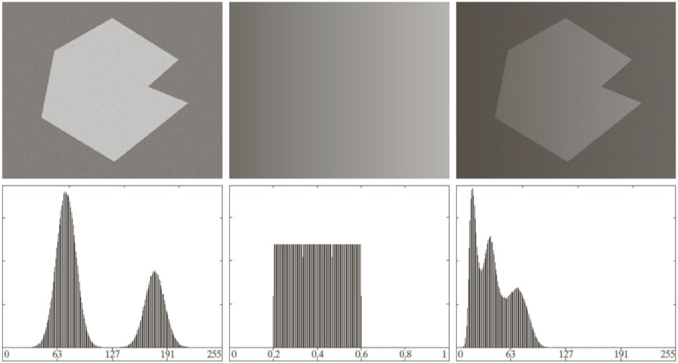


Figure: Left to right: homogeneous illumination, pattern of variable illumination, simulated illumination $\equiv$ left $\times$ middle

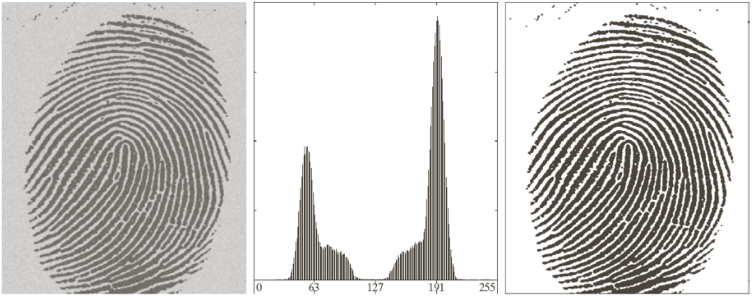
- **How do you do thresholding in the rightmost case?**

Global Thresholding

Iterative method for global thresholding

1. Select an intial estimate for the global threshold, T
2. Perform thresholding with T, yield two groups
   - $G_1 = \{f(x,y) \quad \text{if } f(x,y) > T\}$
   - $G_2 = \{f(x,y) \quad \text{if } f(x,y) \leq T\}$
3. Compute mean m_1 and m_2 for pixels in G_1 and G_2 respectively.
4. Compute new threshold $T = \frac{m_1+m_2}{2}$
5. Repeat Step 2 to 4 until the difference between values of T in two consecutive iterations is less than a predefined parameter ΔT

Global Thresholding: Demonstration



- Started with $T =$ average image intensities
- Stopped with $T = 125.4$, $\Delta T = 0$

Optimal Global Thresholding

Idea of Otsu' method

- Maximize the between-class variance
- $\equiv$ Select a threshold value (intensity) that provide the largest distance between the two classes
- $\equiv$ Make two classes best separated according to the selected threshold value.

Questions?

The between-class variance: How is it defined?

Optimal Global Thresholding

Concepts:

- L: number of gray-levels
- **gray-level:** $0, 1, 2, .., L-1$
- **image size:** $M \times N$
- n_i: number of pixels with intensity i
- $M \times N = n_1 + n_2 + .. + n_{L-1}$
- **normalized histogram**: $\{p_i = \frac{n_i}{M \times N}\}$, for $i \in [0, L-1]$

Optimal Global Thresholding

- Assume that threshold $T = k$, for $k \in [0, L-1]$
- $\Rightarrow$ Two classes:
  1. $C_1 = \{0, 1, .., k\}$
  2. $C_2 = \{k+1, k+2, .., L-1\}$
- **Probability that a pixel is assigned to C_1**: $P_1(k)$
- $P_1(k) \equiv$ **Probability of class C_1 occurring**

$$P_1(k) = \sum_{i=1}^{k} (p_i) \quad (1)$$

item **Probability of class C_2 occurring**: $P_2(k)$

$$P_2(k) = \sum_{i=k+1}^{L-1} (p_i) = 1 - P_1(k) \quad (2)$$

Optimal Global Thresholding

- **Mean value of pixels is assigned to C_1:**

$$
\begin{aligned}
m_1(k) &= \sum_{i=0}^{k} [i \times P(i|C_1)] \\
&= \sum_{i=0}^{k} \left[i \times \frac{P(C_1|i) \times P(i)}{P(C_1)} \right] \\
&= \frac{1}{P_1(k)} \times \sum_{i=0}^{k} (ip_i)
\end{aligned} \tag{3}
$$

- $P(C_1)$: Probability of C_1 occurring $\equiv P_1(k)$
- $P(C_1|i) \equiv 1$: we are considering only C_1
- $P(i) \equiv p(i) \equiv$ Probability of intensity i occurring

Optimal Global Thresholding

- **Mean value of pixels assigned to C_2**:

$$m_2(k) = \sum_{i=k+1}^{L-1} \left[i \times P(i|C_2) \right] = \frac{1}{P_2(k)} \times \sum_{i=k+1}^{L-1} (ip_i) \tag{4}$$

- **Mean value of pixels in the entire image (global mean)**: m_G

$$m_G = \sum_{i=0}^{L-1} (ip_i) \tag{5}$$

Optimal Global Thresholding

- $P_1 + P_2 = 1$
- $P_1 m_1 + P_2 m_2 = m_G$
- **cumulative mean up to level k**: $m(k)$

$$m(k) = \sum_{i=0}^{k} (ip_i) \quad (6)$$

Optimal Global Thresholding

- **Between-class variance**: $\triangleq \sigma_B^2(k)$

$$\sigma_B^2(k) = P_1(m_1 - m_G)^2 + P_2(m_2 - m_G)^2 \quad (7)$$

$$\begin{aligned} \sigma_B^2(k) &= P_1P_2(m_1 - m_2)^2 \\ &= \frac{(m_G P_1 - m)^2}{P_1(1 - P_1)} \\ &\equiv \frac{\left[m_G P_1(k) - m(k)\right]^2}{P_1(k)\left[1 - P_1(k)\right]} \end{aligned} \quad (8)$$

- **Optimized threshold** T:

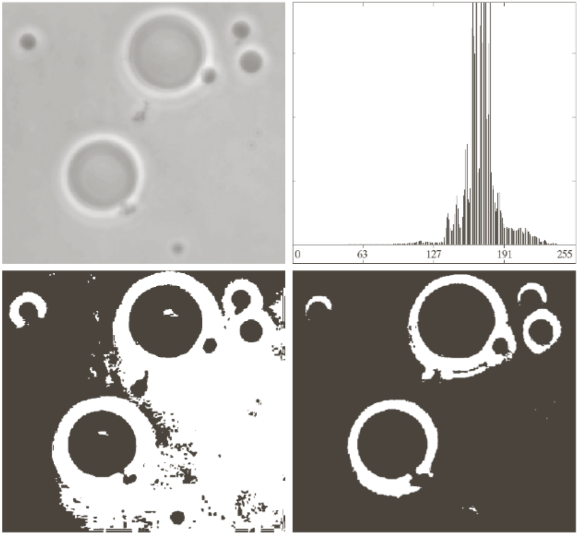
$$T = \underset{k}{\operatorname{argmin}}\, \sigma_B^2(k) \quad (9)$$

Optimal Global Thresholding

Otsu's Algorithm

1. Compute the normalized histogram of the input image, i.e., p_i, for $i \in [0, L-1]$
2. Compute the cumulative sum $P_1(k)$
3. Compute the cumulative mean $m(k)$
4. Compute the global mean m_G
5. Compute the between-class variance $\sigma_B^2(k)$, for $i \in [0, L-1]$
6. Find and return k (as threshold T) for which $\sigma_B^2(k)$ is maximum.

Optimal Global Thresholding: Demonstration



a b
c d

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.)

Thresholding - Challenging cases

Some challenging cases

1. The input was corrupted by noises
2. The size of objects and background is unbalanced
3. The input was acquired with variable illumination

Thresholding - Challenging cases: with noise

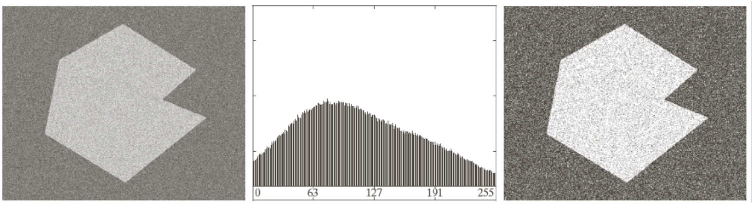
with noise: solutions

- Denoise before thresholding
- **Unfortunately, it is not always to remove noise with an acceptable result**

Remind

- For additive noise:
  1. Estimate probability density function of noise
  2. Remove noise using filters (linear and non-linear)
- For multiplicative noise:
  1. Create $g(x, y) = log(f(x, y)$ to transform multiplicative to additive noise
  2. Remove additive noise

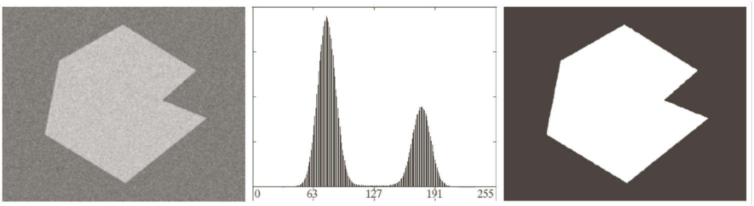
Thresholding - Challenging cases: with noise



From left to right:

- Input image: corrupted with additive Gaussian noise $(\mu = 0, \sigma = 10) \leftarrow$ **assume that possibly estimated**
- Histogram: show a unimodel density function $\Rightarrow$ **impossible to perform thresholding directly**
- Result of Otsu' method

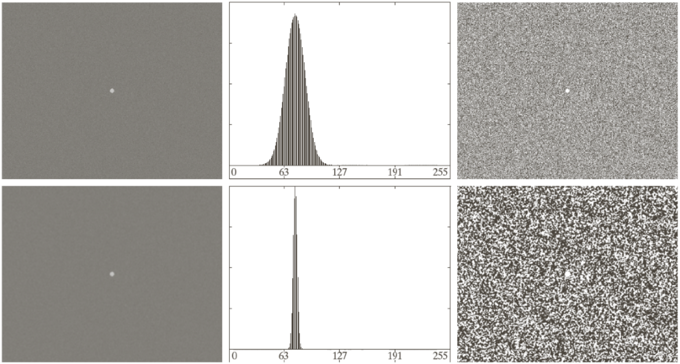
Thresholding - Challenging cases: with noise



From left to right:

- Input image: smoothing the corrupted image with average kernel ← **assume that possibly estimated to know type and parameters of noise**
- Histogram: show a bimodel density function with a valley ⇒ **possible to perform thresholding**
- Result of Otsu's method

Thresholding - Challenging cases: unbalanced size



- Left column: Noisy images without (top) and with (bottom) smoothing. The size of background (large area) and foreground (a small circle) are quite different - **unbalanced size**.
- Middle column: Histogram shows unimodel ⇒ **impossible to perform thresholding directly**
- Right column: result of Otsu's method

Thresholding - Challenging cases: unbalanced size

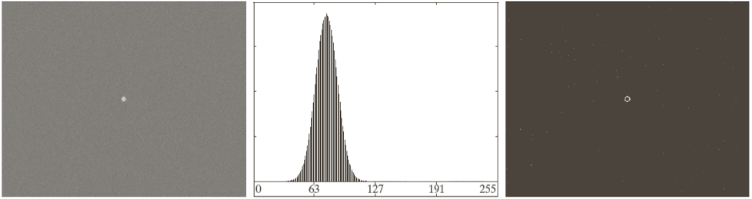
Unbalanced sizes - A solution

Focus on pixels on and round edge only in the input image

A solution:

1. Compute gradient or Laplacian of the input image $f(x, y)$.
2. Chose a threshold T for obtaining pixels with large rate of intensity change.
3. Threshold the image obtained from Step 1 with T and return $g_T(x, y)$. $g_T(x, y)$ is used to select pixels on and around edges in the input image.
4. Compute histogram of pixels in $f(x, y)$ that are marked by $g_T(x, y)$
5. Determine threshold T_f from the histogram obtained from Step 4, by any method, e.g., Otsu's method.
6. Perform thresholding $f(x, y)$ with T_f

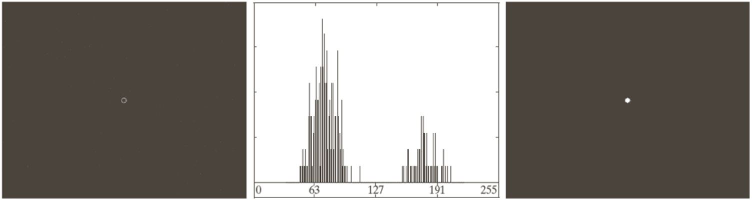
Thresholding - Challenging cases: unbalanced size



Left to right:

- Input image: object and background have unbalanced sizes.
- Histogram of the input
- Gradient image, thresholded at 99.7 percentile.

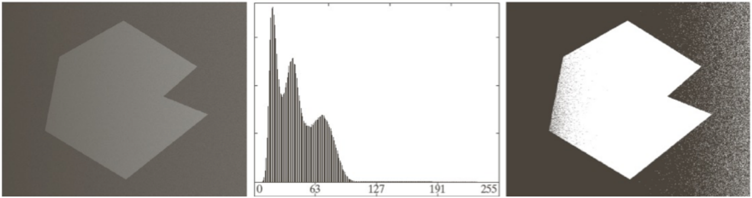
Thresholding - Challenging cases: unbalanced size



Left to right:

- Pixels on and around edges in the input image $\equiv f(x,y).*f_g(x,y)$. $f_g(x,y)$ is thresholded gradient image, shown in previous figure. $.* \equiv$ point to point multiplication.
- Histogram of selected pixels
- Thresholded image with a threshold $T = 134$ calculated by Otsu's method

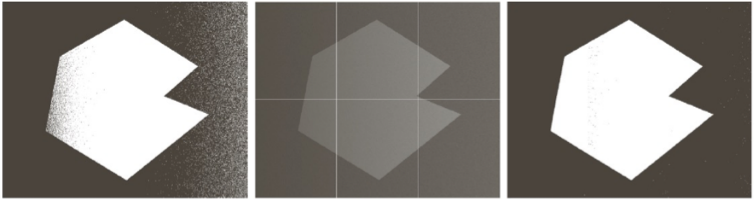
Thresholding - Challenging cases: nonuniform illumination



Left to right:

- Input image: object;s pixels are the same. **Non-illumination** $\Rightarrow$ object's pixels vary in their gray-level gradually
- Histogram of the input image, has **more than one valleys** $\Rightarrow$ **difficult for global thresholding**.
- Thresholded image, using iterative global thresholding.

Thresholding - Challenging cases: nonuniform illumination



Left to right:

- Input image: Thresholded image, using Otsu's method.
- Subdivide the input image into small **non-overlapped rectangles** to expect they **have uniform illumination**.
- Input image: Thresholded image, using Otsu's method **for each small rectangle**.

Thresholding - Challenging cases: nonuniform illumination

Nonuniform illumination - Discussion

- When ones **subdivide** the whole input image **into small non-overlapped rectangle**. Each rectangle is modeled as having **uniform illumination** $\Rightarrow$ can perform each rectangle by methods workable with uniform illumination, e.g., iterative global thresholding, Otsu's method, etc.
- Subdivision also **introduces another challenging**. Each rectangle can has **unbalanced sizes** of object and background $\Rightarrow$ solve this added problem too!

Local Thresholding

Local Thresholding

- **Suitable for cases**: threshold T at (x, y) functionally depends on pixels in a neighborhood of (x, y)

General framework:

For each pixel (x, y) in the input $f(x, y)$

1. Let S_{xy} be the neighborhood of (x, y)
2. Let T_{xy} be the threshold being computed for pixel (x, y)
3. Compute T_{xy} from S_{xy}
4. Perform thresholding with the computed T_{xy} according to the following equation.

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

Local Thresholding

General framework:

- Thresholding step can be done by using a general rule as follows:

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

In general, **Q(local parameters)** is a **predicate**. An example is as follows:

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

Local Thresholding

General framework: An example

- Let a and b be nonnegative parameters.
- $\sigma_{xy} \equiv$ standard deviation of pixels in S_{xy}
- $m_{xy} \equiv$ average mean of pixels in S_{xy}
- $m_G \equiv$ global mean of pixels in input image
- Compute threshold T_{xy} by one of the following equations.

$$T_{xy} = a\sigma_{xy} + bm_{xy}$$

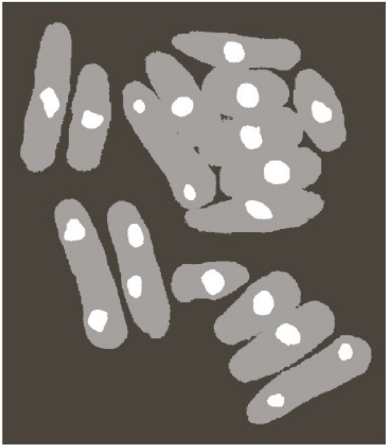
Or,

$$T_{xy} = a\sigma_{xy} + bm_G$$

Local Thresholding: Demonstration

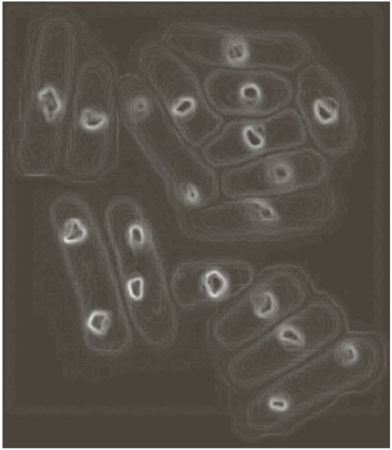
Figure: Original Image: has **three groups of intensities**

Local Thresholding: Demonstration



- This result is obtained by thresholding with 2 optimized thresholds values.
- Bright dots can be detected correctly. However, gray intensities tends to be merged together.

Local Thresholding: Demonstration



- Local region: 3×3. The result is obtained using $\sigma_{x,y}$
- Boundary of objects can be locate correctly.

Local Thresholding: Demonstration

- Local region: 3×3. $a = 30, b = 1.5$. The result is obtained using predicate given above, with replacing m_{xy} by m_G
- Objects are locate correctly.

Local Thresholding: Moving Average

This is a special case of local thresholding

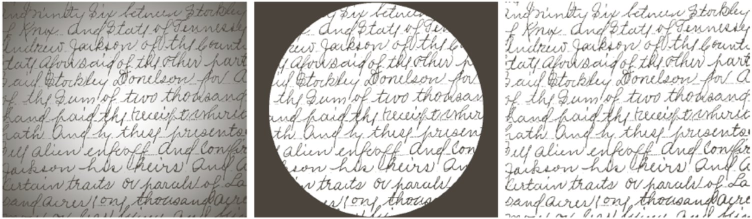
- Often used in document processing

- Scan the input from left to right and right to left for next line, i.e., zig-zag order.
- Compute the average for a window of n recent pixels
- Let z_{k+1} be the intensity encountered at step $k+1$ on the zig-zag scanning process.
- Compute the intensity mean for this new point as follows:

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

- Perform thresholding with $T = bm_{xy}$. b: a constant, m_{xy} is $m(k)$ computed for pixel (x, y)

Local Thresholding: Moving Average - Demonstration



From left to right:

- Input image: corrupted by spot shading
- Thresholded image by using Otsu's method
- Thresholded image by using moving average method. $n = 20$ ($5\times$ stroke's width), $b = 0.5$

Local Thresholding: Moving Average - Demonstration

From left to right:

- Input image: corrupted by sinusoidal intensity variation.
- Thresholded image by using Otsu's method
- Thresholded image by using moving average method.
  $n = 20$ ($5\times$ stroke's width), $b = 0.5$

Region Growing

- What is region's meaning in Region Growing?

Region consists of points that

- Space: these points are connected (**proximity**).
- Feature: features extracted for these points must **satisfy some criteria**, for examples,
  - similar in intensity or color values
  - extracted textures, moments, etc must satisfy some criteria

Region Growing

- **proximity** ⇒ can expand the region from **seed points**, i.e., **growing**
- **extracted features must satisfy some criteria** ⇒ verify the **criteria** during the growing

Region Growing

Basic ideas

- At the beginning, a region contains only **seed points**, one or many.
- The algorithm selects **neighboring points** of the region (use 4- or 8-connectivity) to add to the region. The selected points must **be similar to seed points** according to some **predefined criteria**, for examples,
  - in the same intensity range with seed points
  - similar in color, etc
- The algorithm can perform the above selection process **many passes**.

Region Growing

Questions

- What are "seed points"?
- How do we obtain "seed points"?
- How do we specify selection rules (predefined criteria)?
- When does the algorithm stop?

Region Growing

- What are "seed points"?
- How do we obtain "seed points"?

Seed points:

- A typical point for the region.
  - in space: inside of the region, or the **region's centroid**
  - in feature space: satisfy the predefined criteria
- **Example of selecting seed points**
  1. perform thresholding with high threshold T to select typical region.
  2. erode the resulting regions to obtain the centroids.
  3. resulting centroids $\equiv$ seed points

Region Growing

- How do we specify selection rules (predefined criteria)?

Selection rules depends on:

- The problem under consideration
- Type of input image
- Examples:
  - Satellite images heavily depends on color ⇒ use selection rules base on color
  - In general, a feature extraction process should be performed

- When does the algorithm stop?

- Region growing stops if there is no more proximity points satisfying the selection rules.

Region Growing

Inputs

- $f(x, y)$: input image
- $S(x, y)$: array contains marks (value 1) for seed points.
- $Q(x, y)$: feature-based selection rule, a predicate, being applied at each (x, y)

Step 1: Find typical seed points from mask $S(x, y)$

1. Find connected components in (x, y)
2. Erode connected components to single point $\equiv$ the centroid of the components.
3. Set $S(x, y) = 1$ if (x, y) is the centroid. Otherwise, set $S(x, y) = 0$

Region Growing

Step 2: Apply feature-based selection rule to obtain $f_Q(x, y)$

1. Apply feature-based selection rule Q for each pixel (x, y) to obtain $f_Q(x, y)$

$$f_Q(x,y) = \begin{cases} 1 & \text{if } Q(x,y) \text{ is } \textbf{true} \\ 0 & \text{if } Q(x,y) \text{ is } \textbf{false} \end{cases}$$

Region Growing

Step 3: Verify the proximity criteria

Form an image $g(x, y)$:

1. Set $g(x, y) = 1$ for each seed point.
2. For each point in $f_Q(x, y)$,

$$g(x,y) = \begin{cases} 1 & \text{if } f_Q(x,y) = 1 \textbf{ AND} \\ & (x,y) \text{ is 8-connected to seed points} \\ 0 & \text{otherwise} \end{cases}$$

Region Growing

Step 4: Discover segments

1. Find connected components in $g(x, y)$
2. Assign different values (1, 2, etc) to each of the discovered components
3. each component $\equiv$ a segment

Region Growing: Demonstration

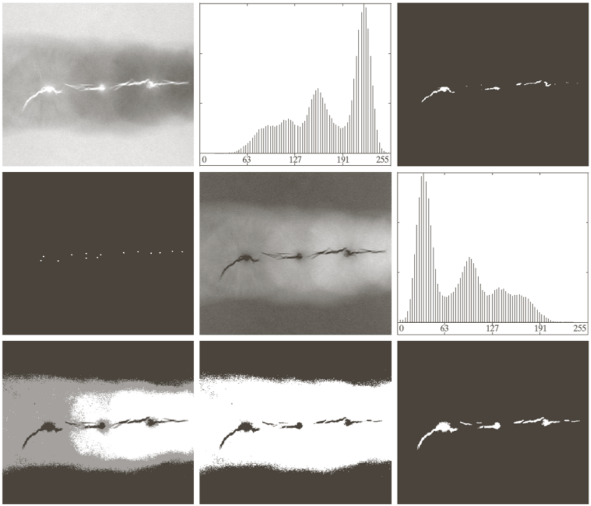


FIGURE 10.51 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between (a) and (c). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference 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 and Merging

Basic ideas

- Perform a splitting process
  - Split the input image or regions into **disjoint regions**, usually **disjoints quadrants**, if the input image and regions are not satisfied **segmentation's criteria**.
- Perform a splitting process
  - Merge two adjacent regions into one, if **the union of these satisfy the segmentation's criteria**.

Region Splitting and Merging: Splitting

Inputs

- R: be the input image or a region
- Q: be a predicate being verified for any region

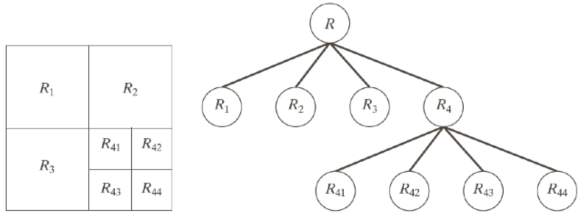
$$Q(R) = \begin{cases} \textbf{true} & \text{if region is uniformed, } \textbf{no splitting required} \\ \textbf{false} & \text{if region is non-uniformed, } \textbf{splitting required} \end{cases}$$

Region Splitting and Merging: Splitting

Data structure helping splitting process

Quadtree:

- Root ⇔ The input image
- Each node: has four descendants corresponding to four disjoint quadrants in image.



a b

FIGURE 10.52 (a) Partitioned image. (b) Corresponding quadtree. R represents the entire image region.

Region Splitting and Merging: Merging

Merging condition

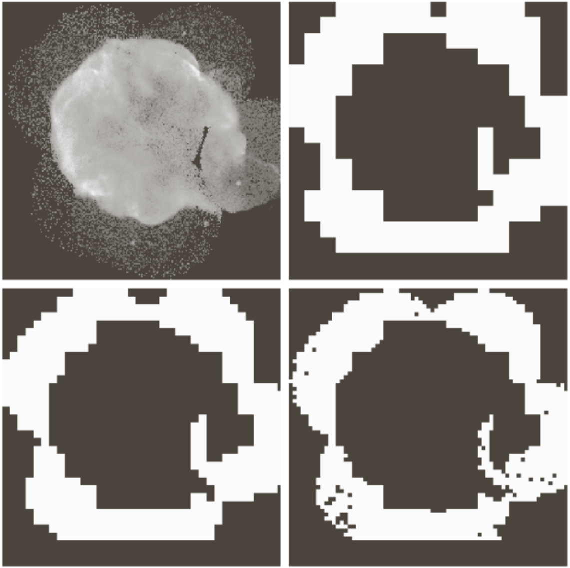
- R_i and R_j be **adjacent regions**
- R_i and R_j are merged into one if $Q(R_i \cap R_j) =$ **true**

Region Splitting and Merging

Algorithm

- Split into four disjoint quadrants any R_i for which $Q(R_i) =$ **false**
- When no further splitting is possible, merge any adjacent regions R_i and R_j for which $Q(R_i \cap R_j) =$ **true**
- Stop when no further merging is possible

Region Splitting and Merging: Demonstration



a b
c d

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 32×32, 16×16, and 8×8 pixels, respectively. (Original image courtesy of NASA.)