Image Segmentation

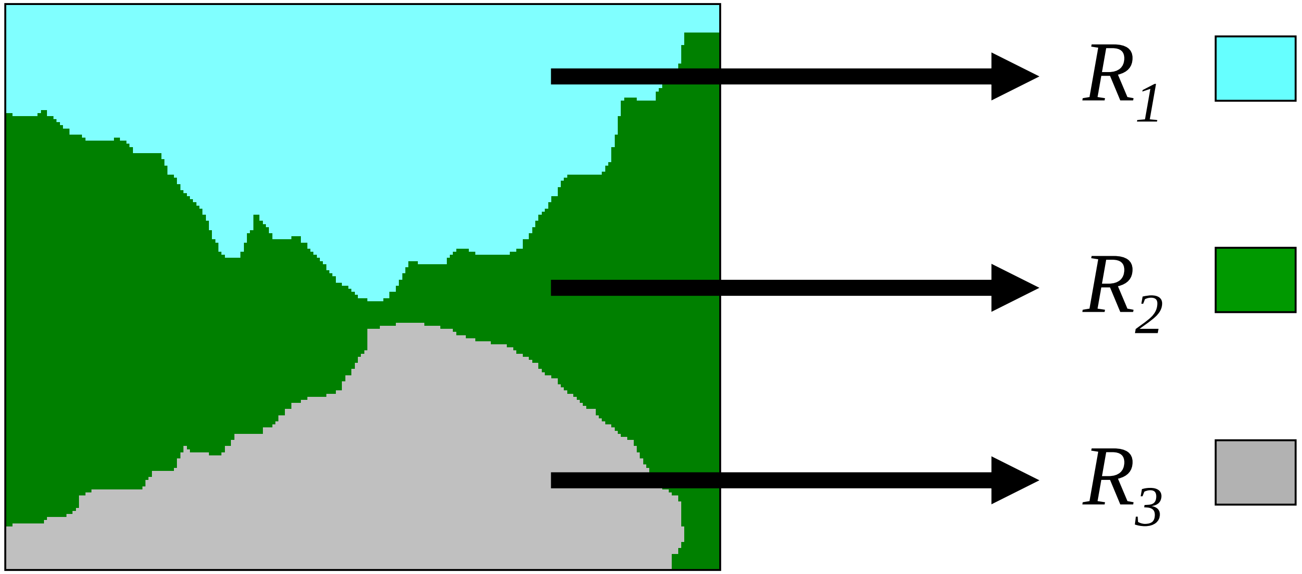
Image Segmentation

- The task is to divide the image pixels into "regions", each containing a subset of image pixels.
- This is a required (and difficult) step before any tasks of image understanding.
- We first focus on the fundamental methods that do not use knowledge of the "real-world objects" (i.e., semantics) in the image.
- The basic ideas used for segmentation:
  - Discontinuity between pixels indicates boundaries between regions.
  - Similar pixels are likely to belong to the same object.

Image Segmentation

- Goal: To partition an image (represented as R) into several regions $R_1, R_2, \ldots, R_n$.
- Example conditions:
  - The union of all R_i ($1 \le i \le n$) equals R.
  - The regions are disjoint (no overlap).
  - Each R_i is a connected set.
  - $Q(R_i)$ is true for each R_i.
  - $Q(R_i \cup R_j)$ is false for each pair of R_i and R_j that are adjacent.
- Here $Q(R_k)$ is a logical predicate (condition) that is used to determine whether R_k should be a single region.

Image Segmentation



Here a possible form of $Q(R_k)$:

Each pixel in R_k needs to have the same color.

Two Approaches to Segmentation

- Similarity-based: Segmentation methods that proceed by grouping similar pixels.
- Discontinuity-based: Segmentation methods that attempt to identify discontinuities (boundaries, edges) between regions.

Line Detection

- A line is an important basic shape element.
- Applications: Wires in circuits, roads in maps or aerial photographs, etc.
- Global method: Hough transform (to be discussed later).
- Local method: Line filters (examples given below).

A set of 3x3 line filters

$$
\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}
\quad
\begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}
\quad
\begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}
\quad
\begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}
$$

Limitations:

- Line width
- Direction resolution
- Response from edges

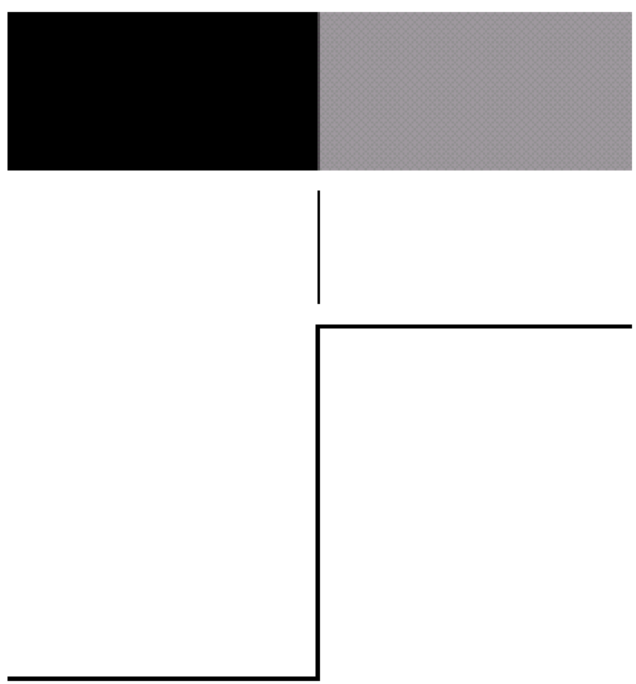
Line Filter Examples

- Only "bright" lines considered here.

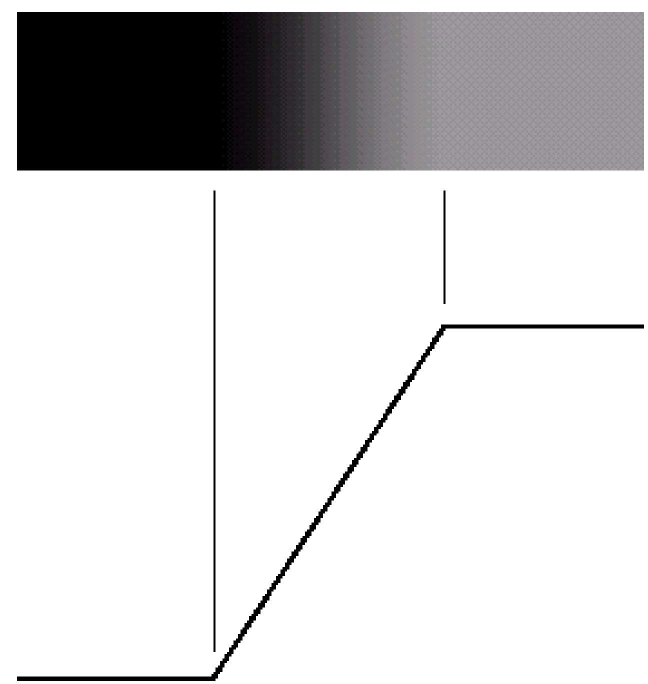
Edge Detection

Edges are discontinuities in an image.

The discontinuity can be abrupt or gradual:



Ideal edge



Ramp edge

Edge Detection

Three steps of edge detection:

- Preprocessing: Such as smoothing / noise reduction.
- Edge point detection: Extract from the image a set of pixels that are candidate edge points.
- Edge localization: Select from the candidate edges points only the points that actually form the edges.
    - Linking candidate edge points into edges.
    - Removing candidate edge points that might not belong to actual edges.

Edge Detection

Edge detection: 1st derivative vs. 2nd derivative:

Add some noise:

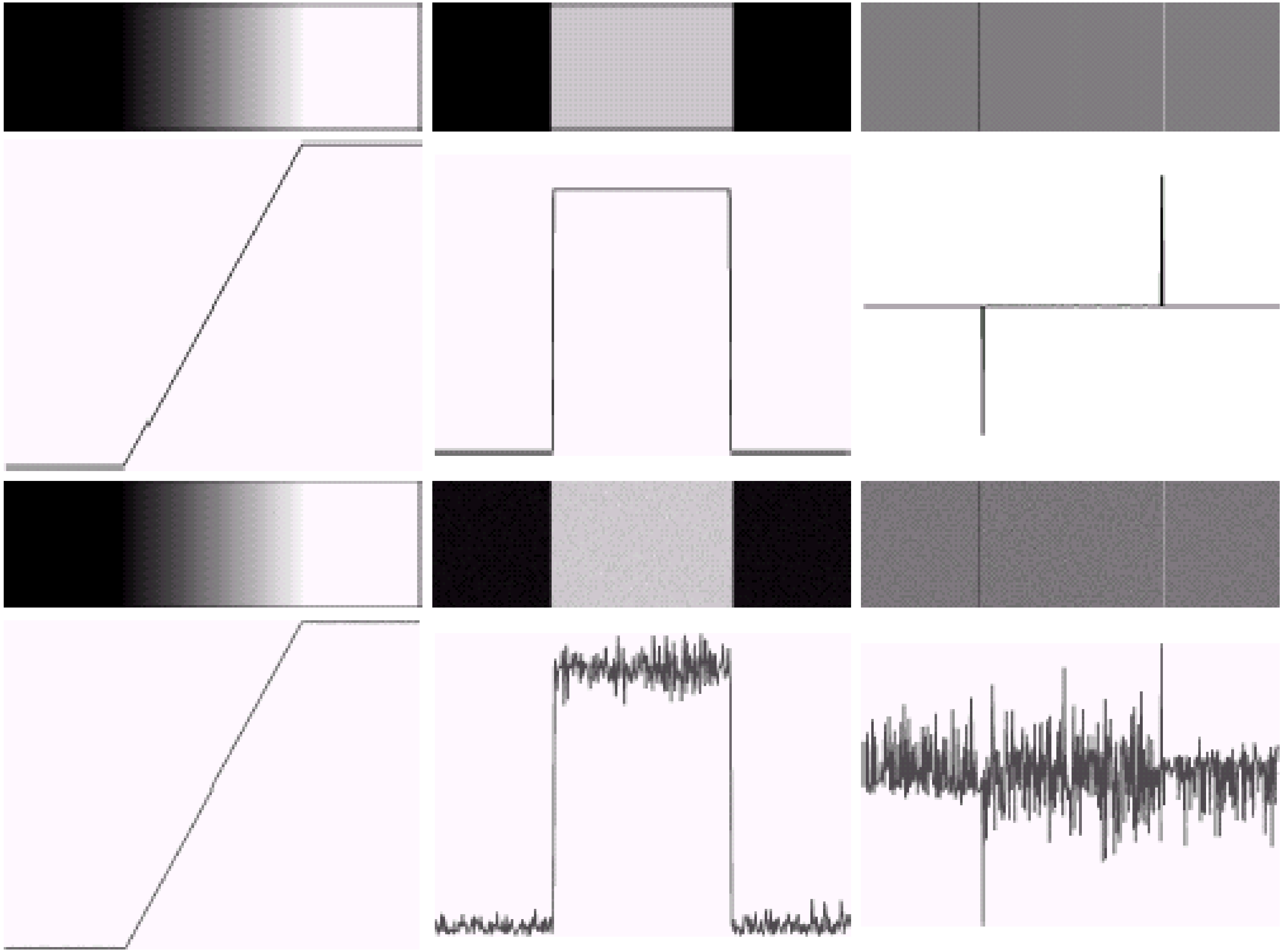


Image Gradients

The gradient of a scalar function:

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

The magnitude of the gradient (in image processing, this sometimes is imprecisely referred to as the gradient itself):

$$|\nabla f| = \sqrt{g_x^2 + g_y^2} = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \qquad \text{(L1 norm)}$$

For simplicity in computation, this is sometimes approximated as:

$$M(x, y) = |\nabla f| \approx |g_x| + |g_y| \qquad \text{(L2 norm)}$$

Filters for Image Gradients

Filters for 1-D first derivatives:

Prewitt Filters:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Sobel Filters:

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Filters that approximate 1-D first derivatives at 45 degrees:

Prewitt Filters:

-1	-1	0
-1	0	1
0	1	1

0	1	1
-1	0	1
-1	-1	0

Sobel Filters:

-2	-1	0
-1	0	1
0	1	2

0	1	2
-1	0	1
-2	-1	0

Image Gradients

Examples:

$|g_x|+|g_y|$

Image Gradients

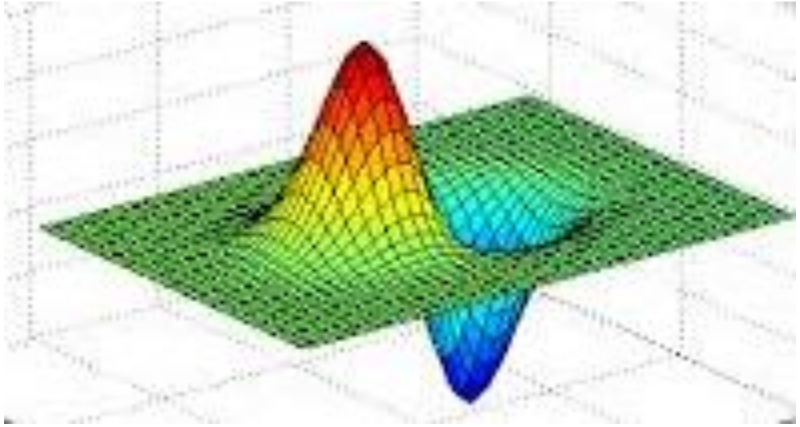
Gradient computation after smoothing:

$|g_x|+|g_y|$

Image Gradients and Edges: Some Notes

- Direction: Gradient direction and edge direction are perpendicular to each other.
- For color images, it is common to compute the gradients in the three color channels. These gradients can then be combined (e.g., L1 norm or L2 norm) for, say, edge detection.
- For Gaussian smoothing, the value of σ affect the response of edges of different widths.

A "derivative of Gaussian" filter for 1-D gradients:



Laplacian for Edge Detection

- Normally, Laplacian is not directly used for edge detection. Reasons:
  - Sensitivity to noise
  - Double edge
  - No information of edge direction
- However, it is still useful for
  - Edge detection by the zero-crossing of Laplacian (after smoothing)
  - Determining whether a given pixel near an edge is at its dark or light side

Laplacian of Gaussian (LoG)

- Smooth the image with a Gaussian smoothing filter first, and then the Laplacian filter is applied.
- The width parameter of the Gaussian filter (σ) can be adjusted to find edges of different widths. Such edges can result from, say, different degrees of blurring.
- Both Gaussian smoothing and Laplacian are linear filters, so they can be combined to form a single composite filter.

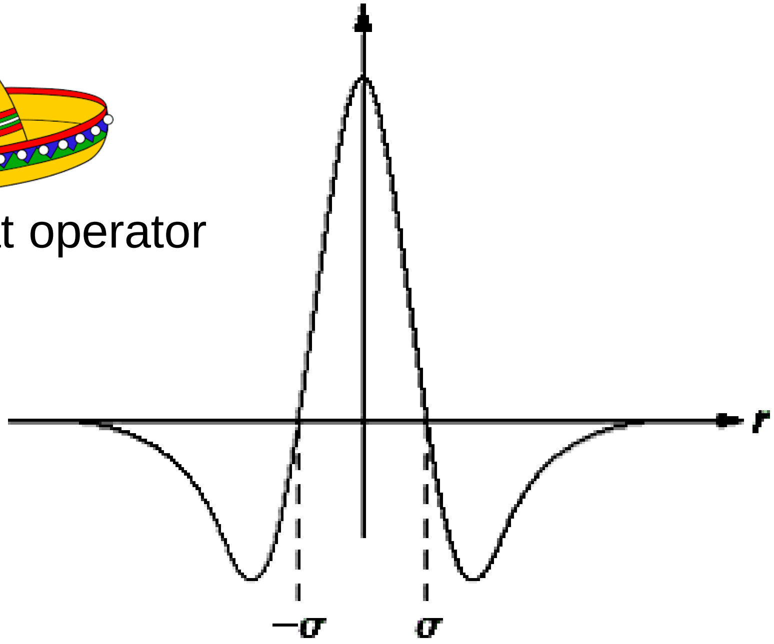
Laplacian of Gaussian (LoG)

$$h(r) = \exp(-r^2 / 2\sigma^2)$$

LoG: $$-\nabla^2 h(r) = -\left(\frac{r^2 - 2\sigma^2}{\sigma^4}\right)\exp(-r^2 / 2\sigma^2)$$

An example composite filter for LoG:

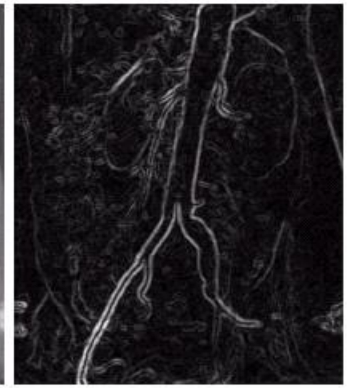
A Mexican-hat operator



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

Laplacian of Gaussian (LoG)

Original Image

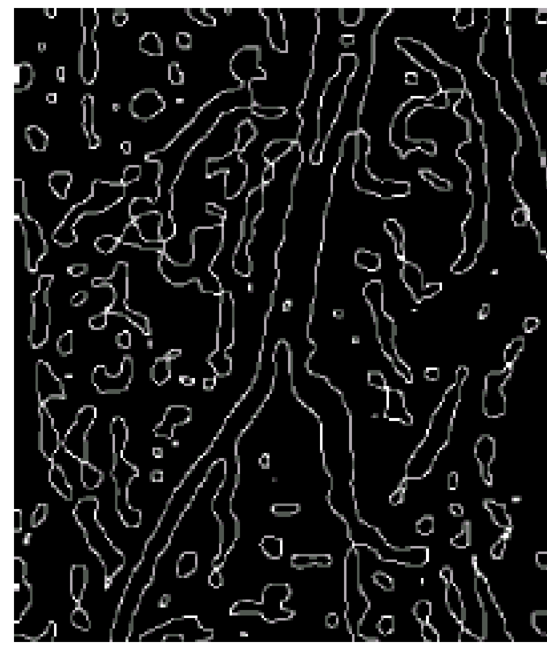


Gradient Magnitude

LoG

Threshold=0

Zero Crossing



Local Processing for Edge Linking

- What we have actually done is the detection of edge points in an image. We have not actually "connect" the edge points to, say, determine the boundary of a region.
- Local processing for connecting edge points:
  - For a given edge points, find other edge points in its predefined (e.g., 3x3 or 5x5) neighborhood.
  - Connect the selected edge point to other "compatible" edge points in the neighborhood.
  - Compatibility can be determined according to gradient magnitude, gradient direction, distance, and relative position, etc. There are many different algorithms.
  - Isolated edge points that can not be connected may actually come from noise and can be discarded.

Canny Edge Detector

- Generally performs better than using simple filters (Sobel, LoG, etc.). The steps include:
  - Gaussian smoothing. The reason is similar to LoG.
  - Gradient magnitude and angle computation using a pair of directional first-derivative operators (e.g., Sobel).
  - Nonmaxima suppression (thinning of edges): Keep only the "strongest" edge point along the gradient direction.
  - Double thresholding: Retain all the "strong" edge points as well as the "weak" edge points that are connected to the "strong" edge points.

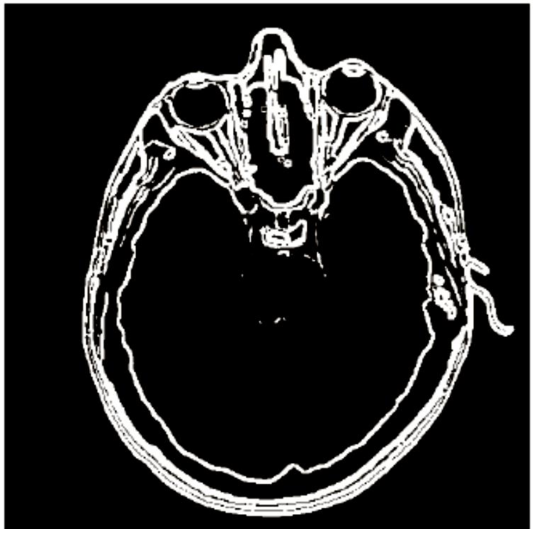
Canny Edge Detector

- Details about double thresholding:
  - Candidate edge pixels with value (gradient magnitude) $\geq T_H$ (higher threshold) are considered valid edge points.
  - Candidate edge pixels with value $< T_H$ and $\geq T_L$ (lower threshold) are considered valid edge points only if they are connected to other valid edge points.
  - The first condition removes some spurious edge pixels that are caused by noise or very small features.
  - The second condition has the effect of edge linking.

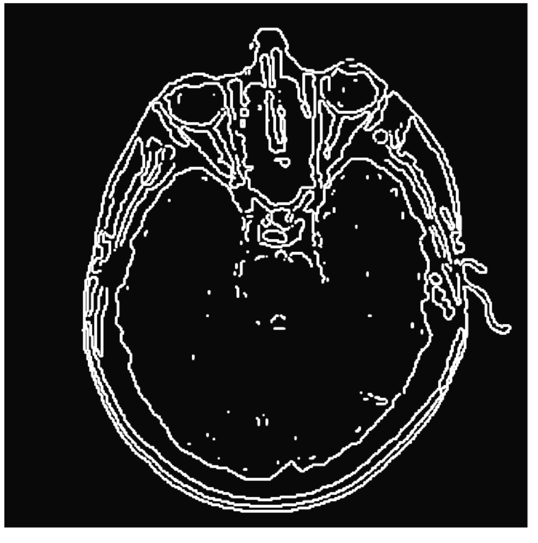
Edge Detection Example

Original

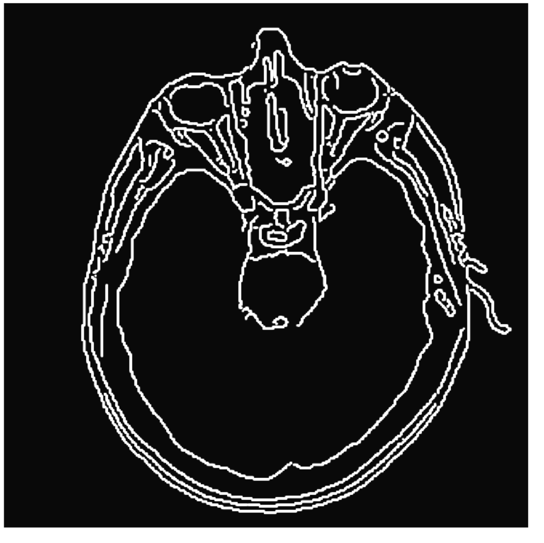
Thresholded Gradient



LoG



Canny

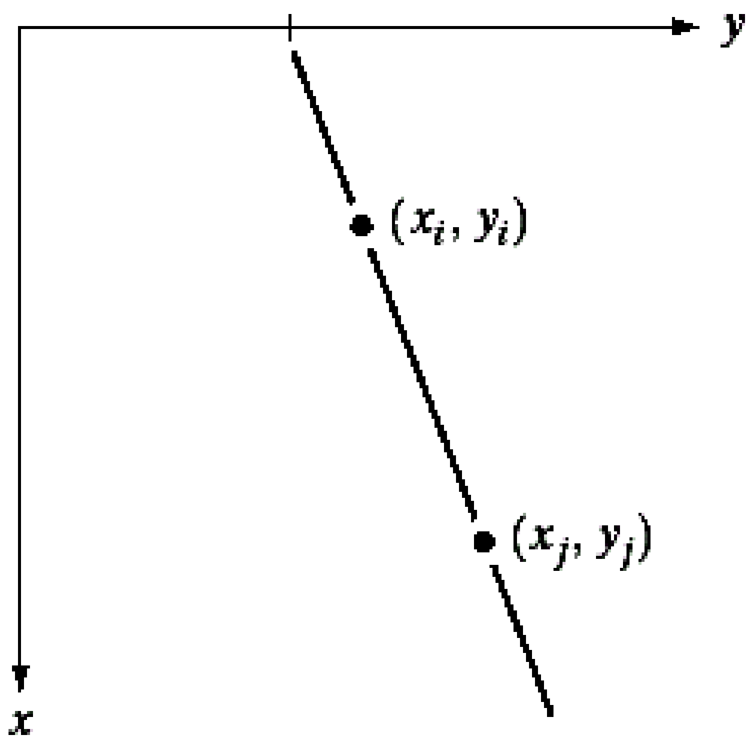


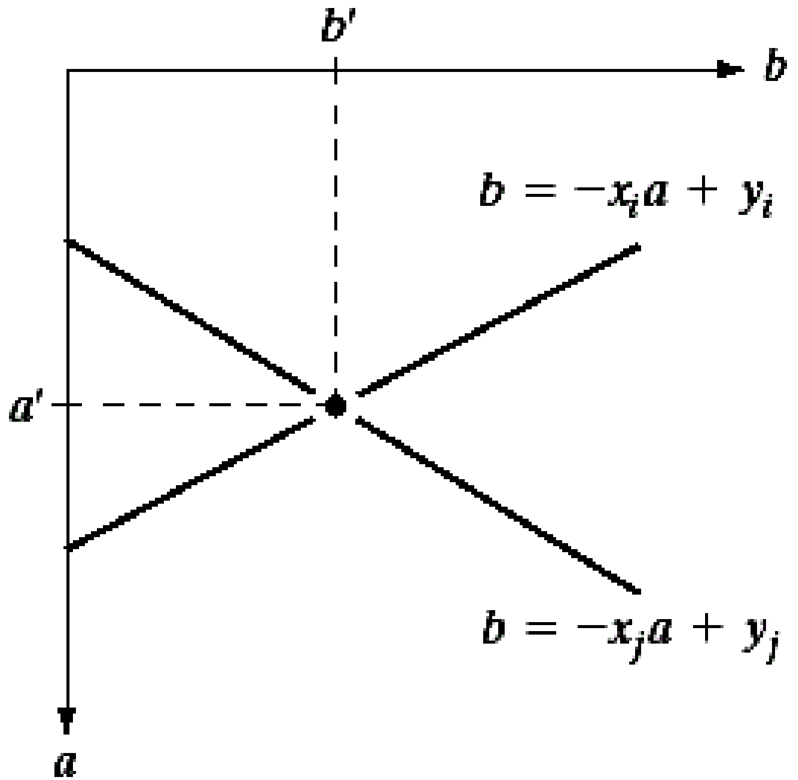
Global Line Detection

- The question here: Can we find all the (significant) straight lines in an image?
- Mostly used in gradient images to find straight edges.
- A Brute-force approach, assuming there are n edge points in the image:
  - $O(n^2)$ line candidates (one candidate line from each pair of edge points)
  - $O(n^3)$ to score the line candidates

Parameter Space of Lines

- A line with the equation $y=ax+b$ is defined by the two parameters a and b.
- A line in the original (image) space is a point in the parameter space, and vice versa.
- One problem is that the parameter a is unbounded.



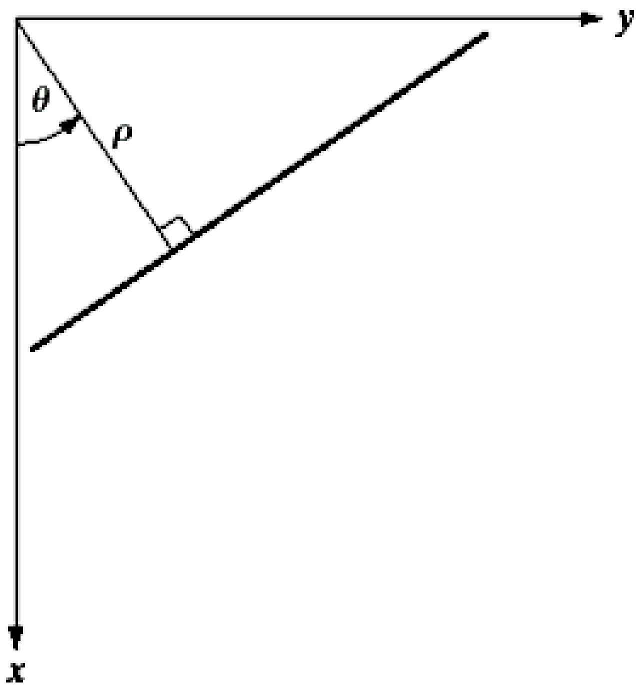


Parameter Space of Lines

- A different parameter space of lines:

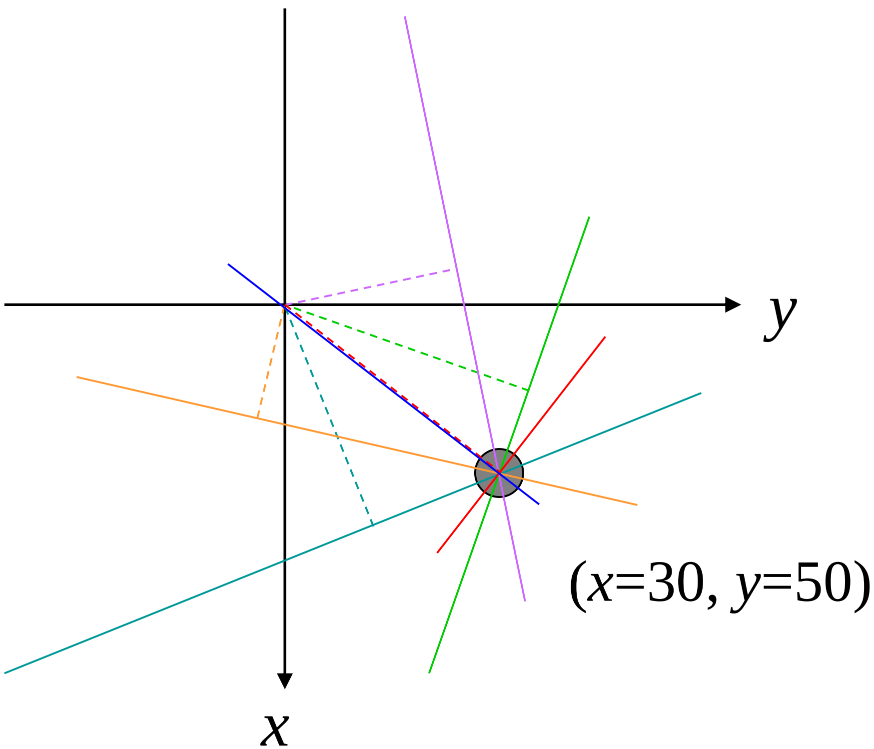
$$x\cos\theta + y\sin\theta = \rho$$

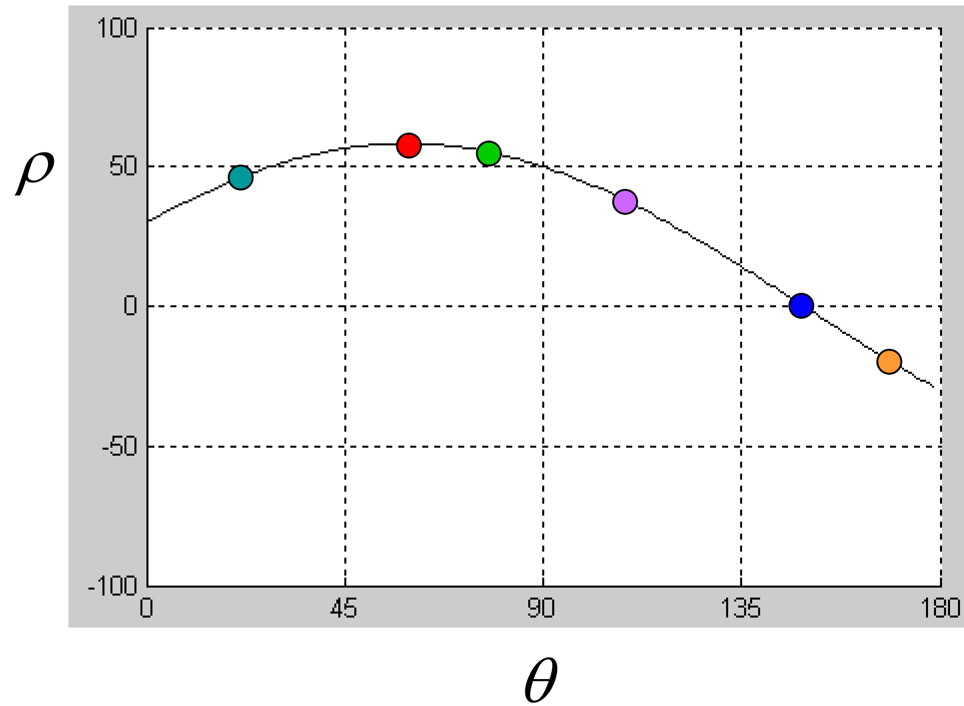
- Two parameters: ρ and θ (think polar coordinates). Both have finite ranges of values.
- The common choice of origin (for computing ρ) is the center of the image.
- A point in the image is a sinusoidal curve in the parameter space.



Parameter Space of Lines

A point in the image is a sinusoidal curve in the parameter space.

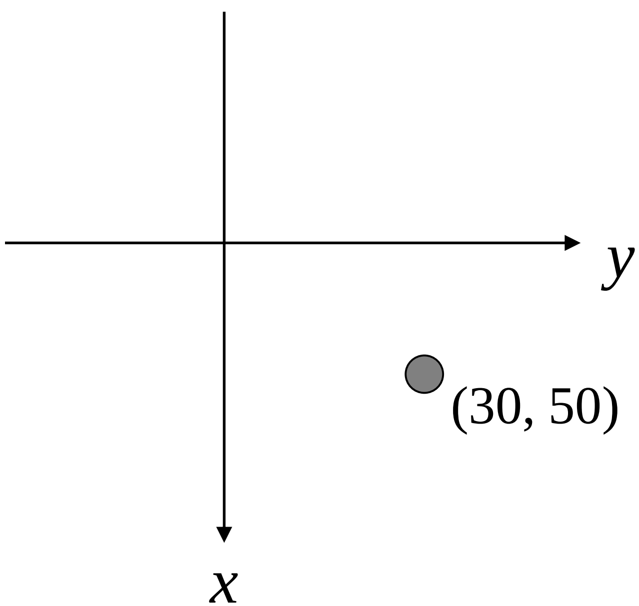


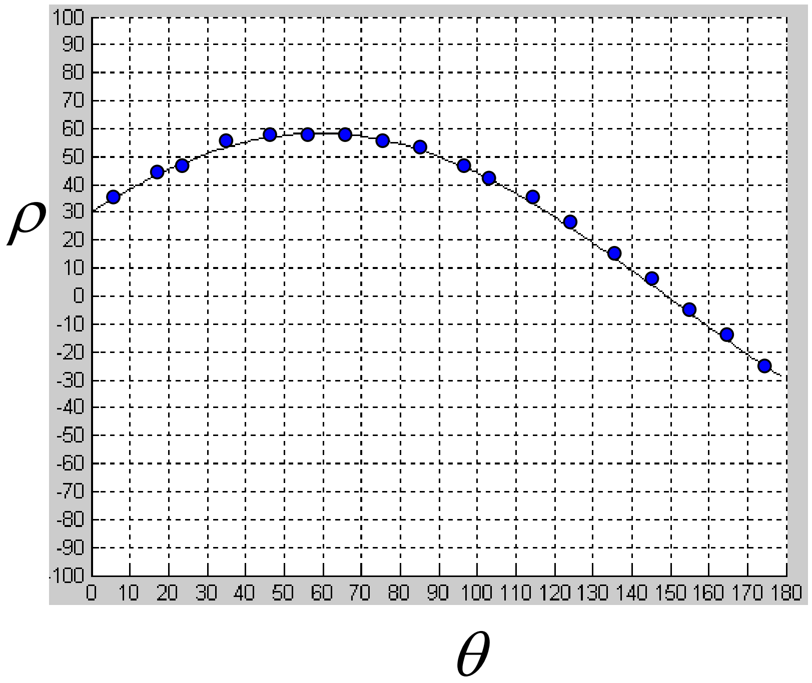


Each point on this curve corresponds to a line that passes through the given point in the image.

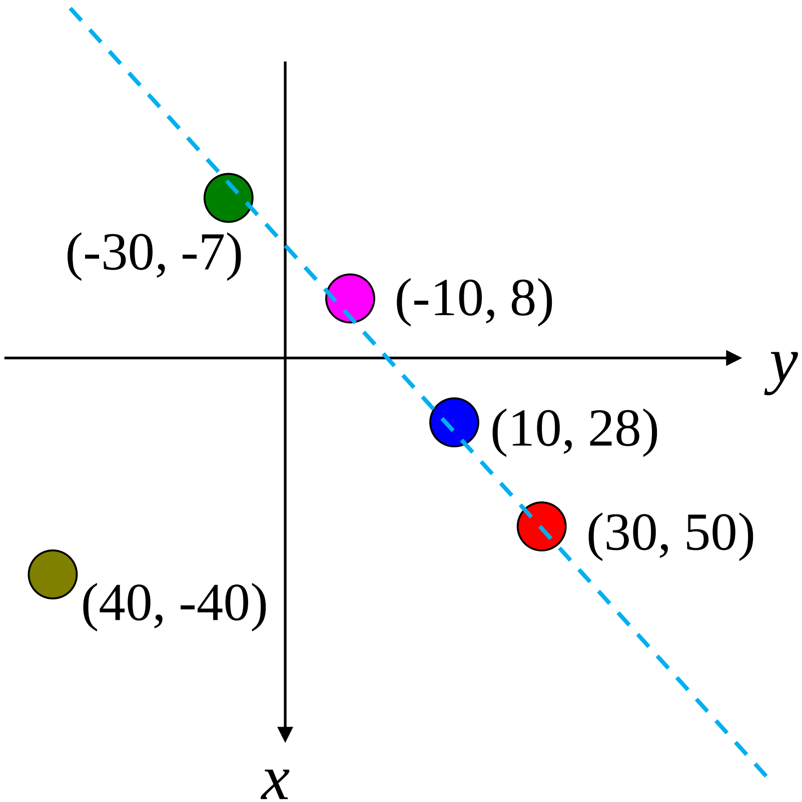
Hough Transform Example

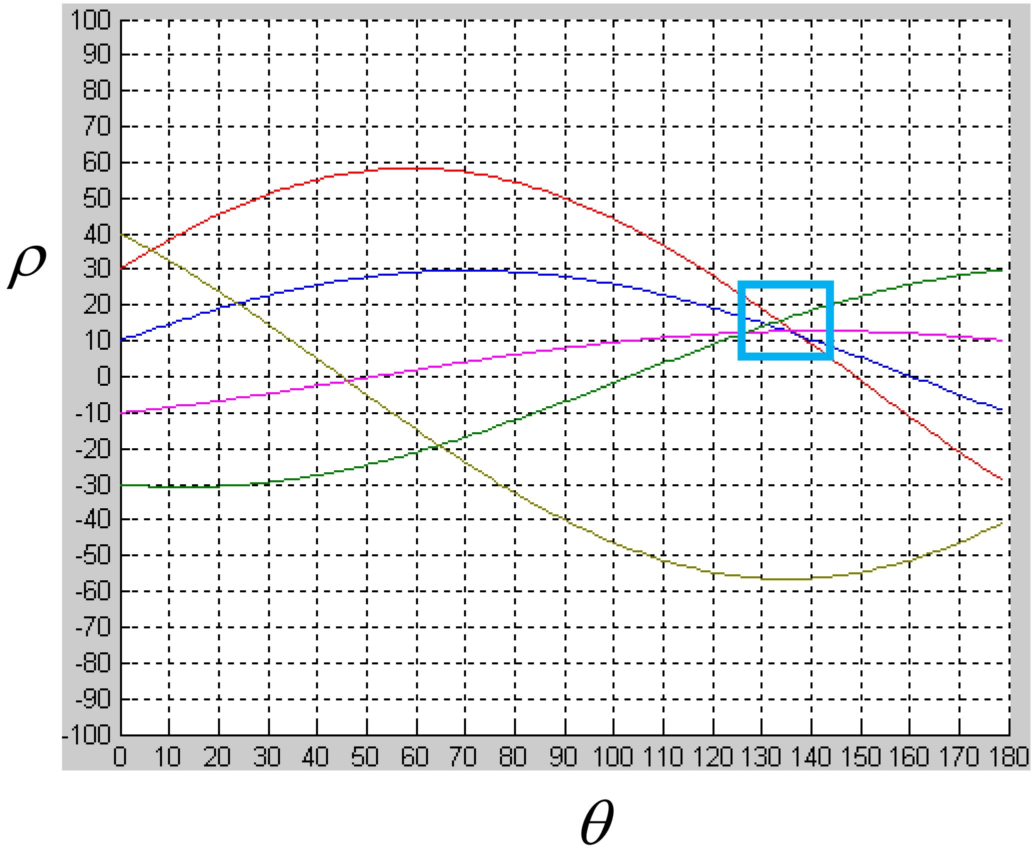
- The idea is to use accumulator cells in the discrete parameter space to record the "score" of each candidate line.
- The algorithm loops through all the edge points in the image. For each edge point, a value is added to all the accumulator cells that represent straight lines that pass through the point.
- Complexity: $O(n\, N_\theta)$





Hough Transform Example

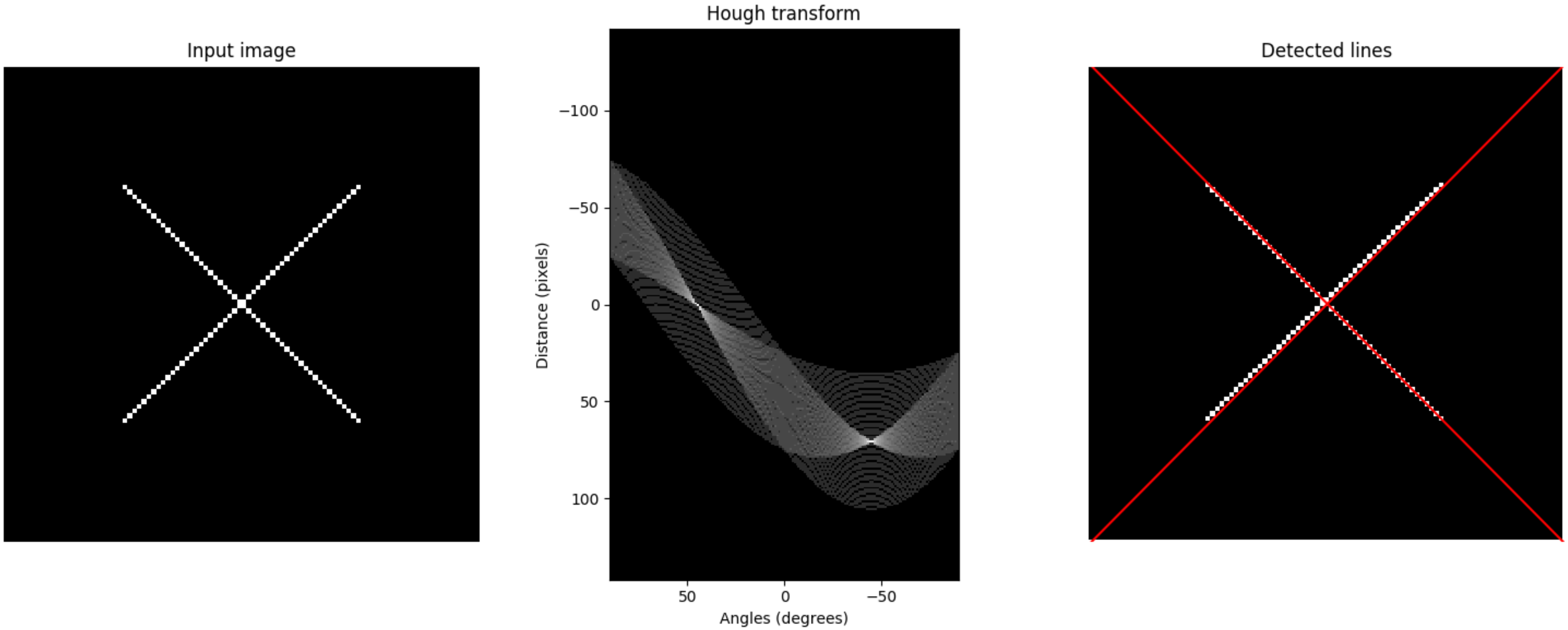




The accumulation cell with the highest total: $\theta = 135°$, $\rho = 15$.

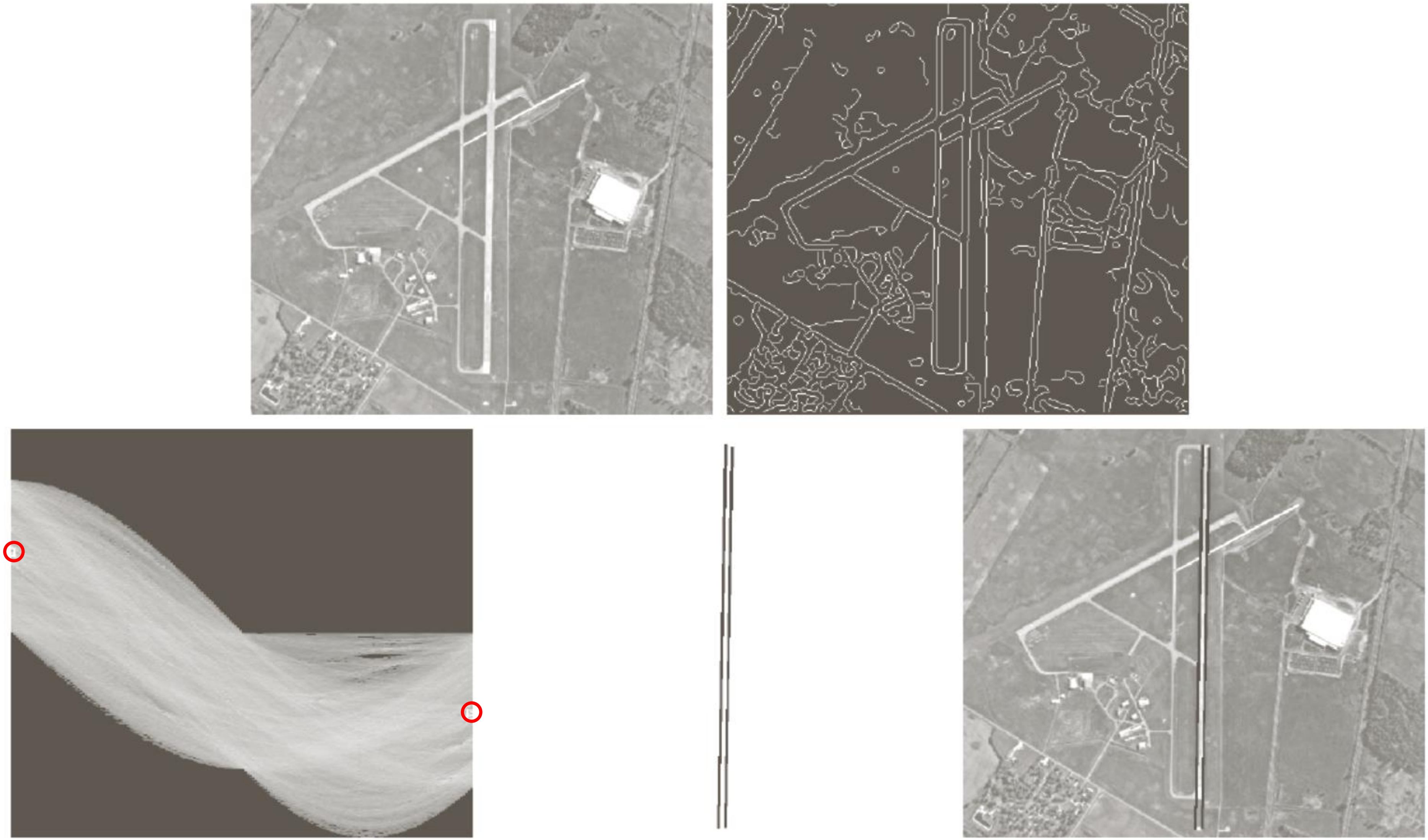
Line equation: $x\cos(135°) + y\sin(135°) = 15$, or $x - y = -15(2)^{1/2}$.

Hough Transform Example



https://scikit-image.org/docs/0.13.x/auto_examples/edges/plot_line_hough_transform.html

Hough Transform Example

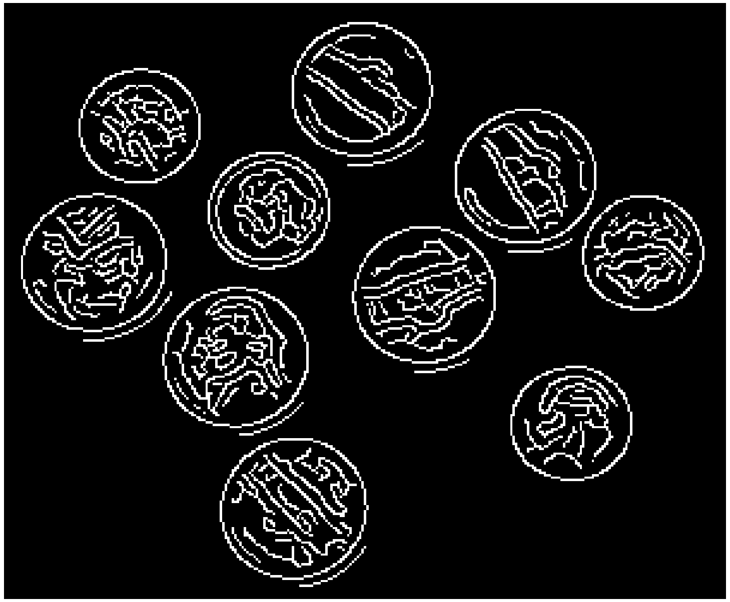


Notes on Hough Transform

- Very often used for line detection.
- Generalization for the detection of more diverse shapes.
  - Parameterized shapes (circles, ellipses, etc.): The parameter space has more dimensions (e.g., 3 for circles and 5 for ellipses).
  - Arbitrary shapes.
- Some disadvantages (However, some preprocessing or post-processing can help):
  - Limitation of parameter resolutions
  - Noisy and zigzagged (not perfectly straight) lines
  - Connection of unrelated line segments

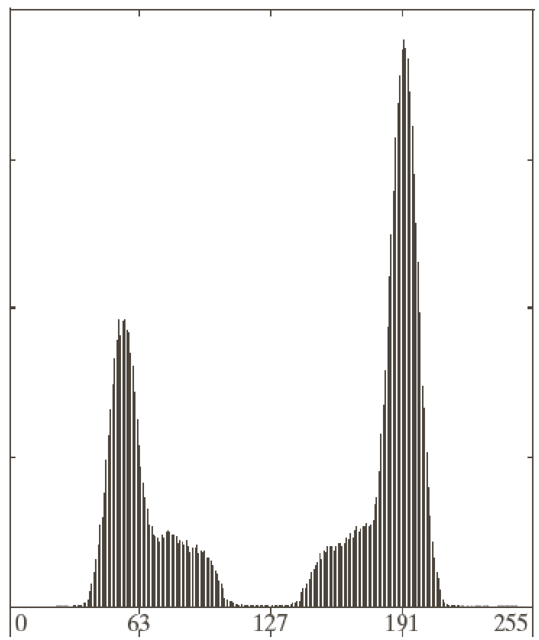
Hough Transform for Circles

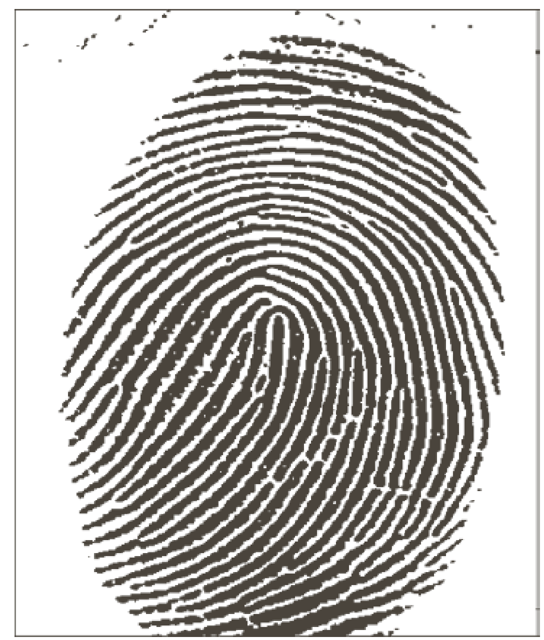
- Three parameters (center coordinates and radius); all bounded.
- Actual process: For each edge point, vary the center coordinates and compute the corresponding radius as the distance from the edge point to the center.



Segmentation by Thresholding

- Use a threshold to separate a whole image into two parts.
- Thresholding is a method of separating pixels into "classes" based on their values.
- The threshold is usually determined from the histogram. For example, for binary segmentation, we can choose the half-way point between the two main peaks.

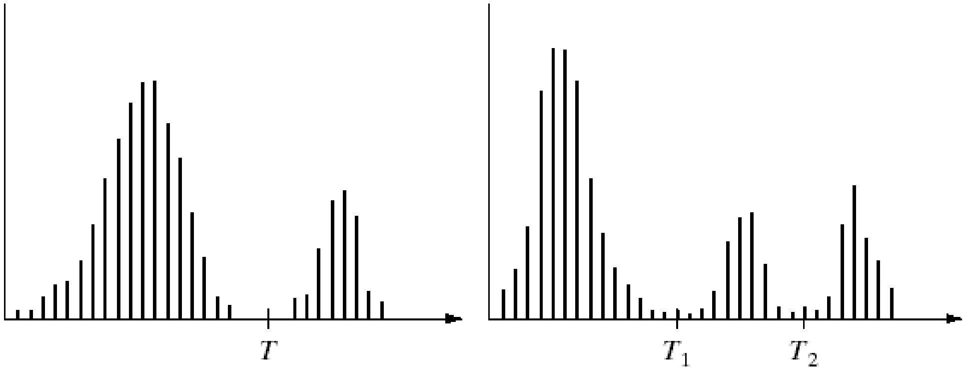




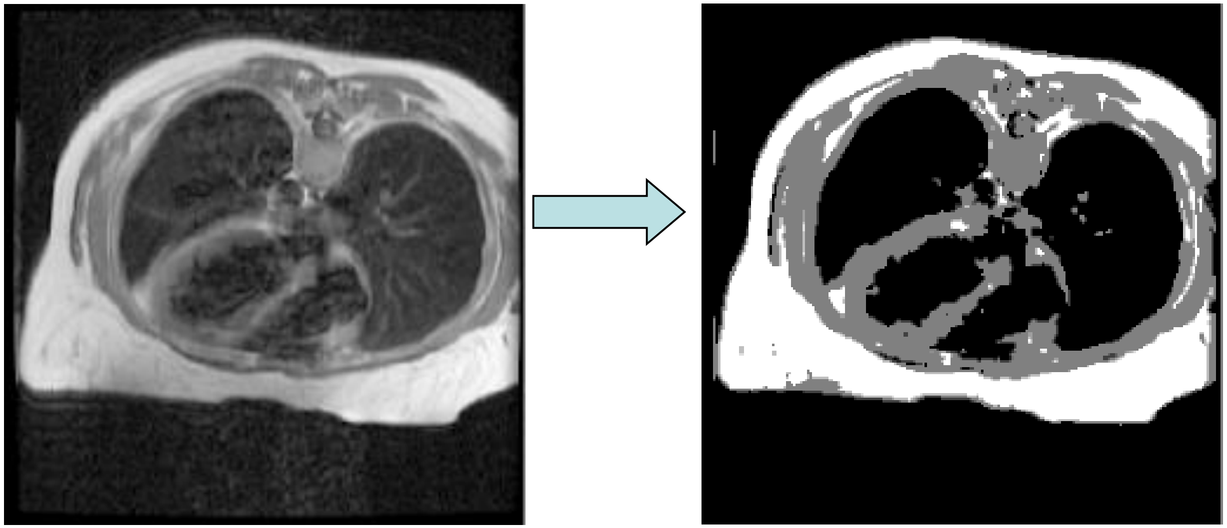
Segmentation by Thresholding

- When appropriate, multiple thresholds can be used to separate the image into more than two parts.

Histograms with two and three peaks:



Example (3 peaks):



Automatic Selection of Thresholds

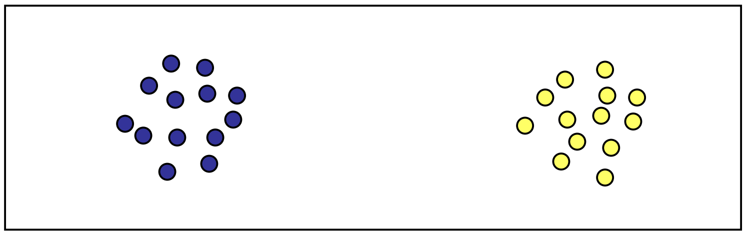
- (Basic Global Algorithm) in textbook
- Start with a threshold T that is the average gray level of the image.
- In each iteration:
    - Compute m_1 and m_2, the average gray levels for those pixels with gray levels above and below T, respectively.
    - Replace T with $(m_1+m_2)/2$.
    - When the change of T between iterations is below some small predefined tolerance, terminate the iteration.

This version is for the single-threshold (two-region) segmentation problem. It can also be extended to more thresholds.

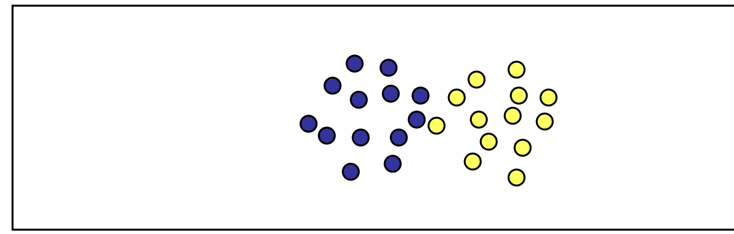
Otsu's Method

The concept of the **separability** between two distributions:

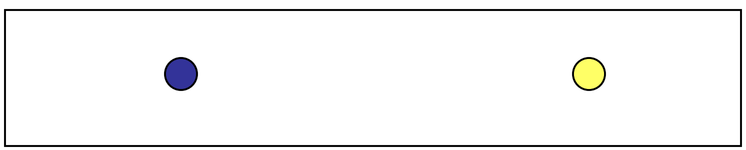
Good separability:



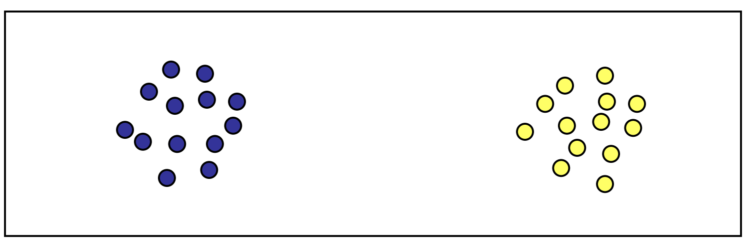
Not-so-good separability



One way to quantify separability: σ_B^2 / σ_G^2



σ_B^2 (between-class variance)



σ_G^2 (global variance)

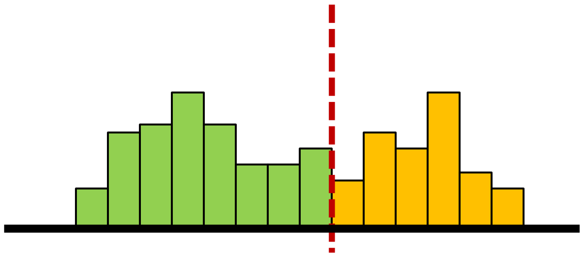
Otsu's Method

Otsu's Method automatically select a threshold to maximize separability:

- For each possible threshold, compute the gray-level separability of the two regions:

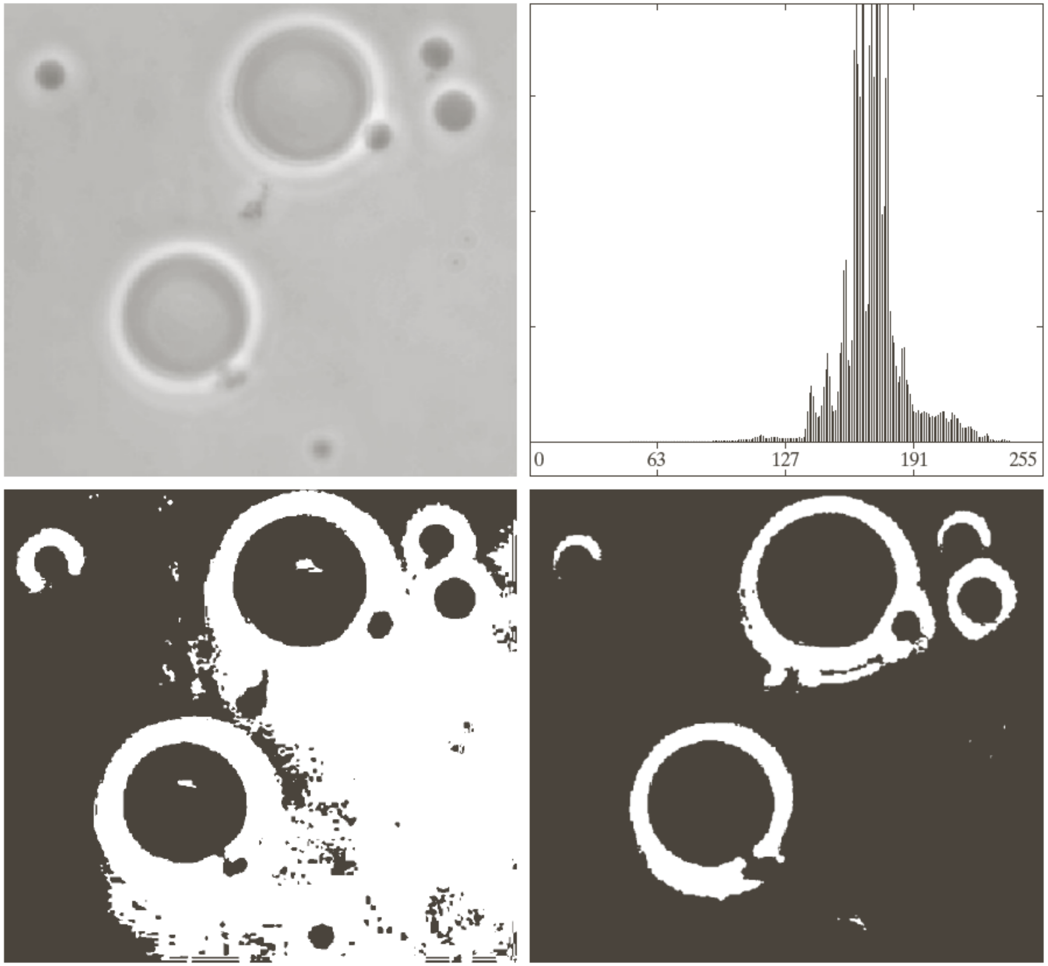
$$\eta(k) = \frac{\sigma_B^2(k)}{\sigma_G^2} = \frac{P_1(k)[m_1(k) - m_G]^2 + P_2(k)[m_2(k) - m_G]^2}{\sigma_G^2}$$

- Select the threshold level (k) that maximizes the separability.



This version is for the single-threshold problem. It can also be extended to more thresholds.

Automatic Selection of Thresholds

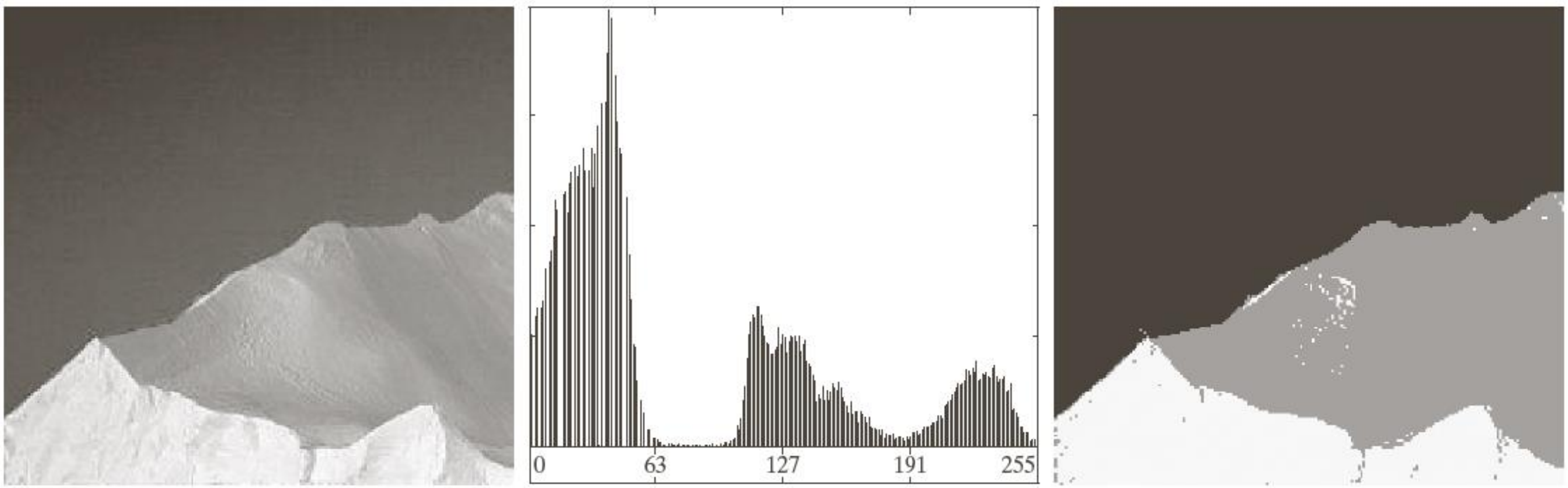


Basic global algorithm

Otsu's method

Automatic Selection of Thresholds

Otsu's method with two thresholds:

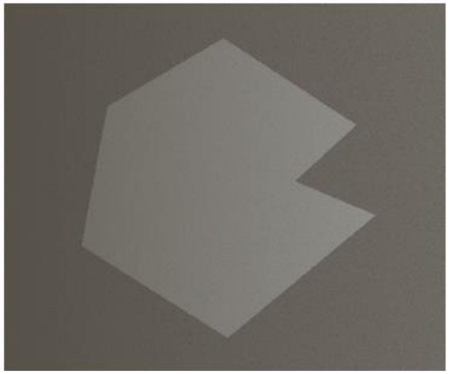


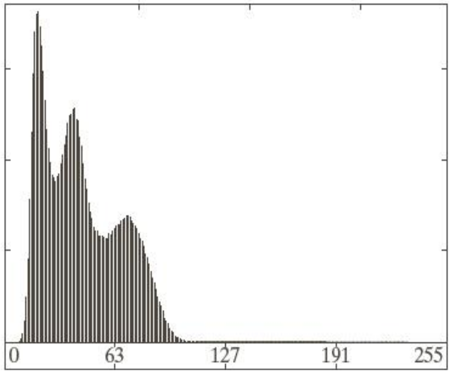
$$\eta(k_1,k_2)=\frac{\sigma_B^2(k_1,k_2)}{\sigma_G^2}=\frac{\sum_{j=1}^{3}P_j(k_1,k_2)[m_j(k_1,k_2)-m_G]^2}{\sigma_G^2}$$

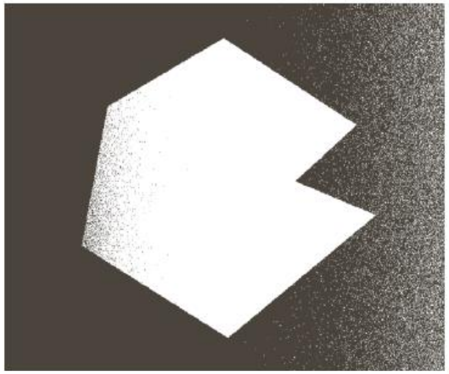
Adaptive Thresholding

In some situations, thresholds need to be selected locally, not globally. The most common cause is the variation of illumination.

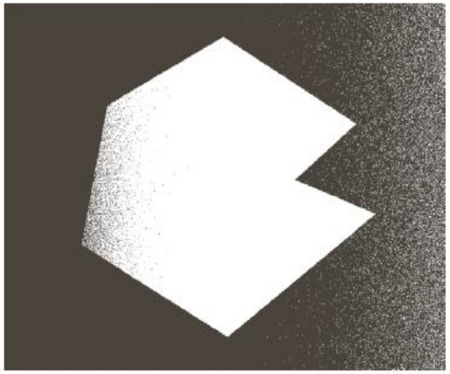
An example of segmentation error that results from global thresholding:







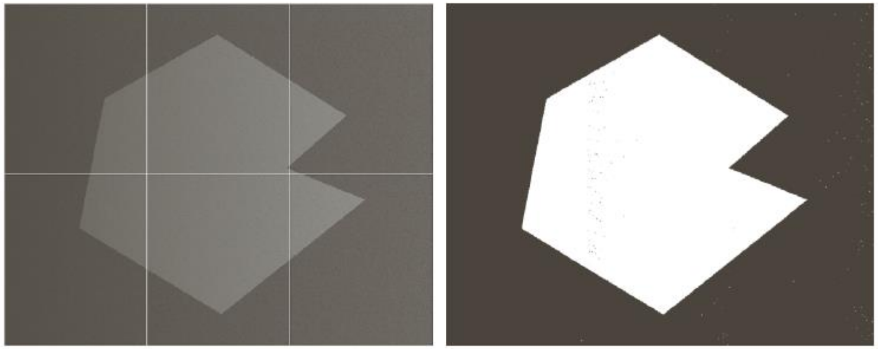
Basic global algorithm

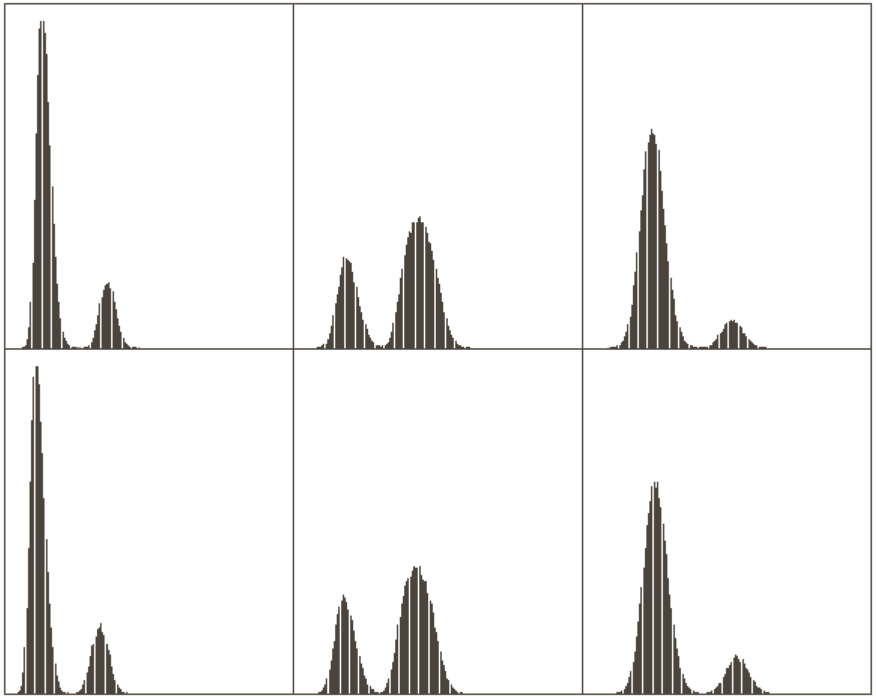


Otsu's method

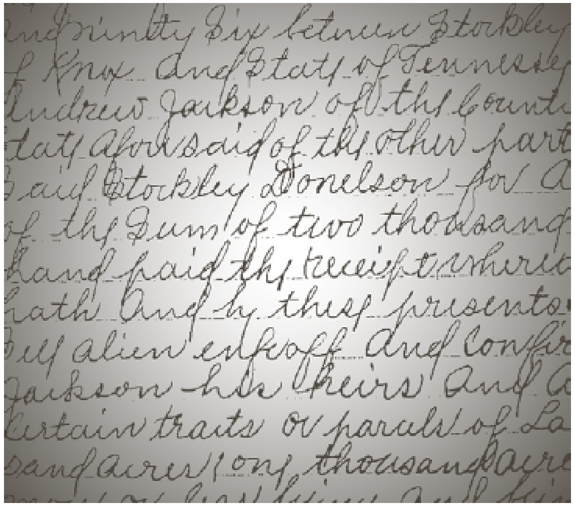
Adaptive Thresholding

The same image, with different thresholds for the 6 regions:

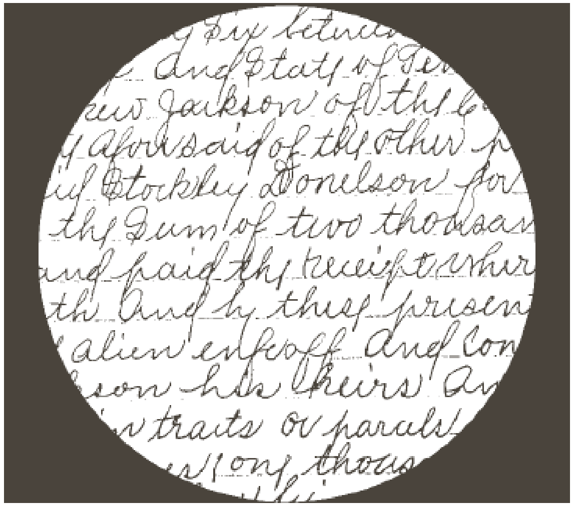




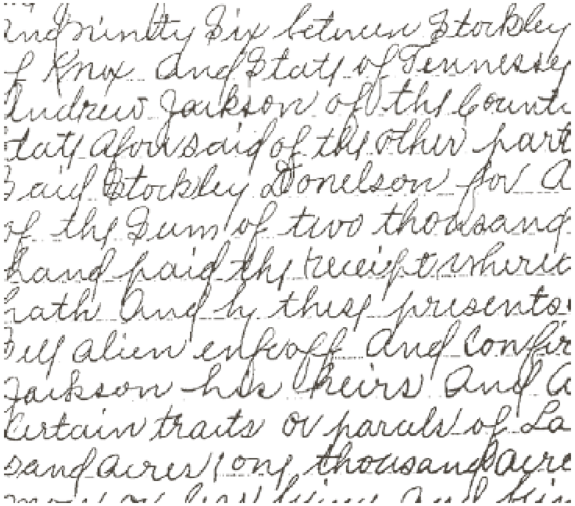
Adaptive Thresholding



global threshold



local threshold



Thresholding by Similarity

Example with skin color segmentation:

Model of Caucasians

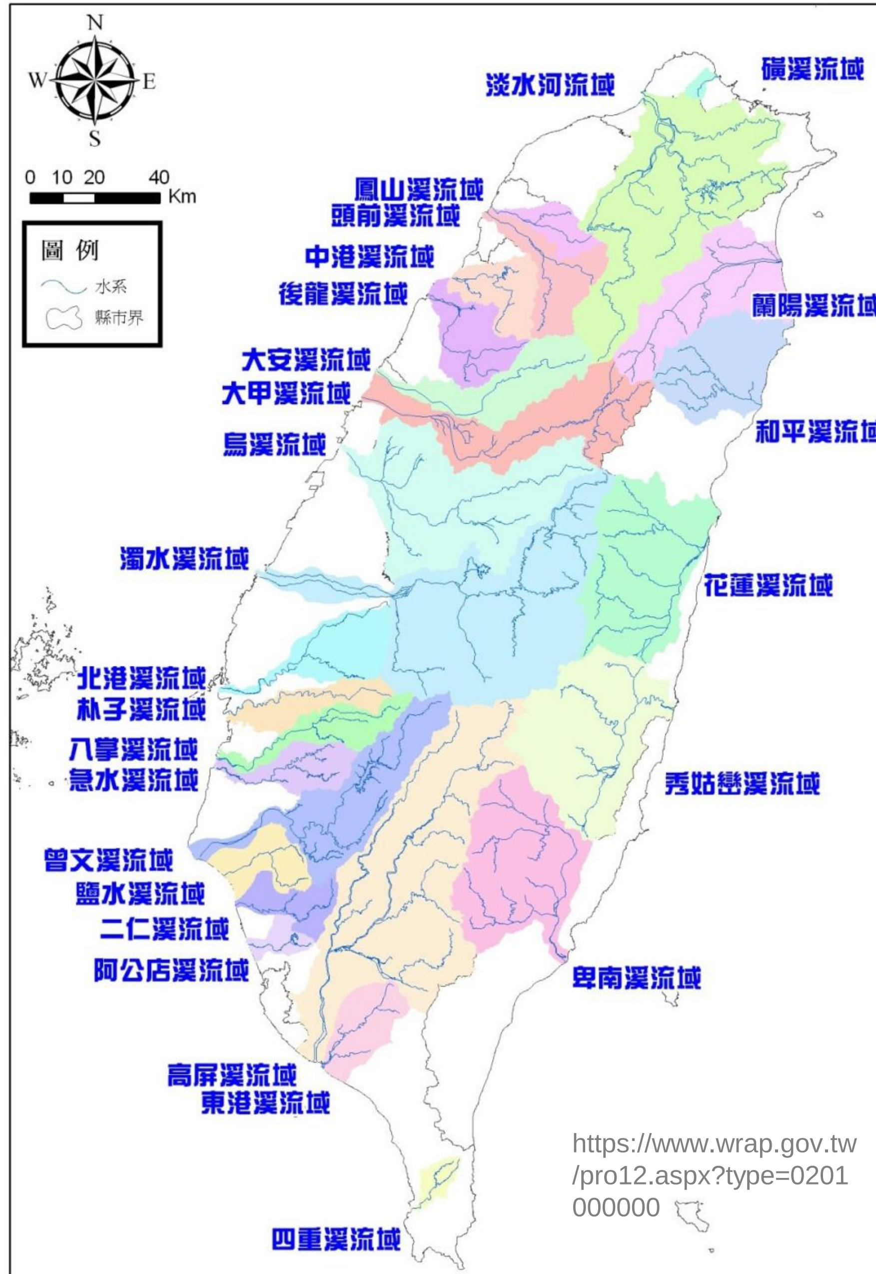
similarity

Model of blacks

thresholding

The Watershed Algorithm

- A watershed is a topographical region where any water within the region will flow toward the same lowest point.
- Watershed lines are the lines that separate the watersheds.

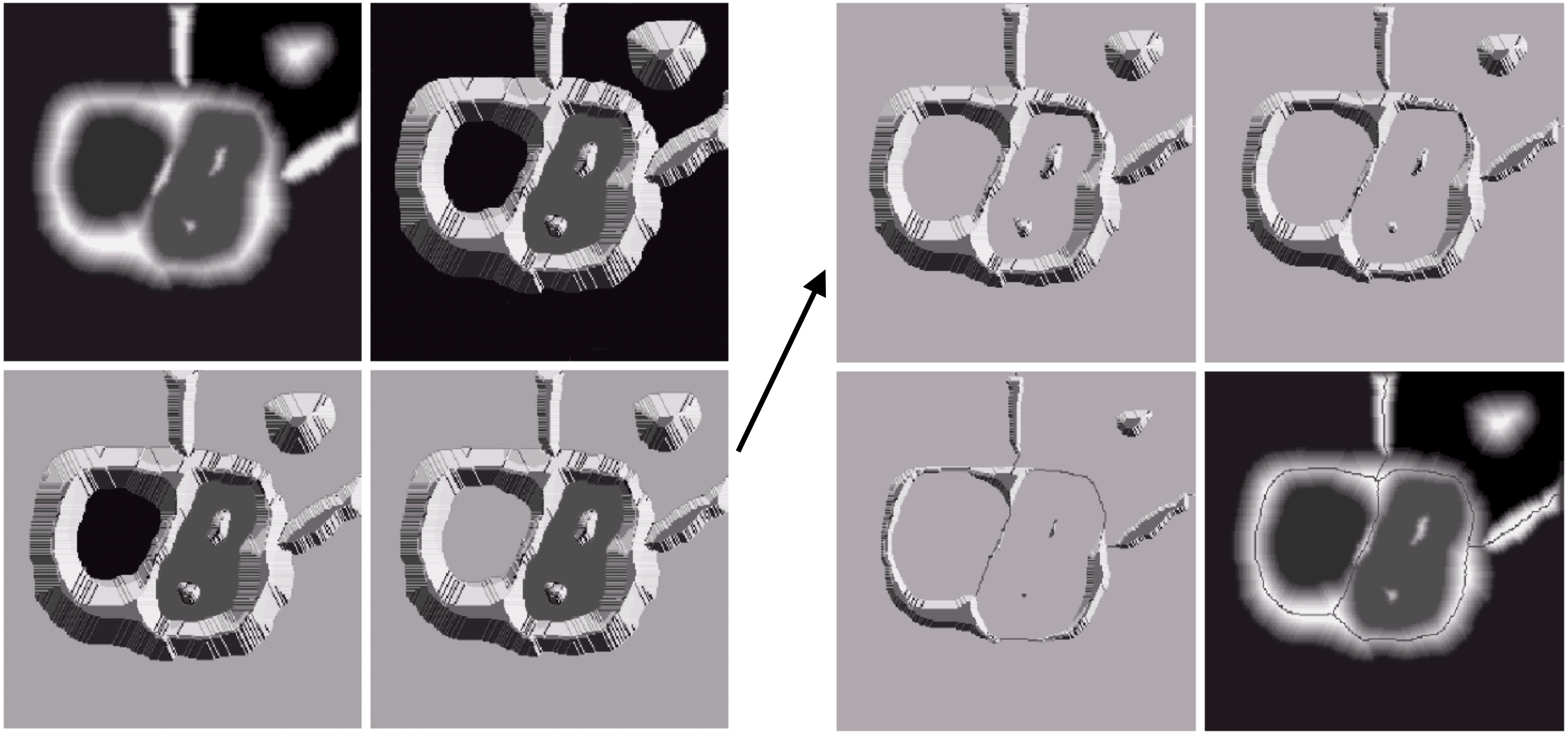


https://www.wrap.gov.tw/pro12.aspx?type=0201000000

The Watershed Algorithm

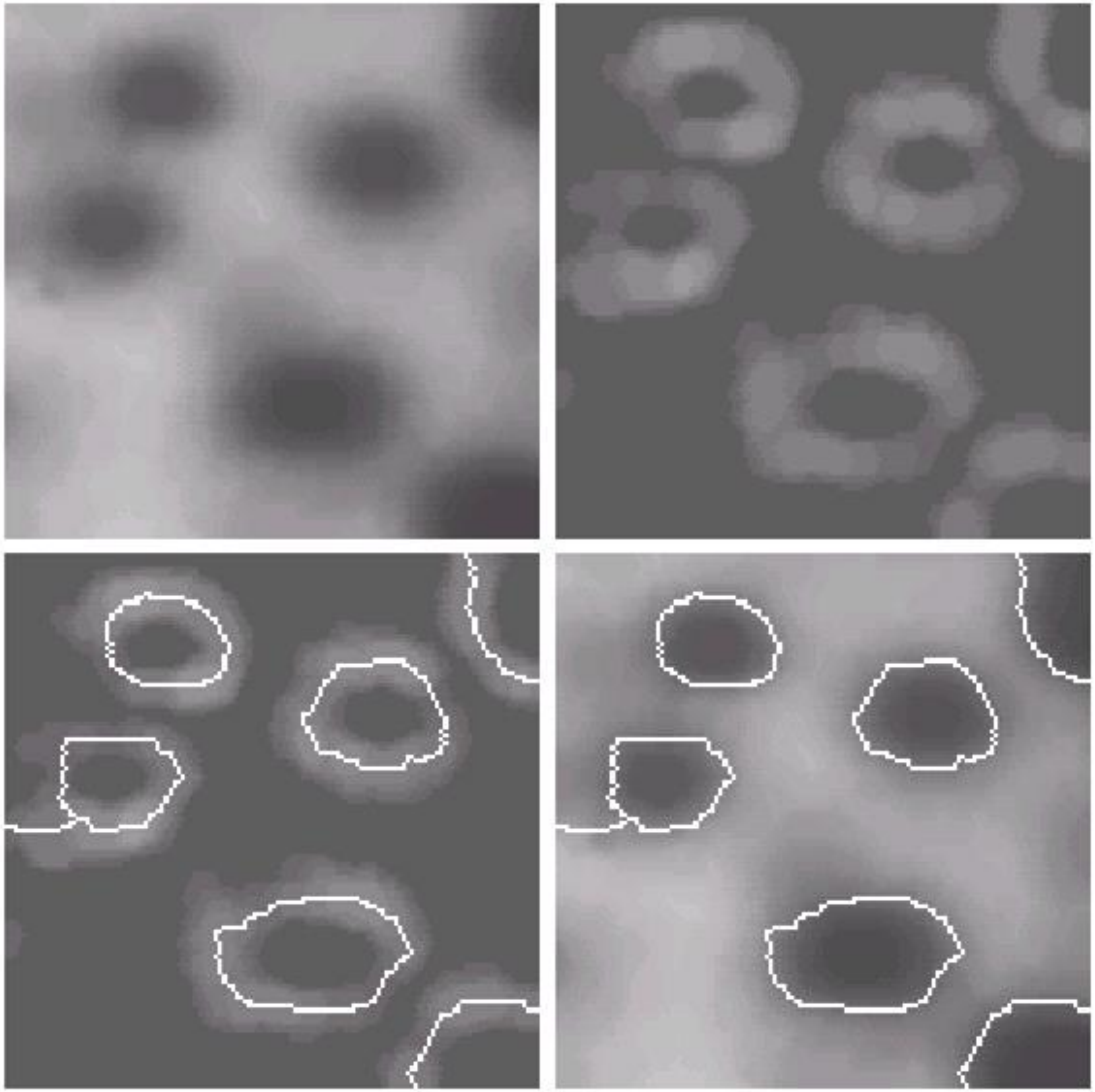
- The algorithm:
  - Treat gray levels in the image (more likely, the "magnitude of gradient" image) as "topographical heights".
  - Let the common water level rises one level at a time, from the lowest toward the highest gray level. The "flooded" pixels in each watershed form a connected component.
  - When two watersheds are about to be merged, build a "dam" to keep the two regions separate.
  - The eventual watersheds are the segmented regions, and the dams are their boundaries.

The Watershed Algorithm



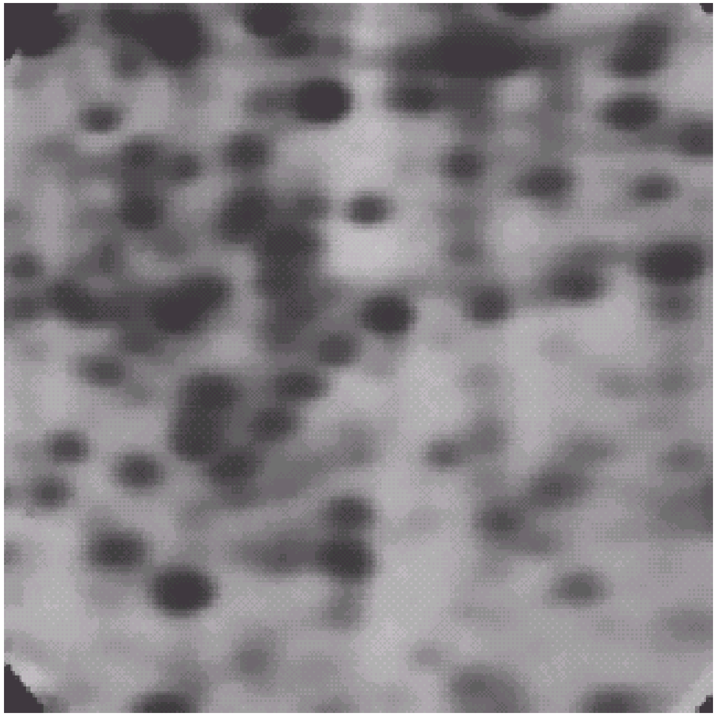
The Watershed Algorithm

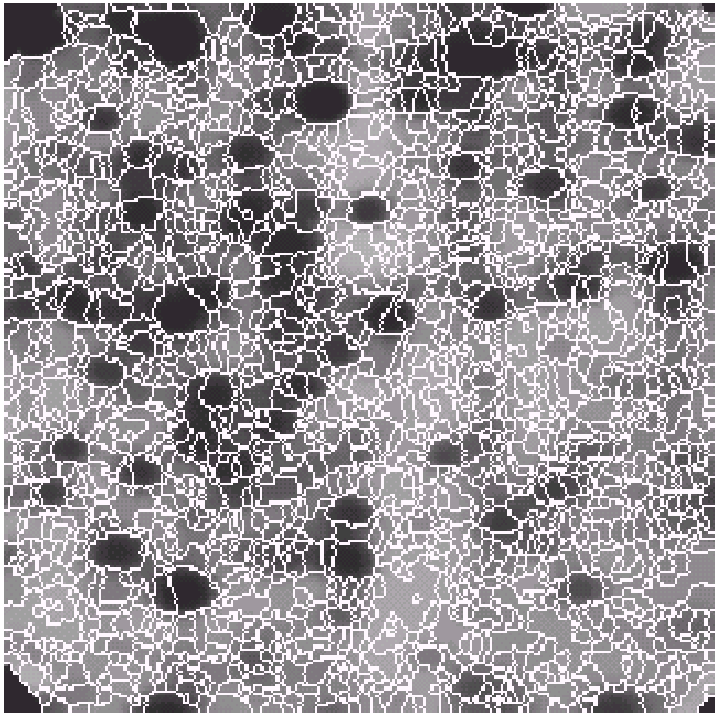
Example:



The Watershed Algorithm

One common problem with the watershed algorithm is over-segmentation (due to noise, etc.):





Possible solutions include preprocessing (smoothing, etc.) and post-processing (region merging, etc.).