Machine Vision

Lecture Set – 04

Regions

Huei-Yung Lin

Robot Vision Lab

Regions and Edges

- Ideally, regions are bounded by closed contours
  - We could “fill” closed contours to obtain regions
  - We could “trace” regions to obtain edges
- Unfortunately, these procedures rarely produce satisfactory results

Region and Edges

- A region in an image is a group of connected pixels with similar properties
- Two approaches to partitioning an image into regions
  - *Region-based segmentation*
    - Value similarity and spatial proximity
    - Two pixels may be assigned to the same region if they have similar intensity values or if they are close to each other
  - *Boundary estimation using edge detection*
    - Edges are found based on differences between values of adjacent pixels
    - Most edge detectors use only intensity characteristics

Region and Edges

- In principle, region segmentation and edge detection should yield identical results
  - Edges may be obtained from regions using boundary-following
  - Regions my be obtained from edges using region-filling
  - (See the previous chapter)

Region Segmentation

- Region segmentation is a partition $R_1, R_2, \ldots, R_n$ such that
  - $\cup R_i = I$
  - $R_i \cap R_j = \varnothing$ if $i \neq j$
  - $P(R_i) = \text{TRUE}$
  - $P(R_i \cup R_j) = \text{FALSE}$, $i \neq j$, R_i adjacent R_j
- P is a function that evaluates similarities of the pixels in the region (or the homogeneity predicate)
- Gray-level to binary image conversion is a simple segmentation (partition to two sets)

3/30/2023

Segmentation Methods

- Thresholding
  - Global knowledge about an image or its part is usually represented by a histogram of image features
  - Also called histogram-based segmentation
- Edge-based
  - Based on the information about edges in the image
  - Use edge detecting operators
- Region-based
  - Construct the regions directly
  - Region-growing methods

Automatic Thresholding

- Automatic thresholding uses knowledge to select a proper threshold value
  - Intensity characteristics of objects
  - Sizes of the objects
  - Fractions of an image occupied by the objects
  - Number of different types of objects appearing in an image
- Automatic thresholding analyzes the gray value distribution in an image
  - Using a histogram of the gray values

Thresholding

- Gray-level thresholding is the simplest segmentation process
  - Objects or image regions are characterized by constant reflectivity of their surfaces
  - A brightness constant or threshold can be determined to segment objects and background
  - Computationally inexpensive and fast
- Thresholding is the transformation of an input image f to an output (segmented) binary image g as follows:

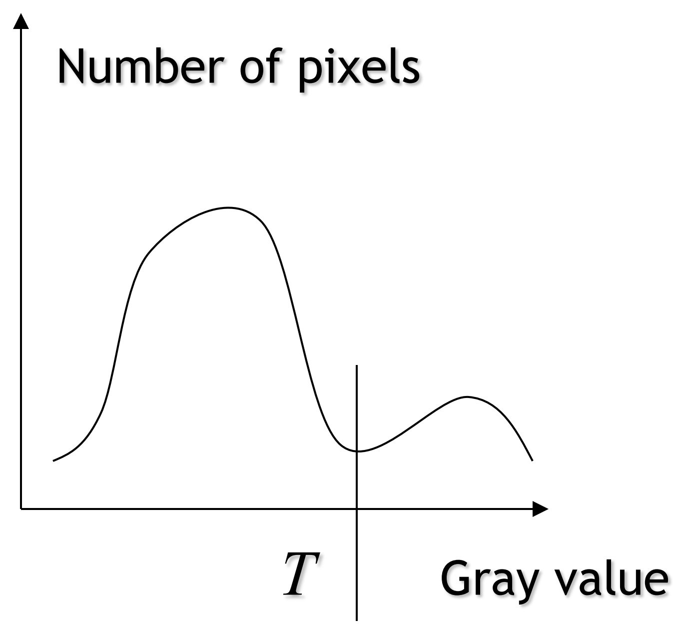
$$
\begin{aligned}
g(i,j) &= 1 \quad \text{for} \quad f(i,j) \geq T \\
&= 0 \quad \text{for} \quad f(i,j) \leq T
\end{aligned}
$$

where T is the threshold, $g(i,j) = 1$ for elements of objects, and $g(i,j) = 0$ for elements of background

Thresholding

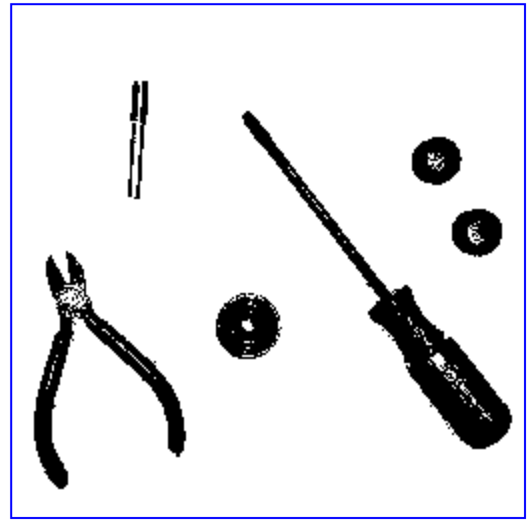
- Algorithm: Basic thresholding
  - Search all the pixels $f(i, j)$ of the image f
  - An image element $g(i, j) = 1$ of the segmented image is an object pixel if $f(i, j) \geq T$, and is a background pixel otherwise

Histogram



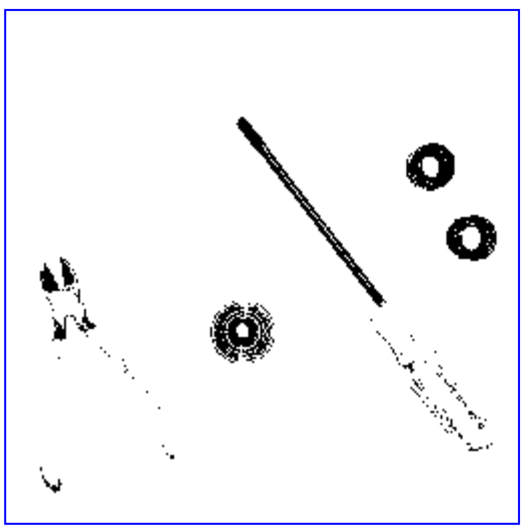
Example

Original image



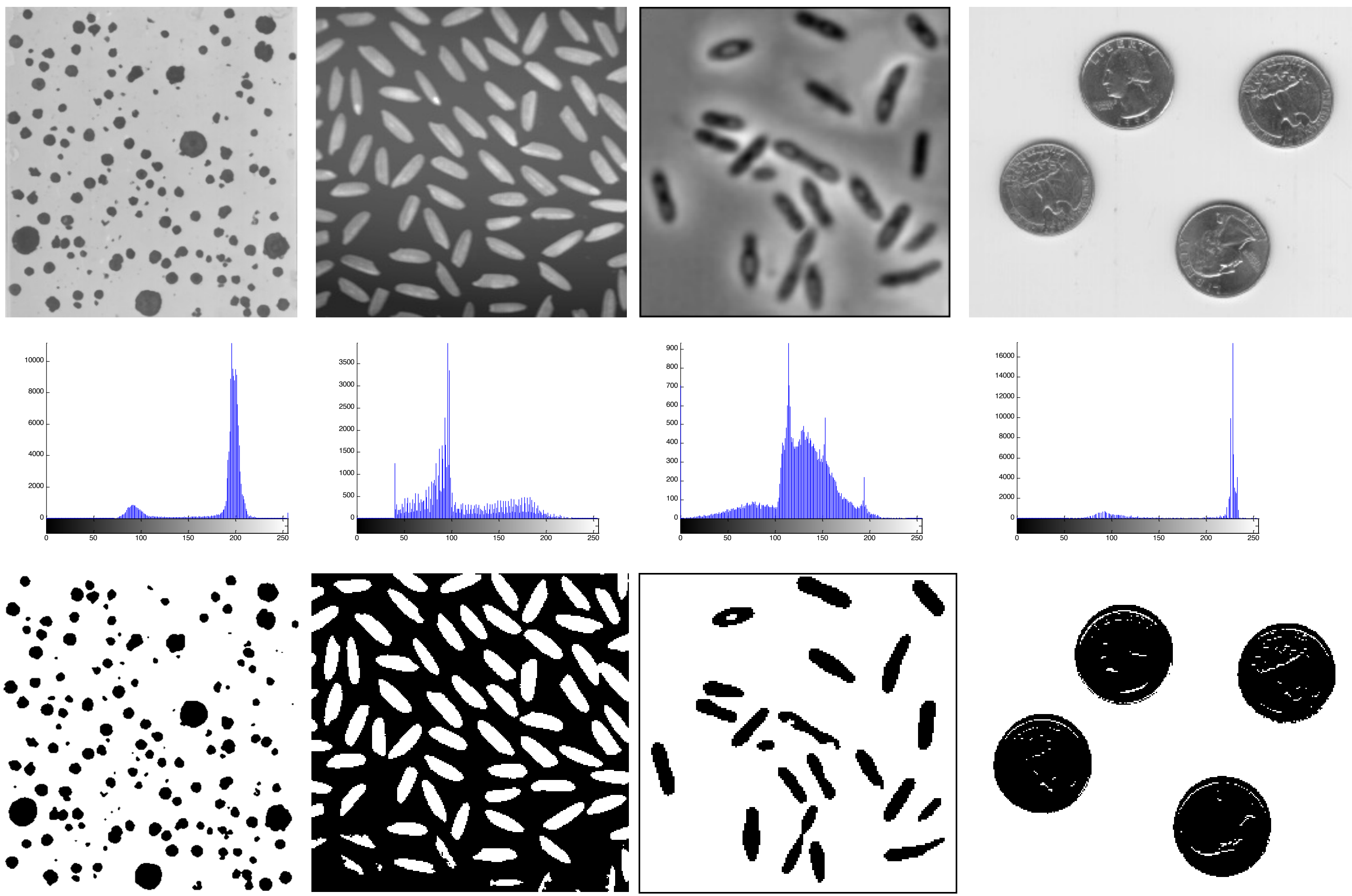
Threshold segmentation

Threshold too low



Threshold too high

Example



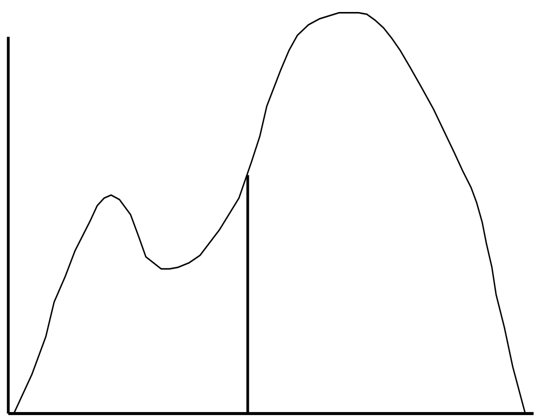
10000
8000
6000
4000
2000
0
0 50 100 150 200 250
3500
3000
2500
2000
1500
1000
500
0
0 50 100 150 200 250
900
800
700
600
500
400
300
200
100
0
0 50 100 150 200 250
16000
14000
12000
10000
8000
6000
4000
2000
0
0 50 100 150 200 250

How to Find The Threshold Value?

- Manual
  - User defines a threshold
- P-Tile
- Mode
- Peakiness detection
- Iterative algorithm
- Other automatic methods

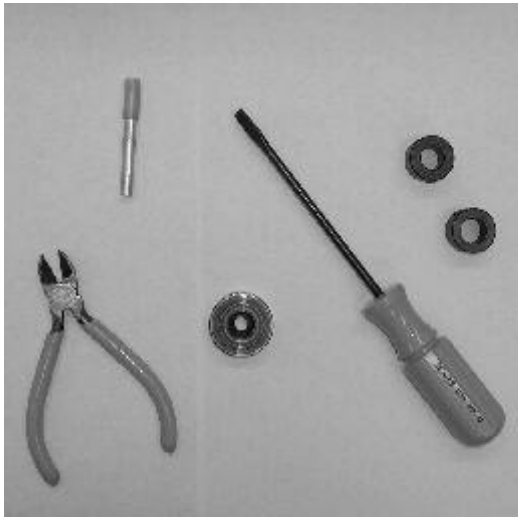
P-Tile Thresholding

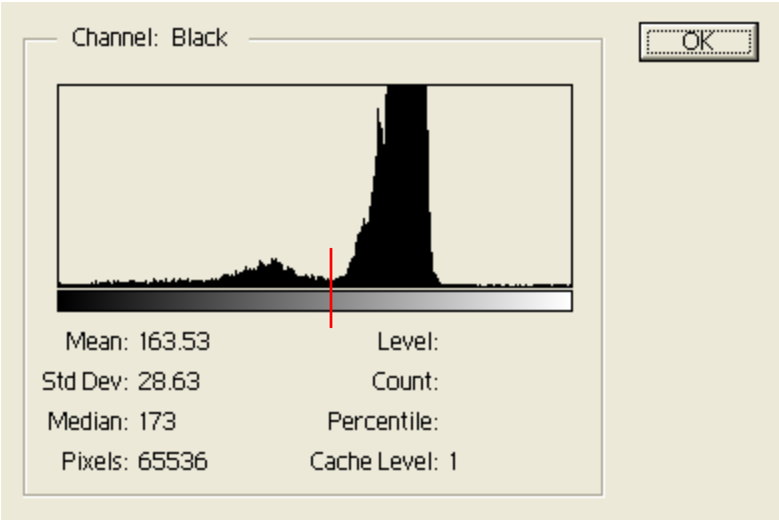
- If we know the proportion of the image that is object (i.e., the size of the object is known!), threshold the image to select this proportion of pixels
- A printed text sheet may be an example if we know that characters of the text cover $1/p$ of the sheet area
- Choose a threshold T (based on the image histogram) such that $1/p$ of the image area has gray values less than T and the rest has gray values larger than T



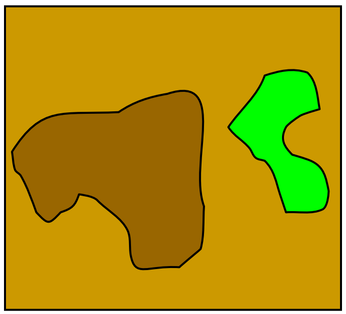
Mode

- Based on histogram shape analysis
- Threshold at the minimum between the histogram's peaks

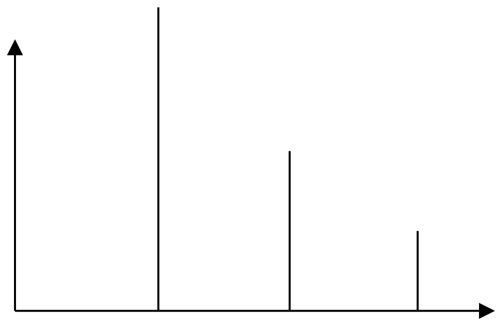




Example

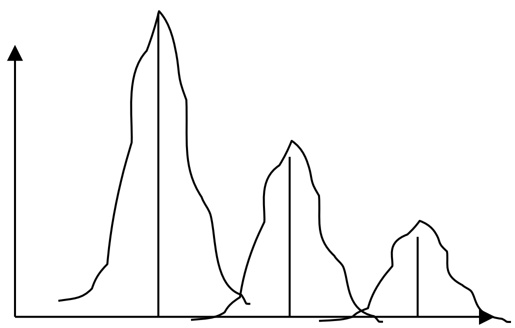


Ideal histogram:



$m_1 \quad m_2 \quad m_3$

Add noise:

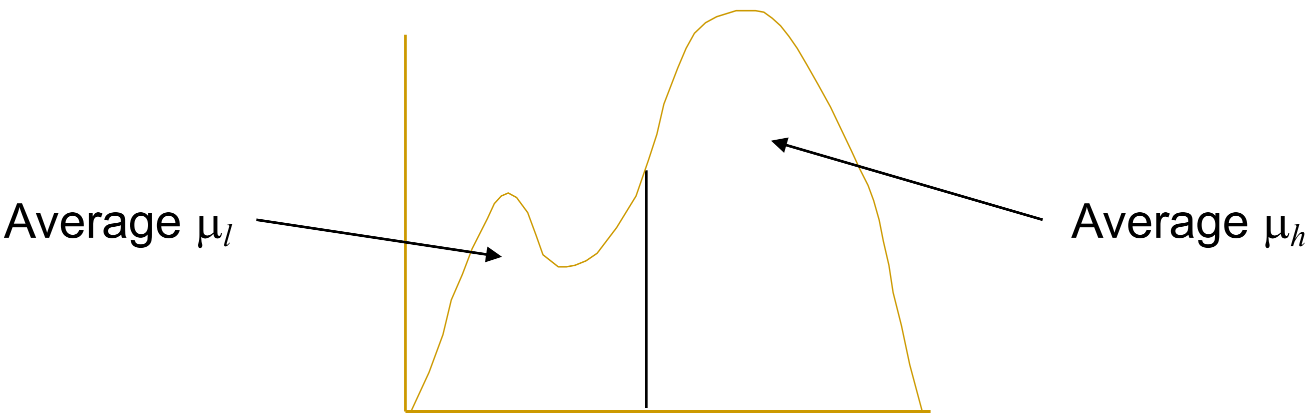


$m_1 \quad m_2 \quad m_3$

The valleys are good places for thresholding to separate regions

Finding the Peaks and Valleys

- Find a threshold T such that

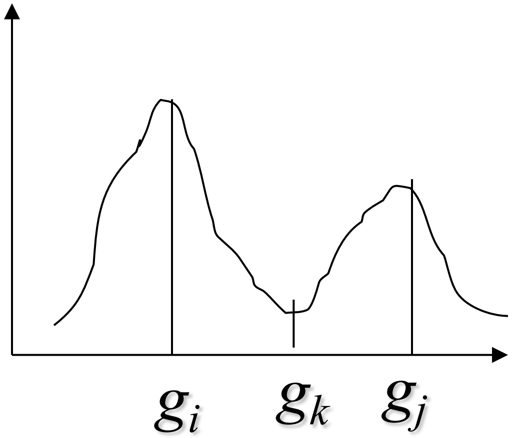


$$T = \frac{\mu_l + \mu_h}{2}$$

- Methods:
  - Peakiness detection
  - Iterative threshold selection
  - Adaptive thresholding
  - Variable thresholding
  - Double thresholding

Peakiness Detection Algorithm

- Find the two highest local maxima in the histogram that are *at some minimum distance apart*
  - Suppose these occur at gray values g_i and g_j
- Find the lowest point g_k in the histogram H between g_i and g_j
- Find the peakiness, defined as $\min(H(g_i),H(g_j))/H(g_k)$
- Use the combination (g_i, g_j, g_k) with highest peakiness to threshold the image
- The value g_k is a good threshold to separate objects corresponding to g_i and g_j



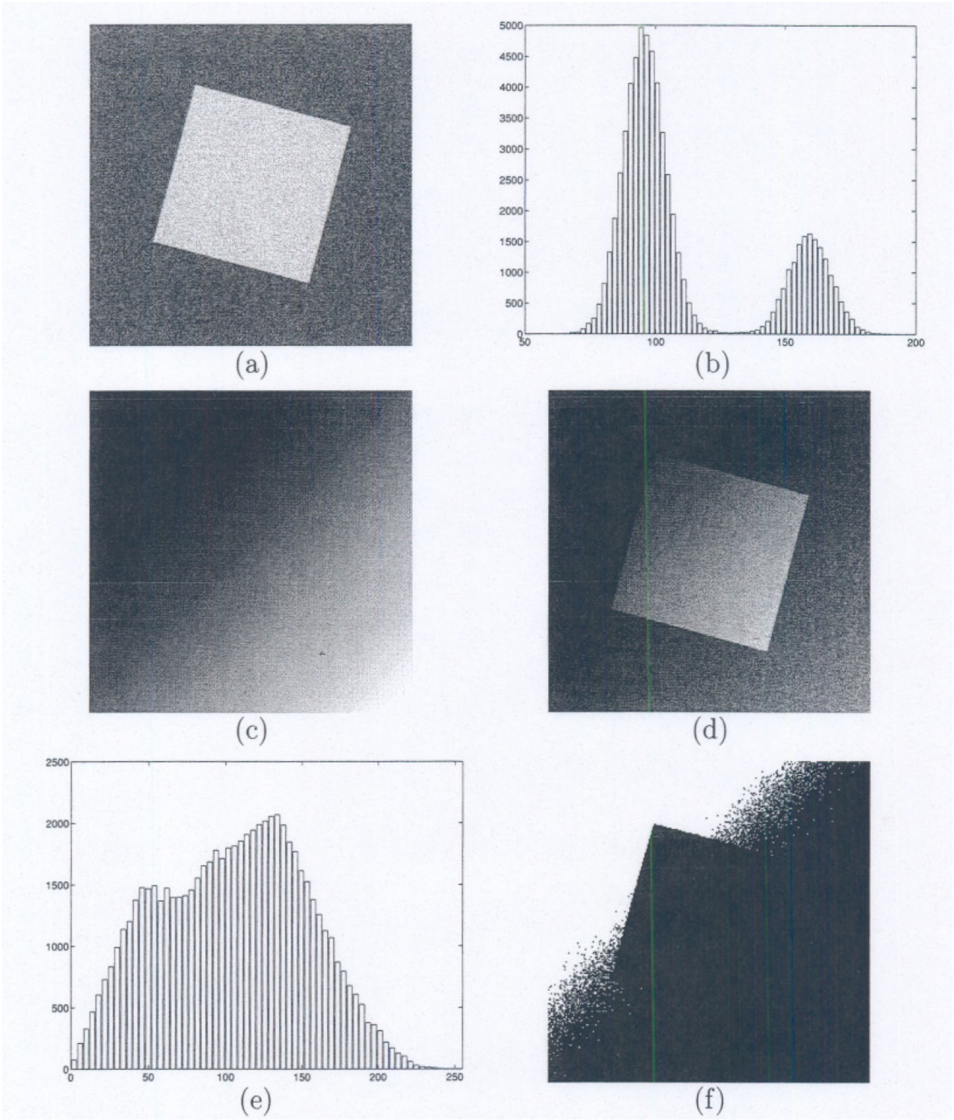
Iterative Threshold Selection

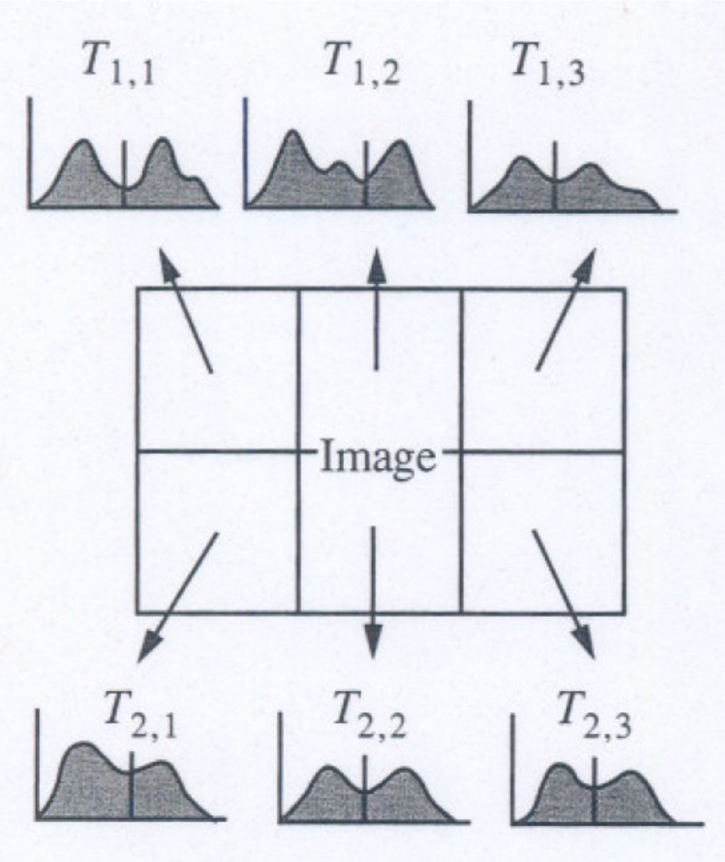
- Iterative threshold selection
  - It starts with an approximate threshold and then successively refines this estimate
- Algorithm: Iterative Threshold Selection
  - Select an initial estimate of the threshold T
    A good *initial value* is the *average intensity of the image*
  - Partition image into two groups, R_1 and R_2, using threshold T
  - Calculate mean gray values μ_1 and μ_2 of the partition R_1 and R_2
  - Select a new threshold $T = (\mu_1+\mu_2)/2$
  - Repeat steps 2-4 until the mean value μ_1 and μ_2 converge

Adaptive Thresholding

- Automatic thresholding scheme may fail if the illumination in a scene is uneven
- The same threshold value may not be usable throughout the complete image
- Solution:
  - Partition the image into subimages
  - Select (different) threshold for each subimage based on its histogram
  - The final segmentation is the union of the regions of those subimages

Adaptive Thresholding

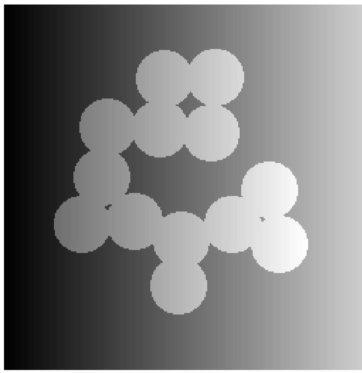


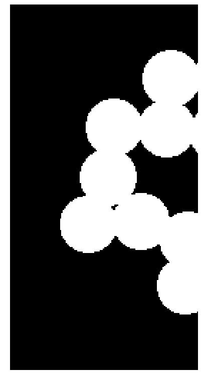


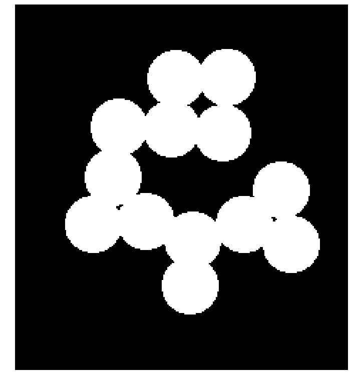
Variable Thresholding

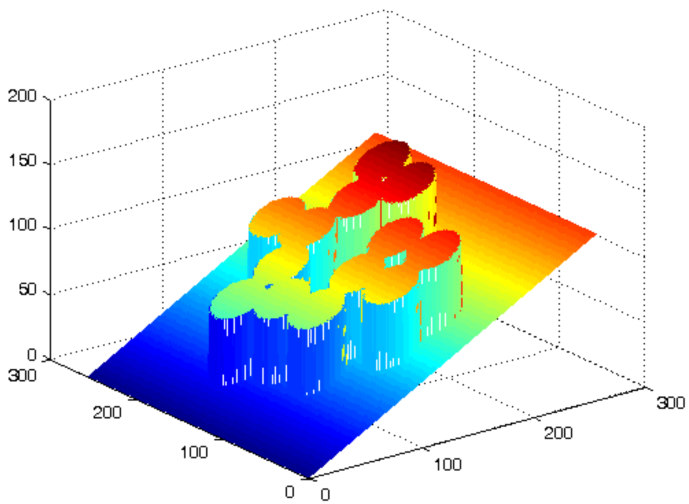
- Also used to deal with uneven illumination
- Approximate the (new) intensity values of the image by a simple function such as a plane
  - The function fit is determined in large part by the gray value of the background
  - Histogram and threshold are then selected based on the fitted function
- Also called background normalization

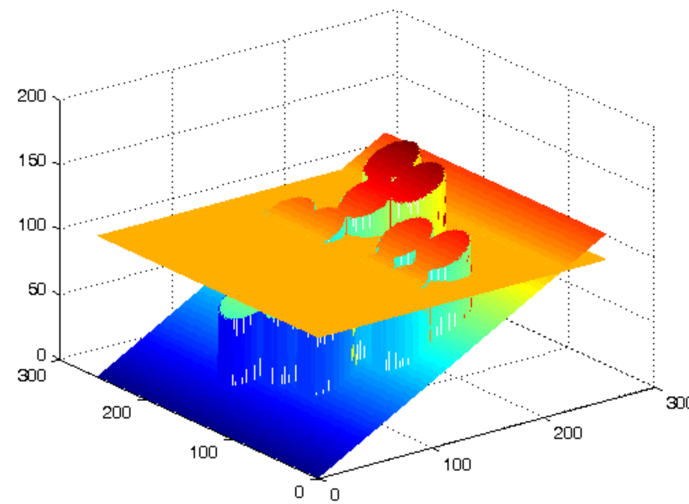
Example





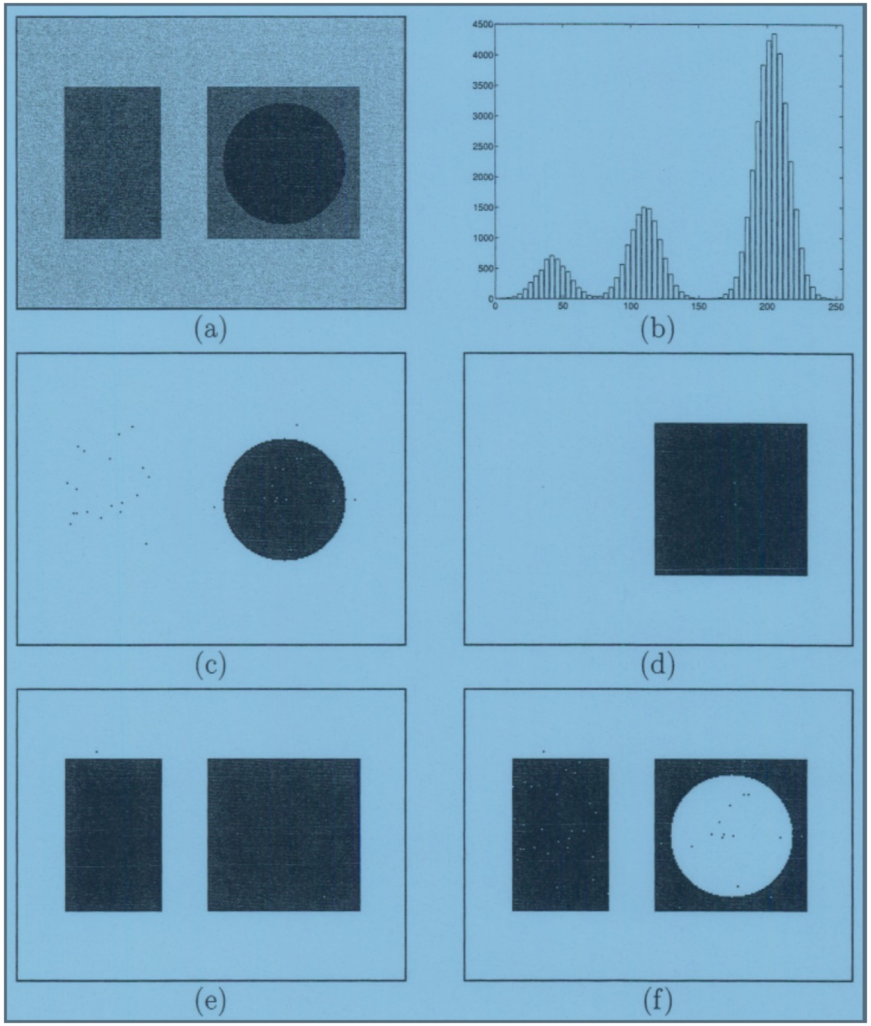



200
150
100
50
0
300
200
100
0
0
100
200
300


200
150
100
50
0
300
200
100
0
0
100
200
300

Double Thresholding

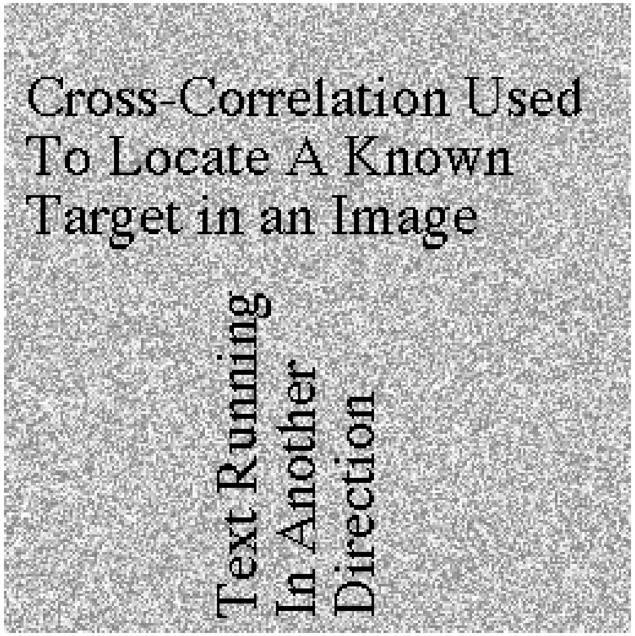
- To segment an object region which is inside another object region
  - Use a conservative threshold to obtain the core of the object and then grow the object regions
  - Use the second threshold to segment the object and background regions

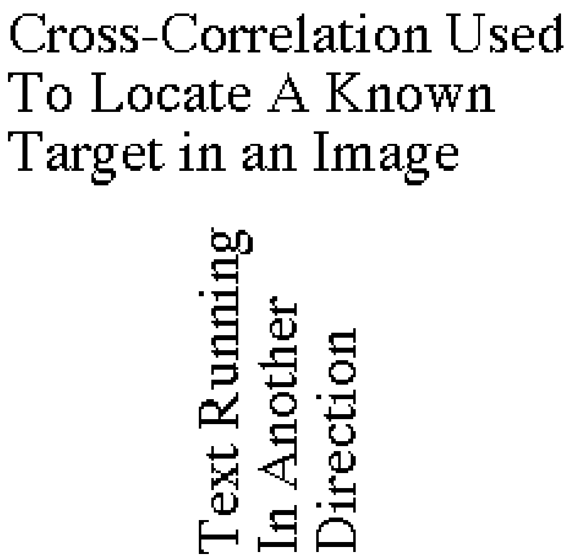


Double Thresholding

- Double Thresholding for Region Growing
  - Select two thresholds T_1 and T_2
  - Partition the image into three regions:
    - R_1: all pixels with gray values below T_1
    - R_2 : all pixels with gray values between T_1 and T_2
    - R_3 : all pixels with gray values above T_2
  - Visit each pixel assigned to region R_2
    If the pixel has a neighbor in region R_1, then reassign the pixel to region R_1
  - Repeat step 3 till no pixels are reassigned
  - Reassign any pixel left in region R_2 to region R_3

Example – Applications



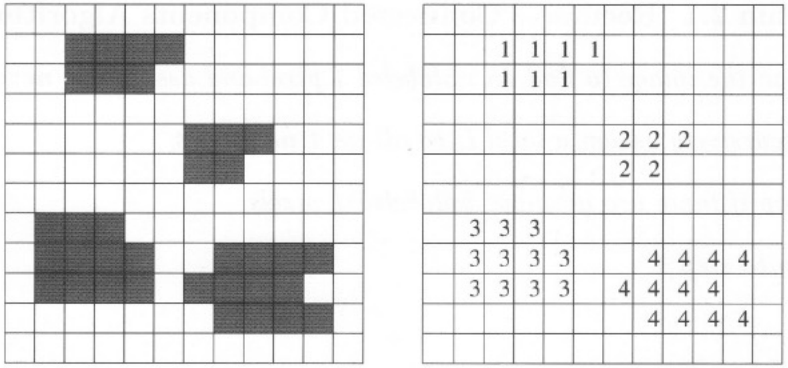


Limitations of Histogram Methods

- The most basic limitations
  - Use global information
  - Ignore spatial relationships among pixels
    - Use histogram, which contains no spatial information
    - Different patterns may have the same histogram!
- Other limitations
  - Constant illumination

Region Representation

- Classification of region representation
  - *Array representation*
  - *Hierarchical representation*
  - *Symbolic representation*
- Array representation
  - Use a same size array as the original image with entries indicating the region to which a pixel belongs
  - If an element of the array has value α, then the corresponding pixel in the image belongs to region α
  - For binary image
    - 0 : background
    - 1 : foreground



Hierarchical Representation

- An image can be represented at different resolutions
  - *Lower resolution*: small array size, data loss
  - *Lower resolution*: less memory and computing requirements
- Hierarchical representation of images allows representation at multiple resolutions
  - Compute properties of image first at low resolution
  - Perform additional computations over a selected area at higher resolution
- Two different representations
  - "Pyramids" and "Quad trees"

Pyramids

- An image pyramid is a collection of representations of an image – the name comes form a visual analogy
  - Each layer of the pyramid is half the width and half the height of the previous layer (referred to as *coarse scale* version)
  - A pyramid is made by stacking the layers on top of each other
  - Pyramids are most convenient if the image dimension is power of two
- Common image pyramids
  - Gaussian pyramid (Gaussian $*$ Gaussian = Gaussian)
  - Laplacian pyramid

4/6/2023

- HW #2 due TODAY
- HW #3 will be given next Thursday (one week late)
- Midterm on 4/20

The Gaussian Pyramid

- In a Gaussian pyramid, each layer is smoothed by a *symmetric Gaussian kernel* and resampled to get the next layer
- Let $S\downarrow$ denote downsample of an image, $P(I)_n$ denote the nth level of a pyramid $P(I)$, then

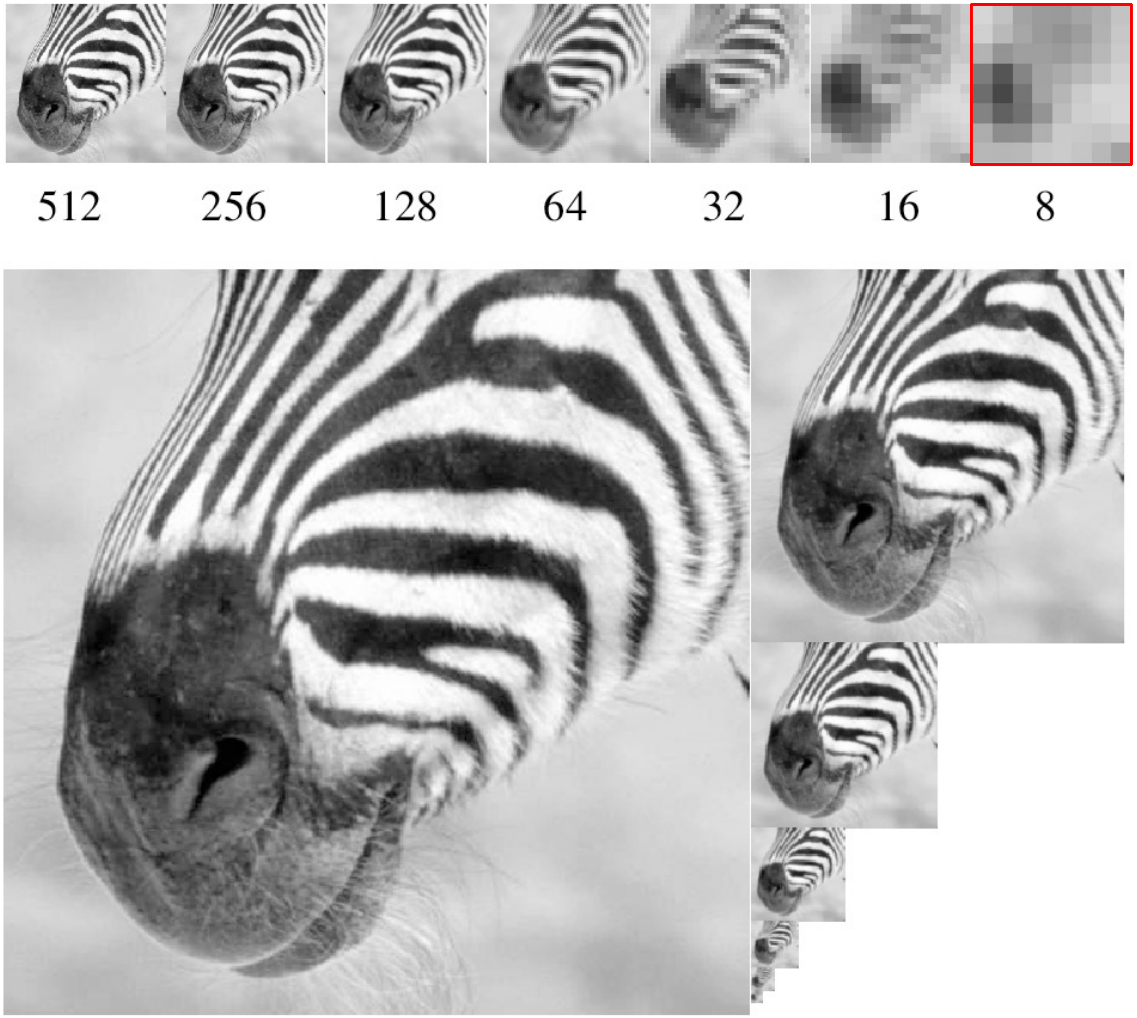
$$P_{Gaussian}(I)_{n+1} = S\downarrow(G_\sigma * P_{Gaussian}(I)_n) = S\downarrow G_\sigma(P_{Gaussian}(I)_n)$$

$$P_{Gaussian}(I)_1 = I$$

The Gaussian Pyramid

- The Gaussian pyramid is an example of image analysis by *a bank of filters* (smoothing filters)
  - It is a highly redundant representation because each layer is a low pass filtered version of the previous layer – we are representing the lowest spatial frequencies many times
  - A layer of the Gaussian is a prediction of the appearance of the next finer scale layer

The Gaussian Pyramid



The Laplacian Pyramid

- *A coarse layer of the Gaussian pyramid predicts the appearance of the next finer layer*
  - The coarsest scale layer of a Laplacian pyramid is the same as the coarsest scale of a Gaussian pyramid
  - Each of the finer scale layers of a L.P. is a difference between a layer of the G.P. and a prediction obtained by upsampling the next coarsest layer of the G.P.

$$P_{Laplacian}(I)_m = P_{Gaussian}(I)_m$$

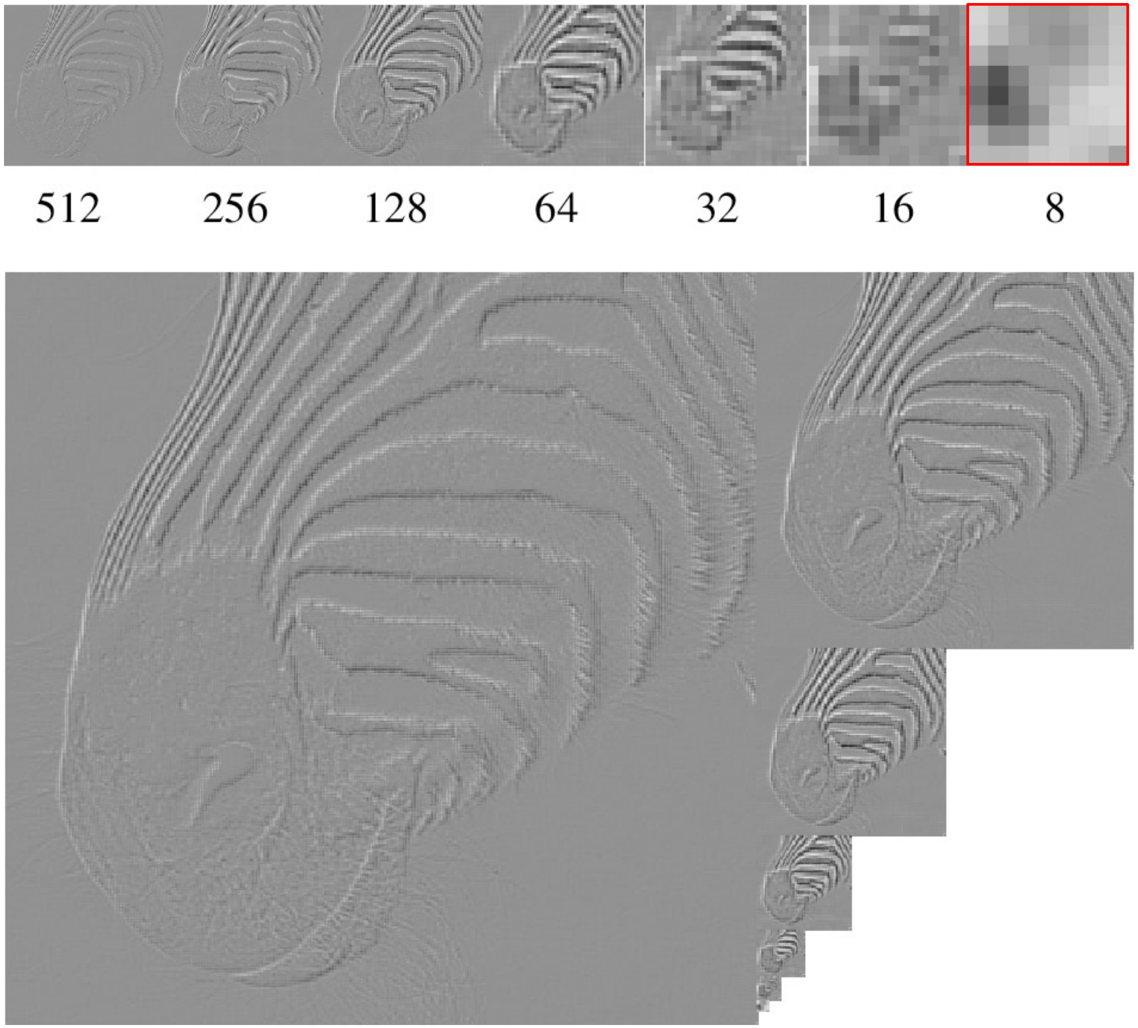
$$P_{Laplacian}(I)_k = P_{Gaussian}(I)_k - S\uparrow(P_{Gaussian}(I)_{k+1})$$

The Laplacian Pyramid

- Each layer of the Laplacian pyramid can be thought of as the response of a *band-pass filter*
  - Low spatial frequencies being subtracted
- The L.P. can be used as an effective *image compression* scheme
  - Different level of the pyramid represent different spatial frequencies

The Laplacian Pyramid

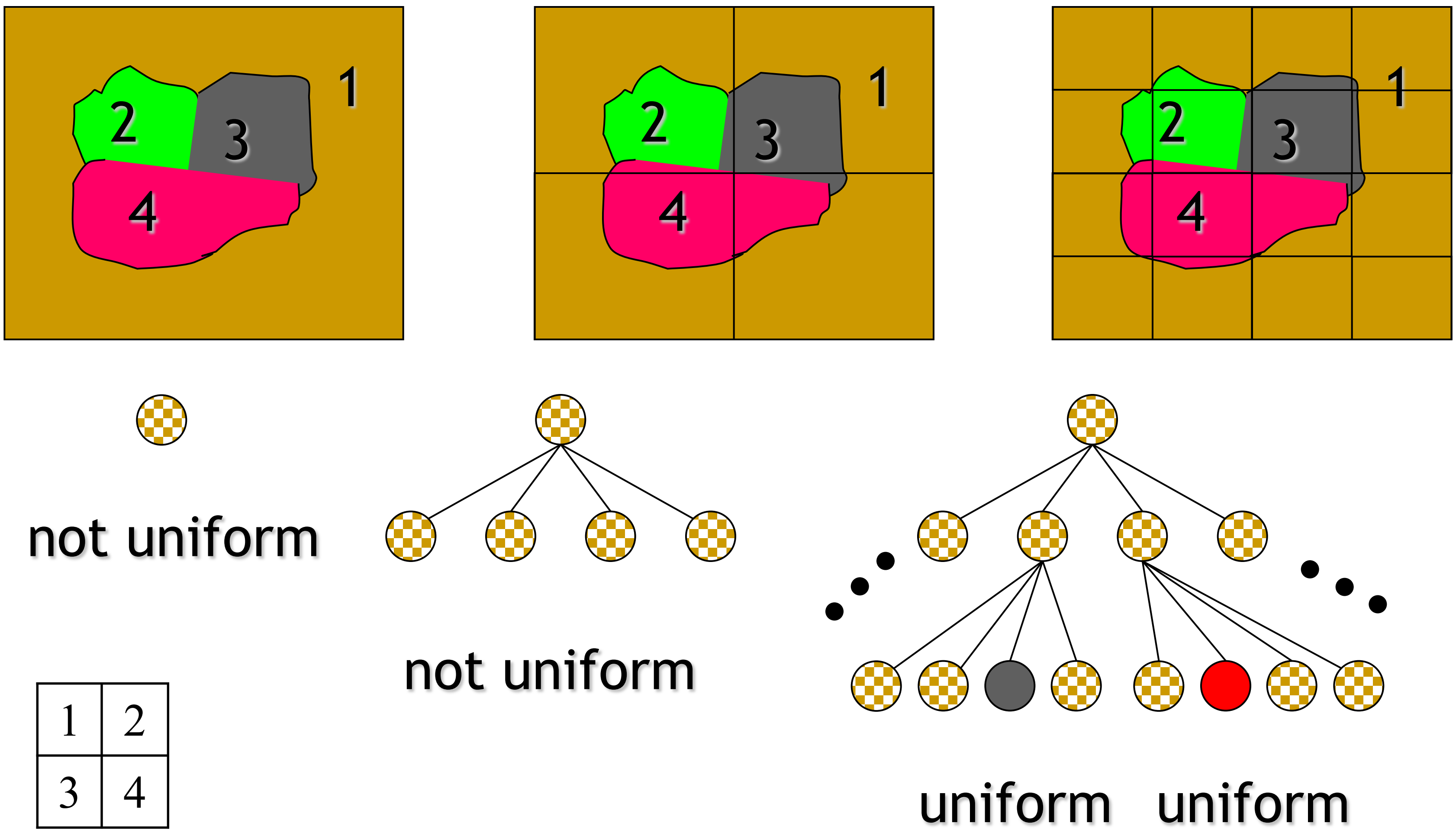
- Each layer corresponds to the output of a band-pass filter
- The stripes give stronger responses at particular scales



Quadtrees

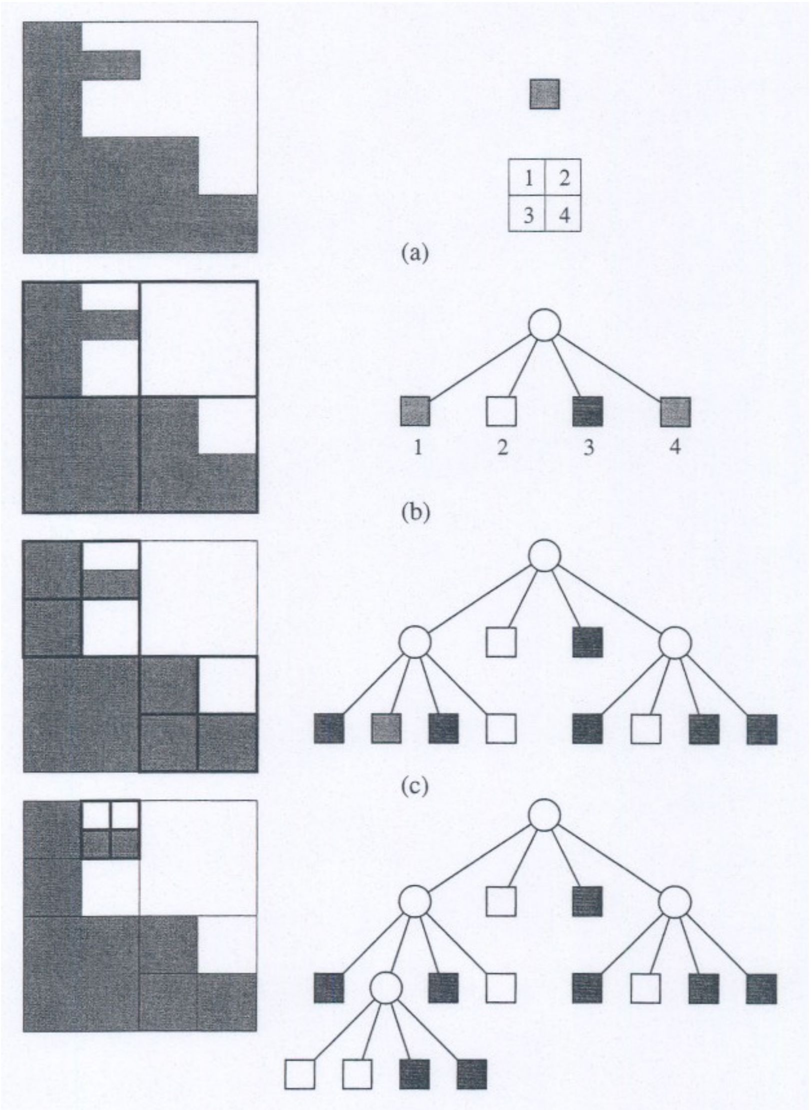
- Quadtrees
  - Trees where nodes have 4 children
  - Obtained by recursive splitting of an image
- Build quadtree:
  - Nodes represent regions
  - Every time a region is split, its node gives birth to 4 children
  - *Leaves* are nodes for uniform regions
- Merging:
  - Siblings that are "similar" can be merged

Quadtree Representation



Quadtree for "Binary Image"

- Contains three types of nodes:
  - white, black and gray
- A region in an image is split into four subregions of identical size
  - If all points in that region are either black or white – no more splitting
  - If that region contains both black and white, called gray region – further splitting required
  - The splitting process continues until no gray regions



Symbolic Representation

- A region can be represented using symbolic characteristics
  - Enclosing rectangle
  - Centroid
  - Moments
  - Euler number
- Other non-symbolic representation
  - Mean and variance of intensities

Split and Merge

- After the initial intensity-based region segmentation, the regions need to be *refined* and *reformed*
- Why?
  - Simple intensity algorithms usually result in too many regions
- Due to
  - High frequency noise
  - Gradual transitions between regions
- Methods:
  - Interactively or automatically
  - May use domain and/or image processing knowledge

Split and Merge

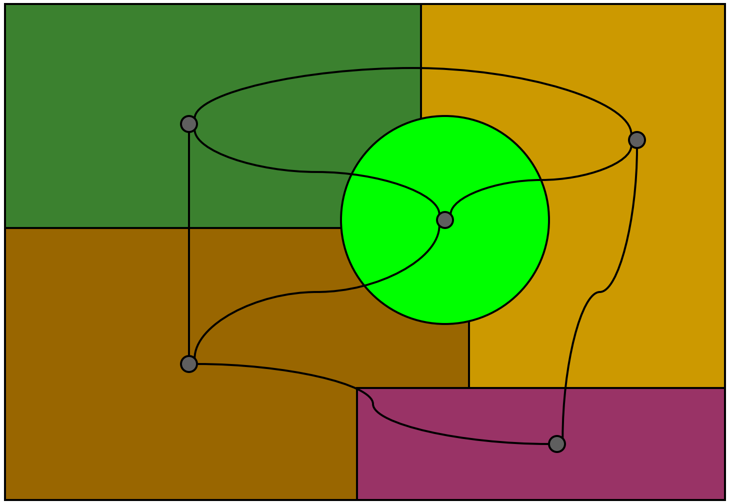
- All pixels belong to region
  - Object
  - Part of object
  - Background
- Automatic refinement is done using a combination of split and merge operations
  - Eliminate false boundaries and spurious regions by merging adjacent regions that belong to the same object
  - Add missing boundaries by splitting regions that contain parts of different objects

Region Merging

- Merge adjacent, similar regions
- What does "similar" mean?
  - Examples:
    - "similar" average values : $|m_i - m_j| < T$
    - "small" spread of gray values: $|g_{\max} - g_{\min}| < T$
      - $g_{\max} = \max\{g(x,y) \mid (x,y) \in R_i \cup R_j\}$
      - $g_{\min} = \min\{g(x,y) \mid (x,y) \in R_i \cup R_j\}$
  - Other "similarity":
    - luminance, statistical properties (mean, variance), color, texture, histogram shape, etc.
- Note:
  - A similar to B, and B similar to C does not imply that A is similar to C
  - Similar to some human behavior!!!

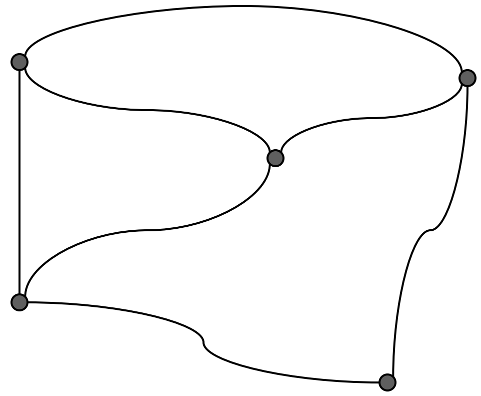
Region Adjacent Graph (RAG)

- A region adjacent graph (RAG) is used to represent regions and relationships among them in an image
- Nodes are used to represent regions, arc between nodes represent a common boundary between regions



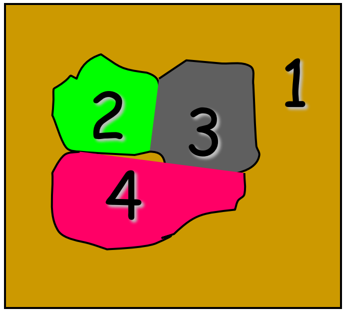
Region Adjacent Graph (RAG)

- A region adjacent graph (RAG) is used to represent regions and relationships among them in an image.
- Nodes are used to represent regions, arc between nodes represent a common boundary between regions



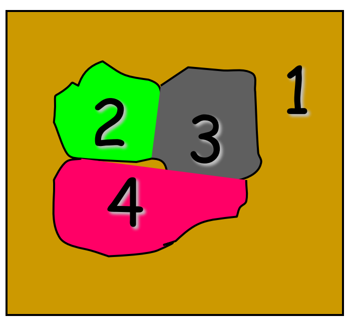
Region Merging Algorithm

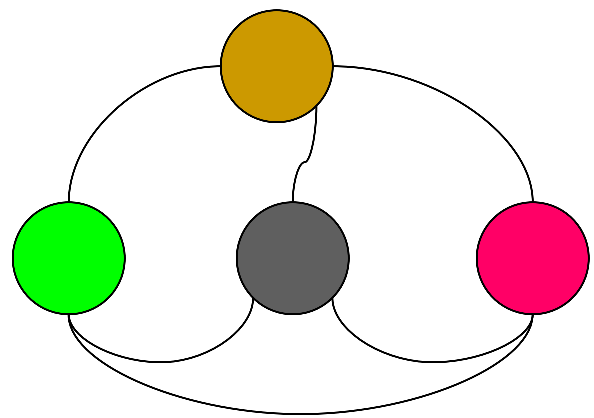
- From initial regions in the image using thresholding (on $n \times n$ regions, or manual selection) followed by component labeling



Region Merging Algorithm

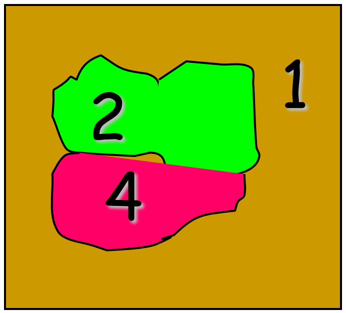
- Prepare a RAG for the image
  - Regions are the nodes
  - Adjacency relations are the links

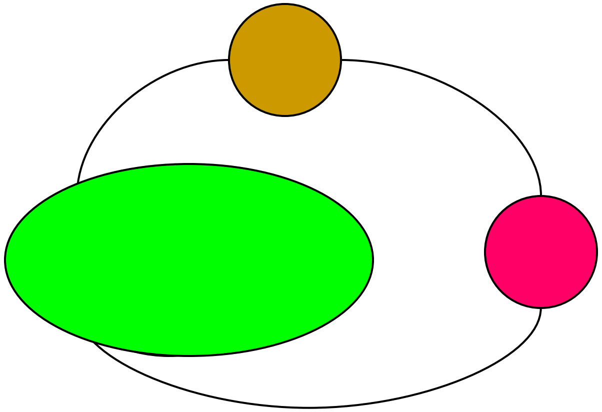




Region Merging Algorithm

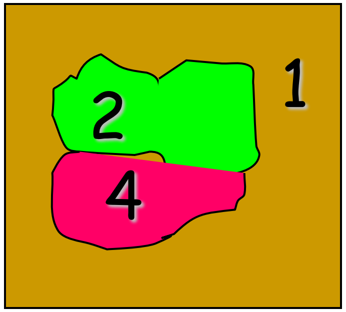
- For each region in an image:
  - Consider its adjacent region and test to see if they are similar
    - Compare their mean intensities
    - Statistical characteristics (similar statistical distribution of intensity values)
    - Remove weak edges (common boundaries between regions)
  - If they are similar, merge them and modify the RAG

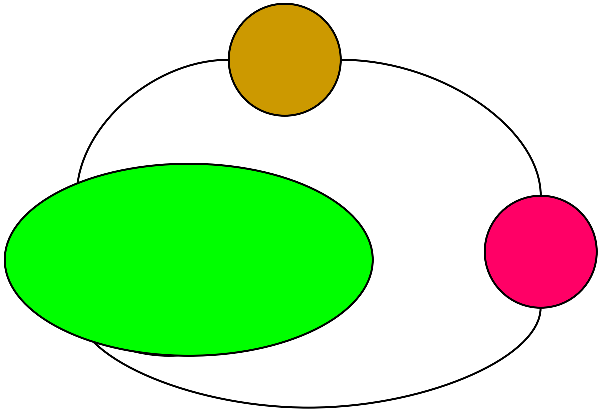




Region Merging Algorithm

- Repeat the previous step till no regions are merged





Using Similarity Measures

- Compare mean intensities (zero order model)
  - If the mean intensities do not differ by more than some predetermined value, the regions are considered similar
- Compare surface fitting (higher order model)
  - Model regions as polynomial surfaces
  - (e.g., take the gradual intensity changes into account)
  - Compute the fitting error
  - Merge regions if it decreases the fitting error

Using Statistics

- Model pixels as drawn from probability distributions
- Different regions have different distributions
- Assume the regions in an image have constant gray value corrupted by statistically independent, additive, zero-mean Gaussian noise
- Merge regions if their pixels came from the same distribution

Merge by Removing Weak Edges

- If the boundary between two regions is weak, then it can be removed and a large region can be obtained
- The "weak edge" can be specified by
  - *The intensity characteristics*
  - *The length of the common boundary*
- The common boundary is dissolved if the boundary is weak and the resulting boundary does not change gray value too quickly

Region Splitting

- If some property of a region is not constant, the region should be split
- Algorithm: Region Splitting
  - From initial regions in the image
  - For each region in an image, recursively perform the following steps:
    - Compute the *variance* in the gray value of the region
    - If the variance is above a threshold, split the region along the appropriate boundary
- Region splitting is generally more difficult than region merging

Split and Merge

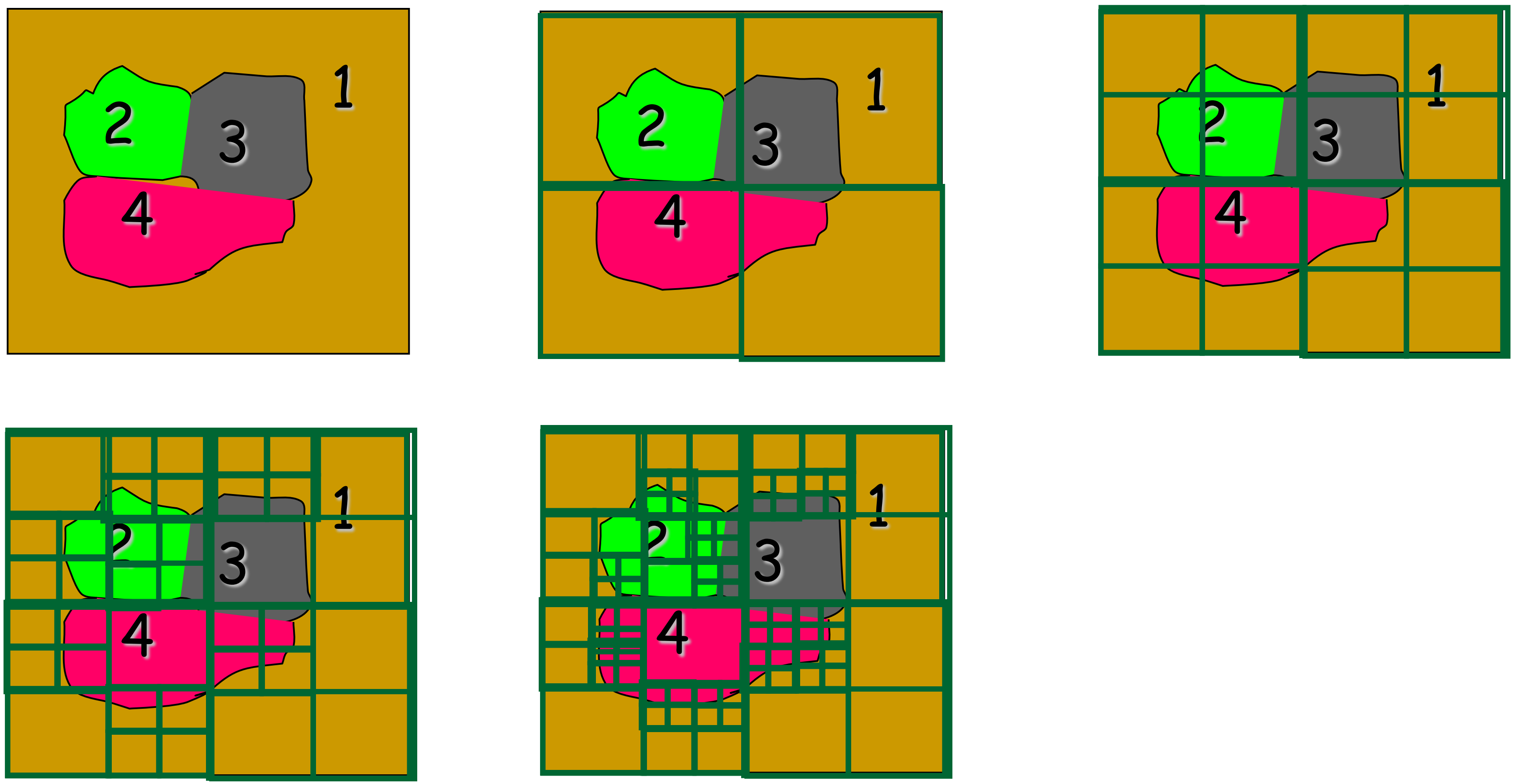
- Split and merge are often used together
- All of the pixels in a region is homogeneous according to a predicate H, i.e., $H(R)$
- Suppose an image is partitioned into a set of regions $\{R_i\}$ and

$$\bigcup_{i=1}^{n} R_i = P \qquad H(R_i) = \text{True} \qquad H(R_i \cup R_j) = \text{False}$$

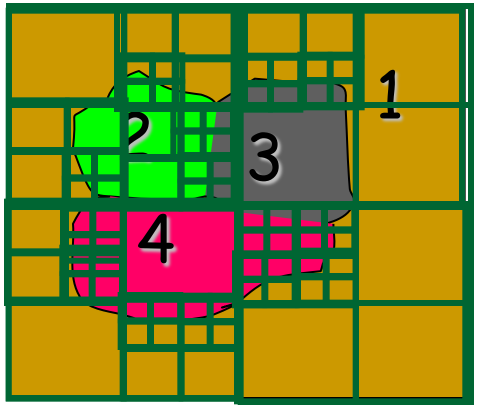
Split and Merge Algorithm

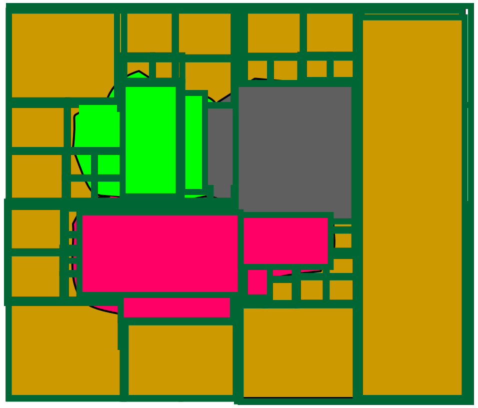
- Start with the entire image as a single region
- Pick a region R
  - If R is false, then split the region into four sub-regions
- Consider any two or more neighboring sub-regions, R_1, R_2, …, R_n, in the image
  - If $H(R_1 \cup R_2 \cup \ldots \cup R_n)$ is true, merge the n regions into a single region
- Repeat these steps till no further splits or merges take place

Split



Merge





4/11/2023

- Midterm Exam next Thursday (4/20)
- No class next Tuesday (4/18)

Region Growing

- A set of algorithms to group pixels with similar attributes together
- The basic idea is to grow from a seed pixel
  - At a labeled pixel, check its neighbors
    - If the attributes of its neighbor is similar to the attributes of the labeled pixel, label the neighbor
    - Repeat until there is no pixel that can be labeled
- A simple case
  - The attribute of a pixel is its pixel value
  - The similarity given by the difference between two pixel values
    - If the difference is smaller than a threshold, then they are similar
    - Otherwise they are not

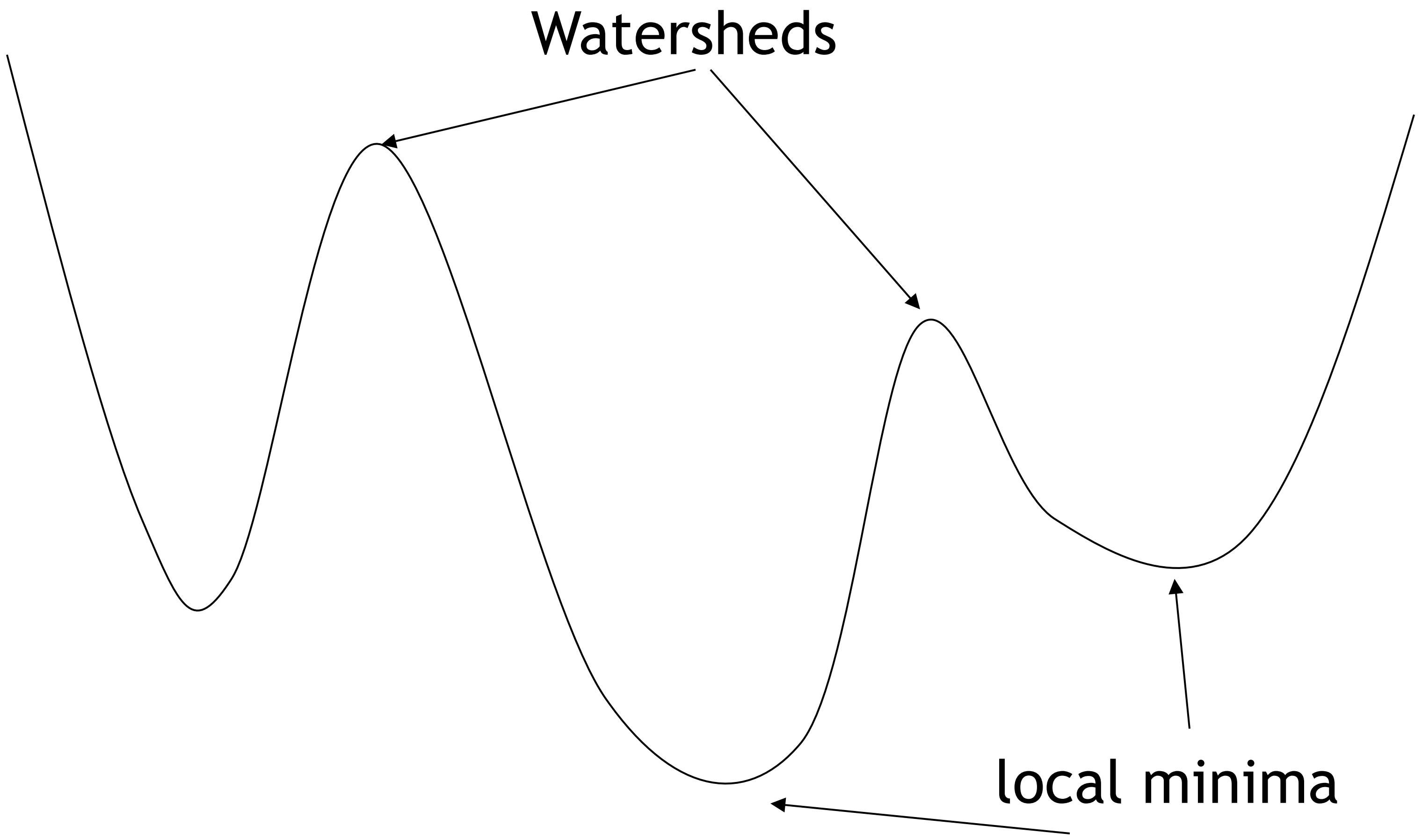
Region Growing

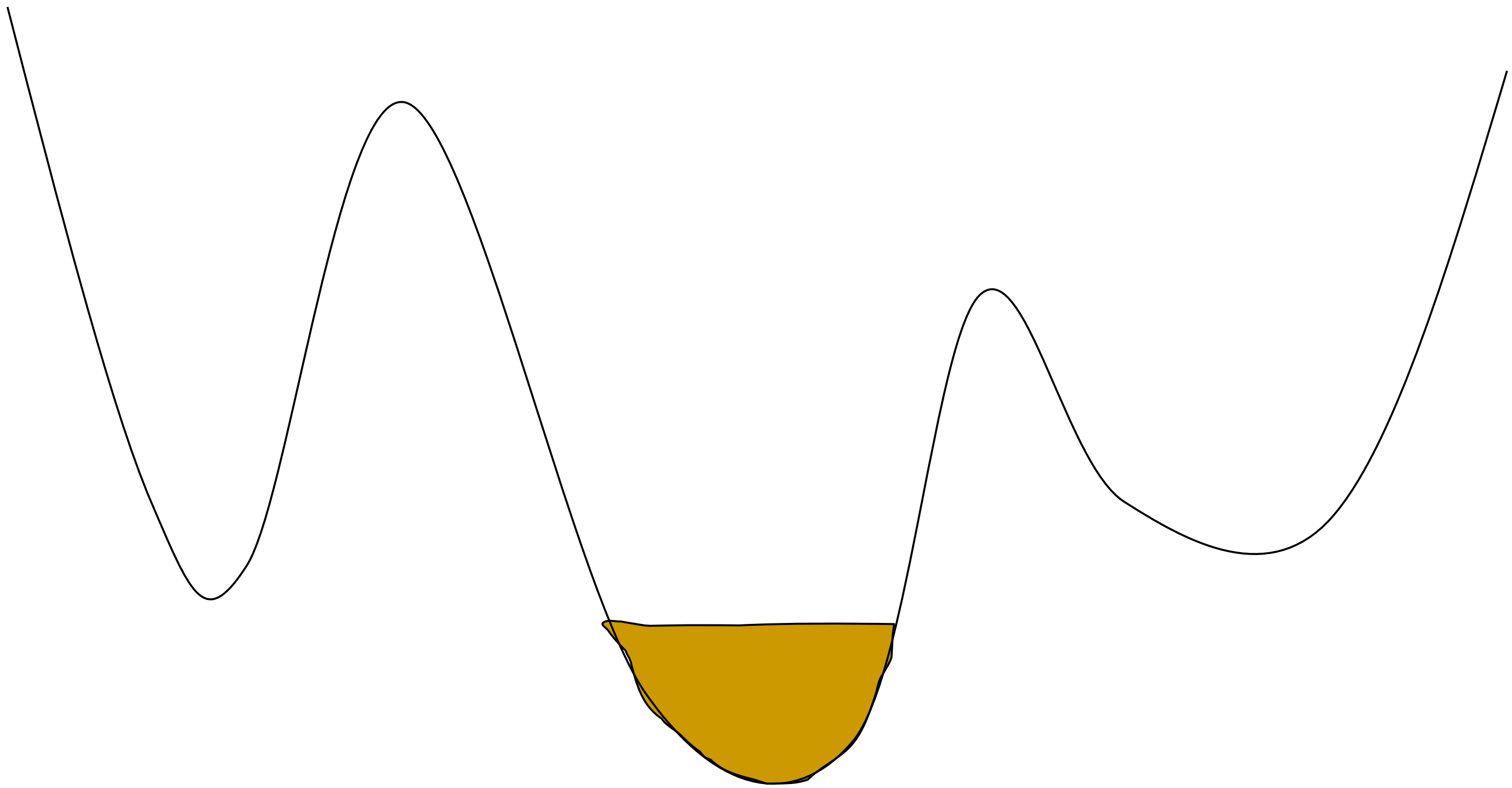
- Algorithm:
  - Generate initial seeds
  - Select a region
  - Add pixels to region
  - If growth stops, go back to step 2
- Efficient implementation
  - Based on scan-line algorithm in graphics
  - Each time we label a line instead of a pixel
  - This procedure is much more efficient than the recursive version

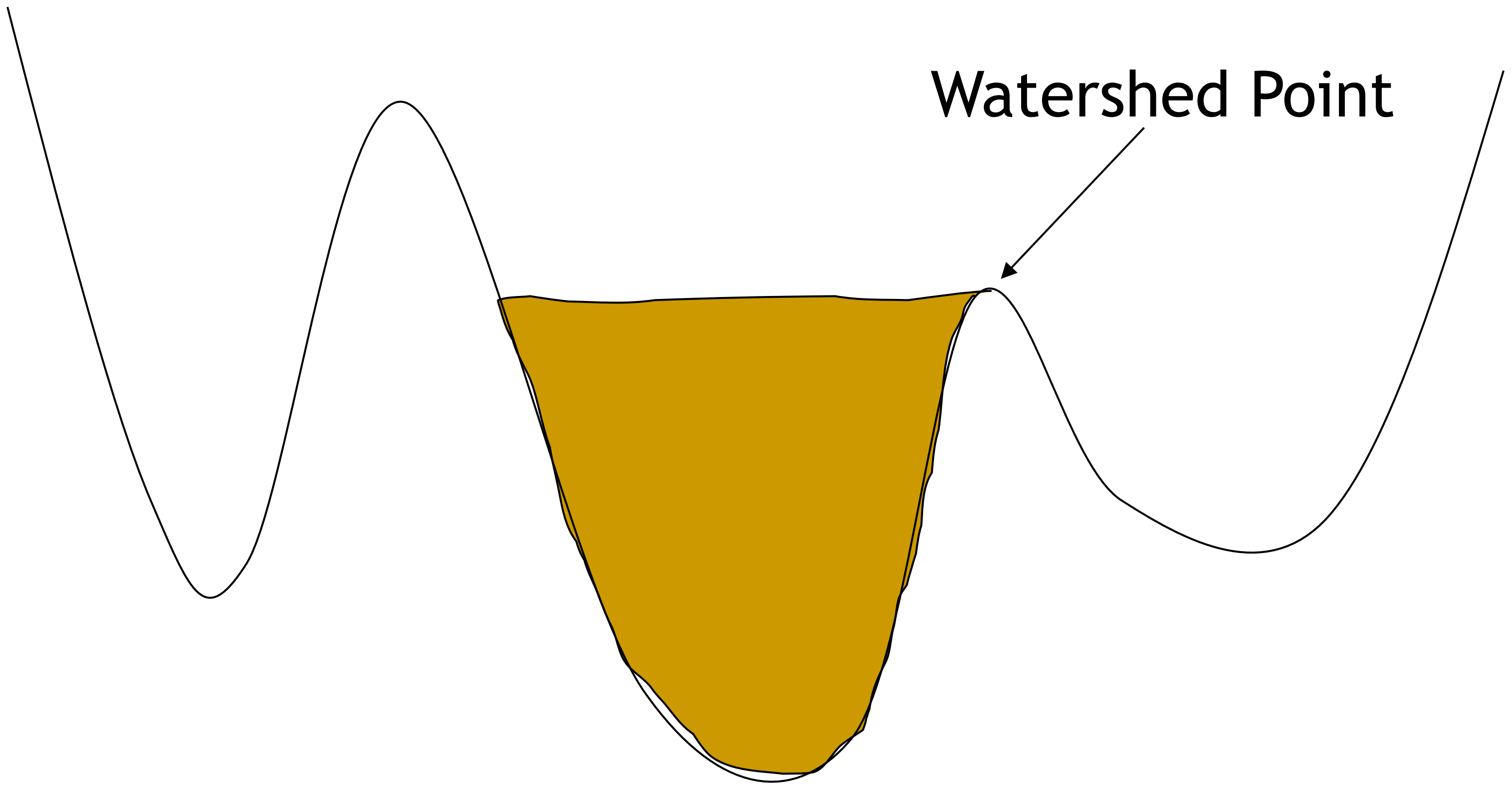
Watershed Segmentation

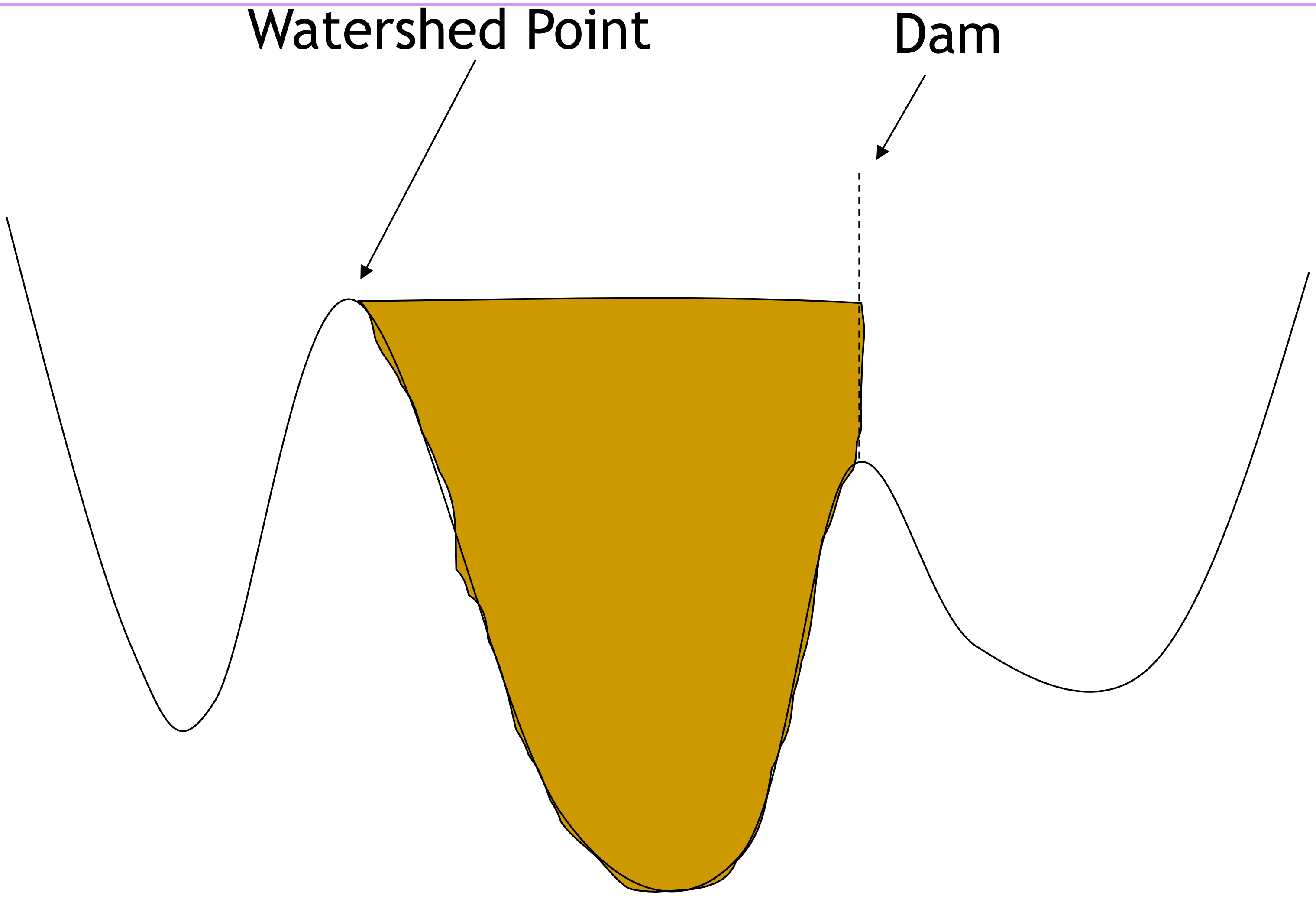
- Compare geographical watersheds
  - Divide landscape into catchment basins
- Edges correspond to watersheds
- Algorithm:
  - Locate local minima
  - Flood image from these points
  - When two floods meet
    - identify a watershed pixel
    - build a dam
    - continue flooding

Example

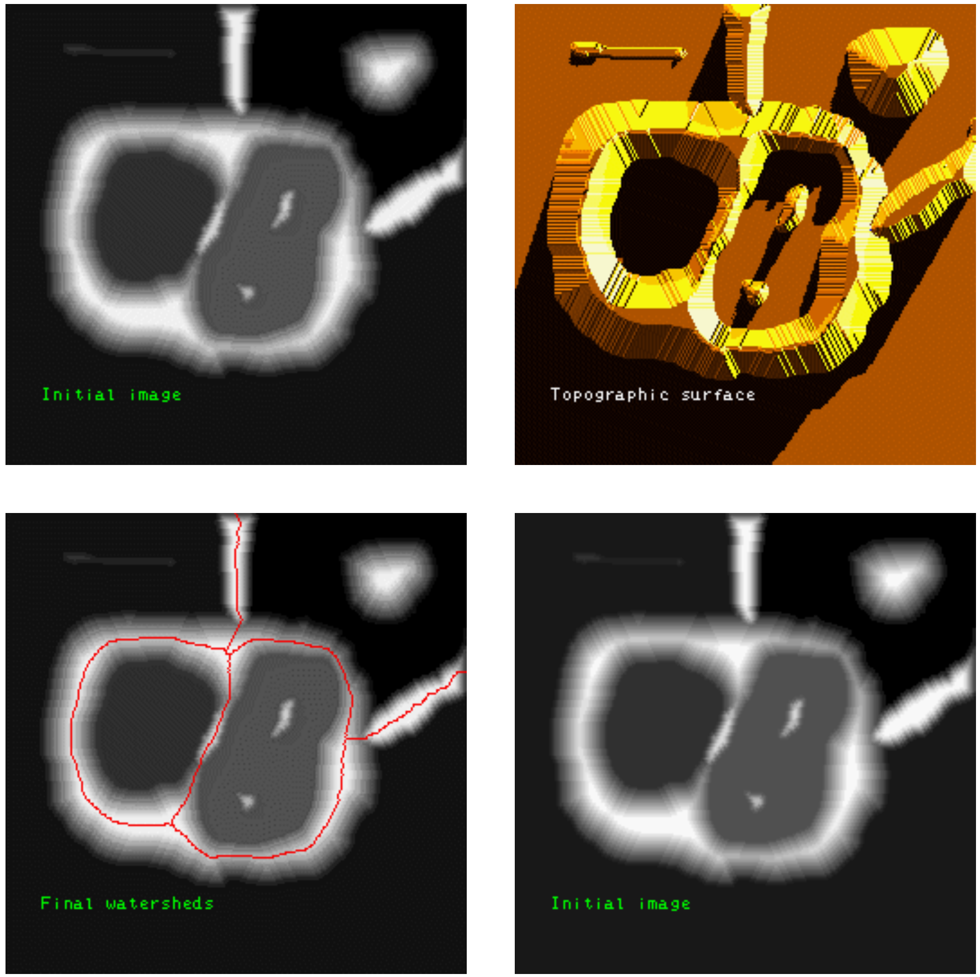





Watershed Point


Watershed Point
Dam

Example


Initial image
Topographic surface
Final watersheds
Initial image

Marker-Controlled Watershed

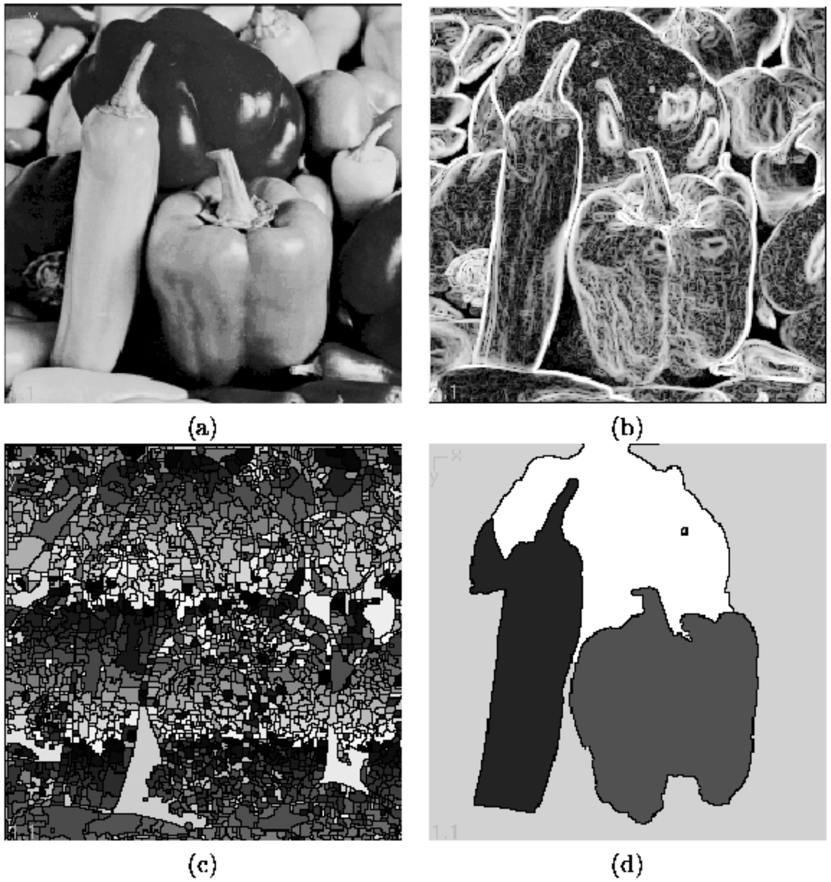


Figure 5.51: *Watershed segmentation: (a) original; (b) gradient image, 3×3 Sobel edge detection, histogram equalized; (c) raw watershed segmentation; (d) watershed segmentation using region markers to control oversegmentation. Courtesy W. Higgins, Penn State University.*