Digital Image Processing
CS 4650/7650, ECE 4655/7655

Image Segmentation

Gonzalez & Woods – Chapter 10

Szeliski – Chapter 5

Forsyth &Ponce – Chapter 14

Instructor: Filiz Bunyak Ersoy bunyak@missouri.edu

CS/ECE 4650/4655 7650/7655 Digital Image Processing

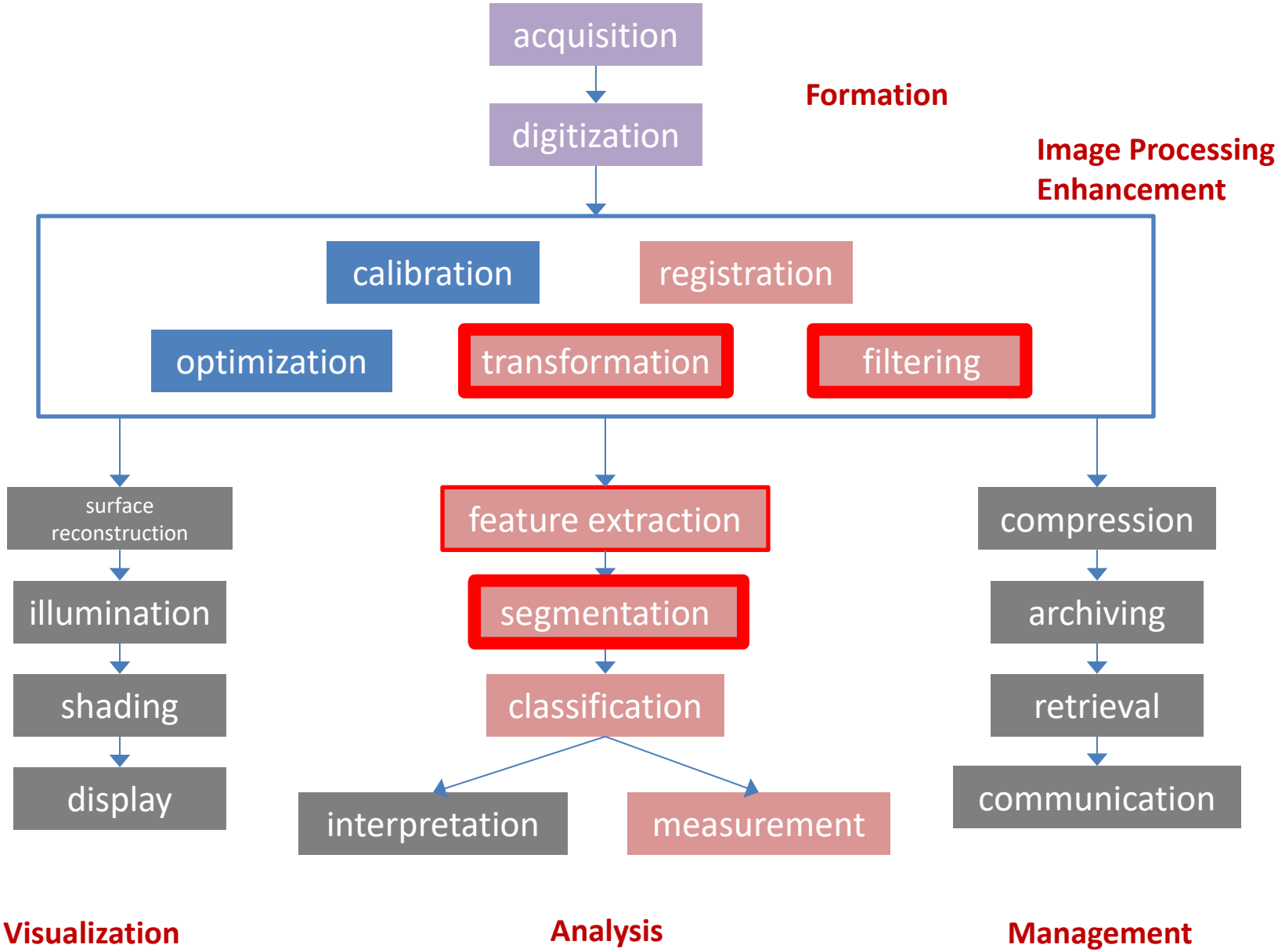
Previous Weeks:

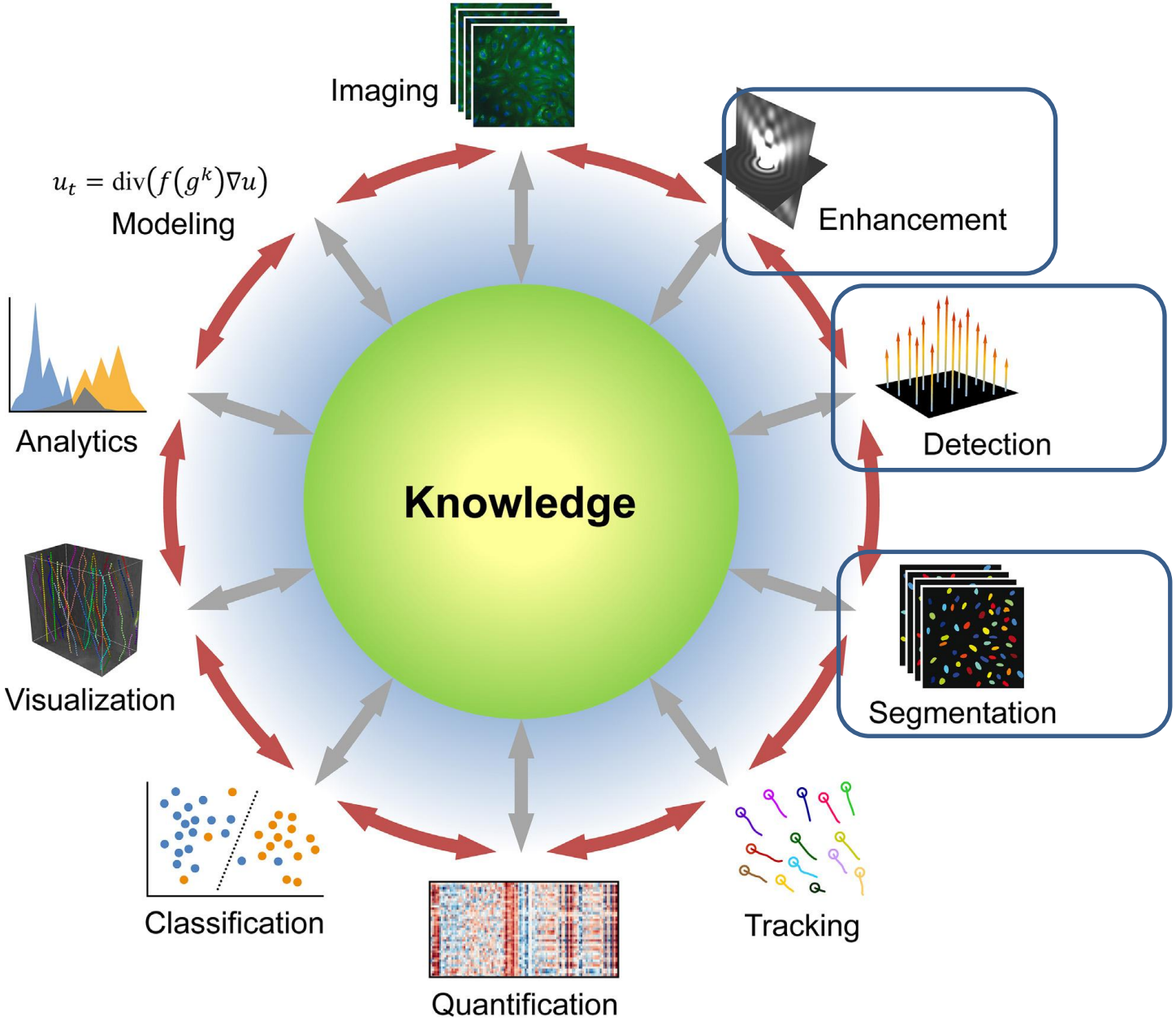
- Four classes of operations most commonly used in image processing
  1. Intensity transformation (Point Processes),
  2. Local image filtering,
  3. Geometrical transformation, and
  4. Image restoration (adaptive median filter)

This Week & next week: Segmentation,

1. Thresholding,
2. **Adaptive thresholding**
3. **Segmentation by Clustering**
4. Other traditional methods
5. Deep learning-based approaches (end of semester)

Figure adapted from:
Biomedical Image Processing by Thomas M. Deserno





Common tasks in image analysis.

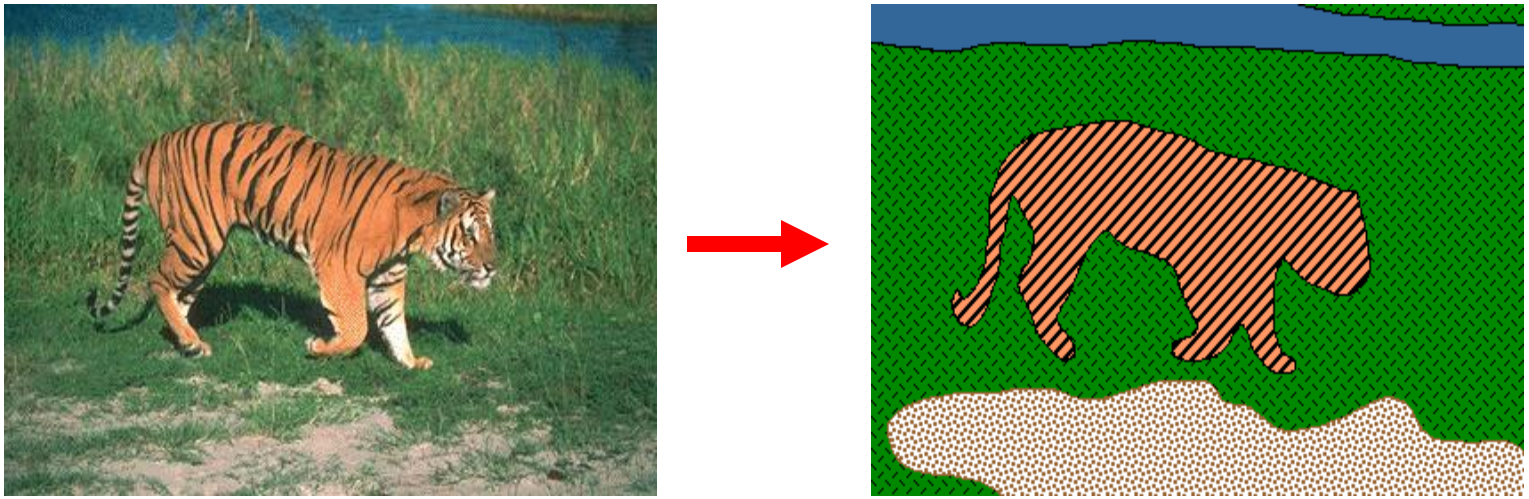
Meijering E. **A Bird's-Eye View of Deep Learning in Bioimage Analysis** Computational and Structural Biotechnology Journal. Aug. 2020

Segmentation

- The purpose of image segmentation is to partition an image into ***meaningful* regions** with respect to a **particular application**

- The segmentation is based on **measurements** taken from the image and might be
  - *greylevel*,
  - *color*,
  - *texture*,
  - *Depth,*
  - *motion…..*

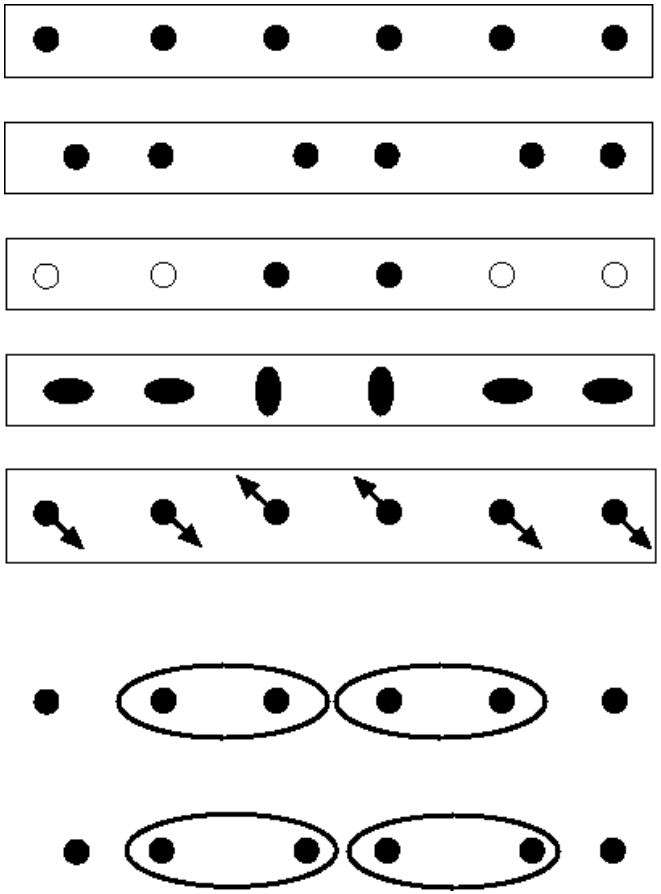
From images to objects

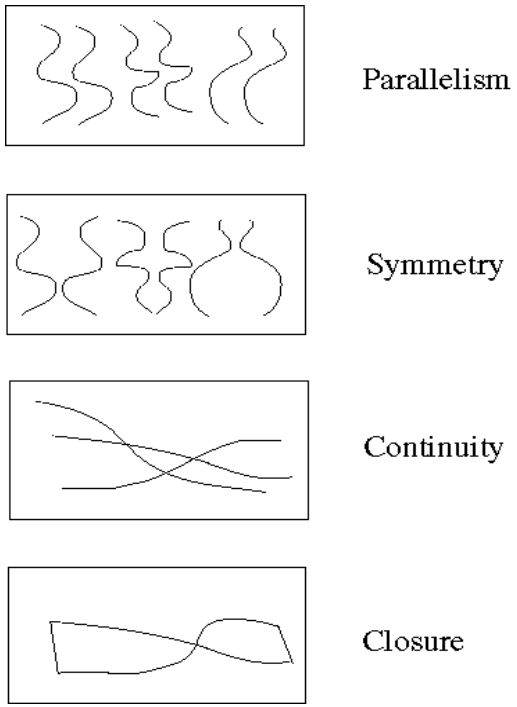


- What Defines an Object?
  - Subjective problem, but has been well-studied
  - Gestalt Laws seek to formalize this
    - proximity, similarity, continuation, closure, common fate

Segmentation cues

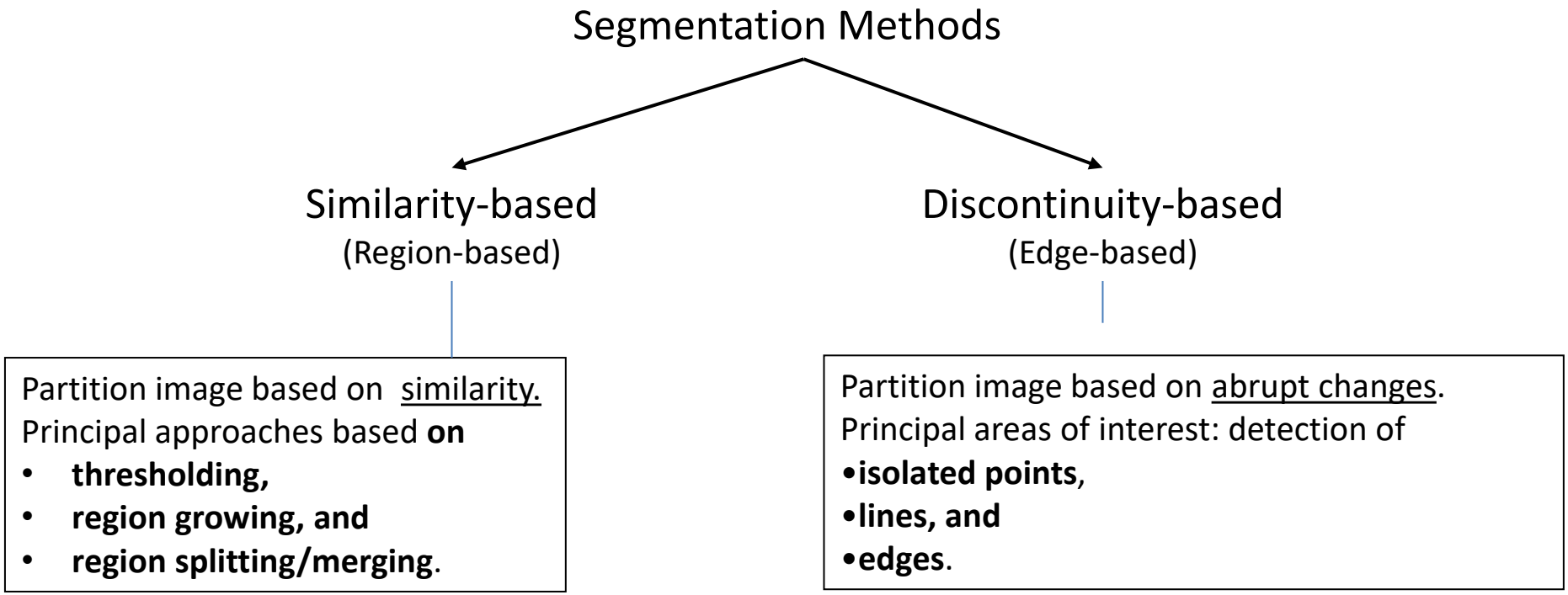
- What Defines an Object? (what are properties of pixels that belong together?)
  - Subjective problem, but has been well-studied
  - Gestalt Laws seek to formalize this: proximity, similarity, continuation, closure, common fate…





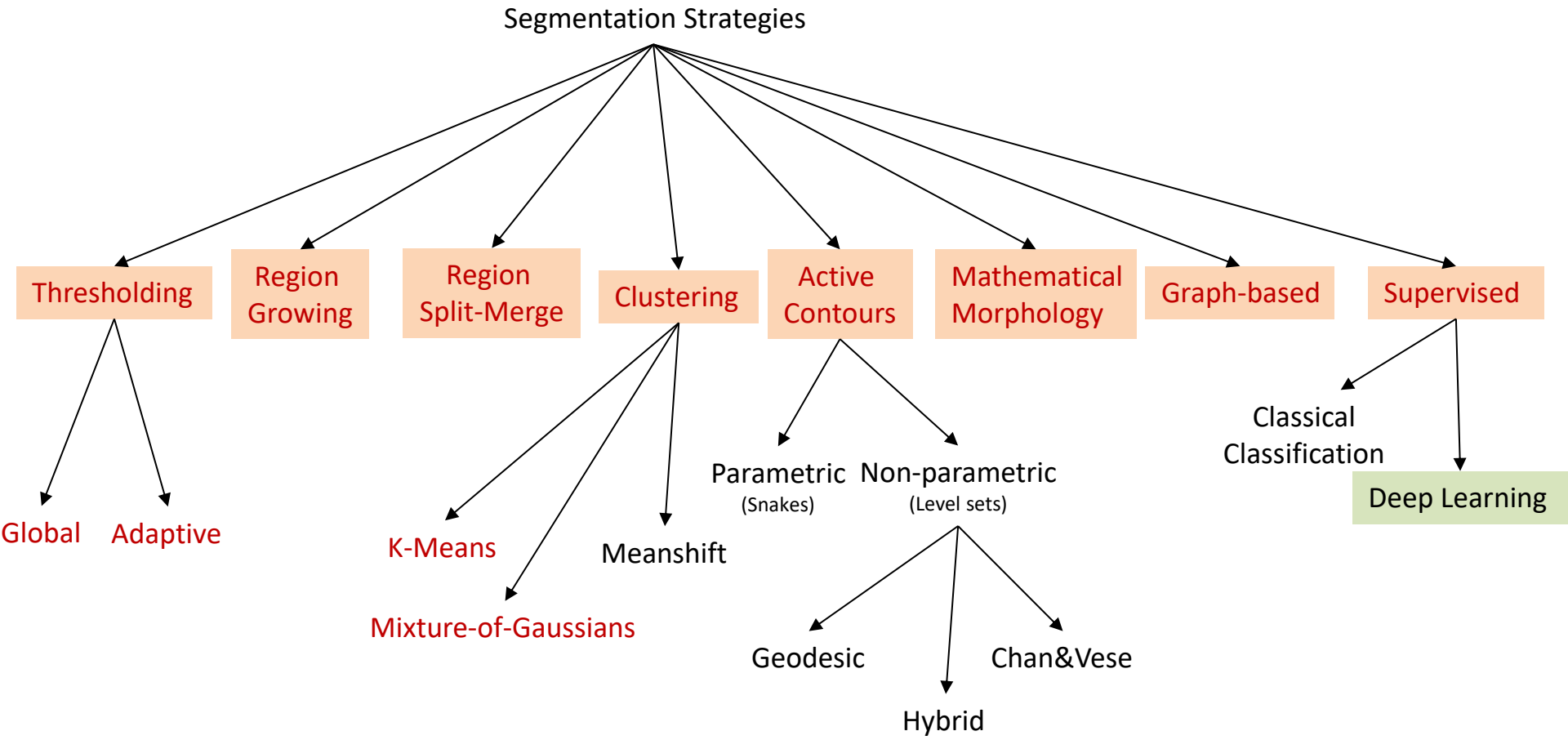
Classification of Segmentation Methods

Segmentation algorithms generally are based on one of two basic properties similarity and discontinuity:



Adapted from: MNT_image_segmentation.ppt
Medical Image Segmentation, Ralph Müller, Ph.D. Institute for Biomechanics, ETH Zürich, Switzerland

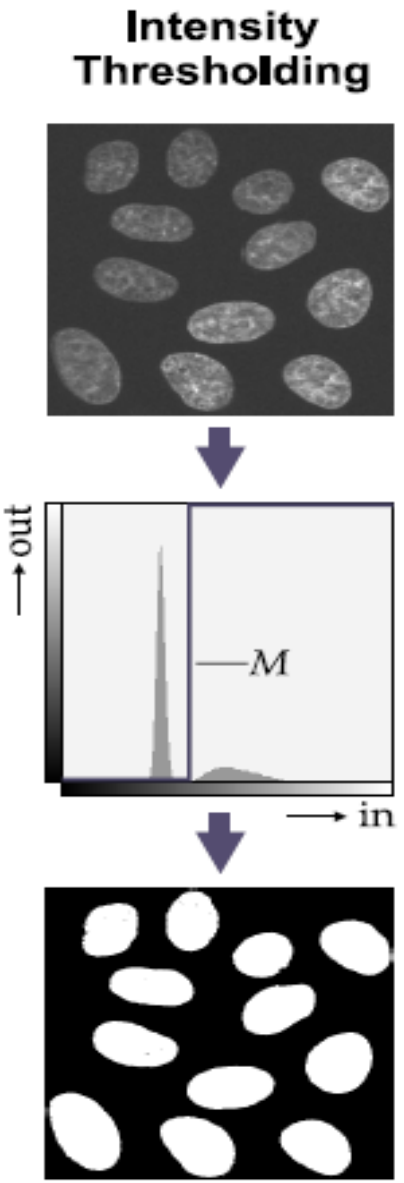
Some Image Segmentation Strategies



1. Global Thresholding

- Binary images can be obtained from gray-scale images by thresholding operations.

- A thresholding operation chooses some of the pixels
  - as the foreground pixels that make up the objects of interest and
  - the rest as background pixels.



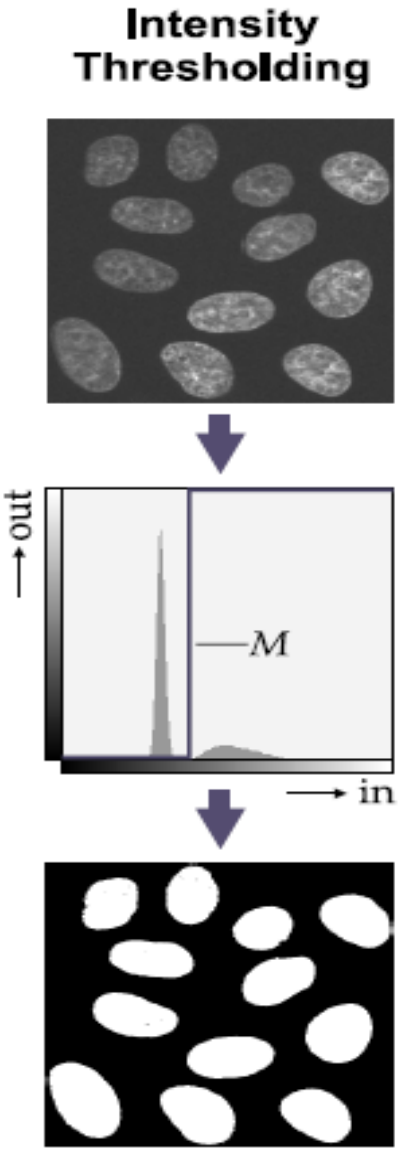
Automatic Thresholding: The Otsu Method

A measure of group homogeneity is variance. A group with high homogeneity will have low variance.

Dividing Criterion

1. choose a dividing score such that minimizes weighted sum of the within-group variance (emphasizes high group homogeneity) **OR**

2. choose a dividing score that maximizes squared difference between the group means (related to the between-group variance)



Otsu Thresholding : Approach #2

weights (class probabilities):

$$q_1(t) = \sum_{i=1}^{t} P(i)$$

$$q_2(t) = \sum_{i=t+1}^{n} P(i)$$

class means:

$$\mu_1(t) = \frac{1}{q_1(t)} \sum_{i=1}^{t} iP(i)$$

$$\mu_2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{n} iP(i)$$

class variances:

$$\sigma_1^2(t) = \frac{1}{q_1(t)} \sum_{i=1}^{t} [i - \mu_1(t)]^2 P(i)$$

$$\sigma_2^2(t) = \frac{1}{q_2(t)} \sum_{i=t+1}^{n} [i - \mu_2(t)]^2 P(i)$$

weighted within-class variance

$$\sigma^2 = \underbrace{\sigma_w^2(t)}_{\text{Within-class, from before}} + \underbrace{q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2}_{\text{Between-class,}}$$

The best threshold can be determined by a simple sequential search:

FOR (all possible values of t)

COMPUTE q_1,q_2, mu_1 mu_2 (using sums)

COMPUTE weighted **within-class variance**

$$\sigma_B^2(t) = q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$$

LOCATE the threshold t that **MAXIMIZES** σ_B^2.

Otsu Thresholding: Approach #3 (Recursive Computation)

RECURSIVE OTSU:

FOR (all possible values of t)

COMPUTE q_1, q_2, μ_1, μ_2 **(using recursion given below)**

COMPUTE weighted **between-class variance**

$$\sigma_b^2(t) = q_1(t)[1 - q_1(t)][\mu_1(t) - \mu_2(t)]^2$$

LOCATE the threshold t that **MAXIMIZES** σ_b^2.

Initialization...

$$q_1(1) = P(1); \quad \mu_1(0) = 0$$

Recursion...

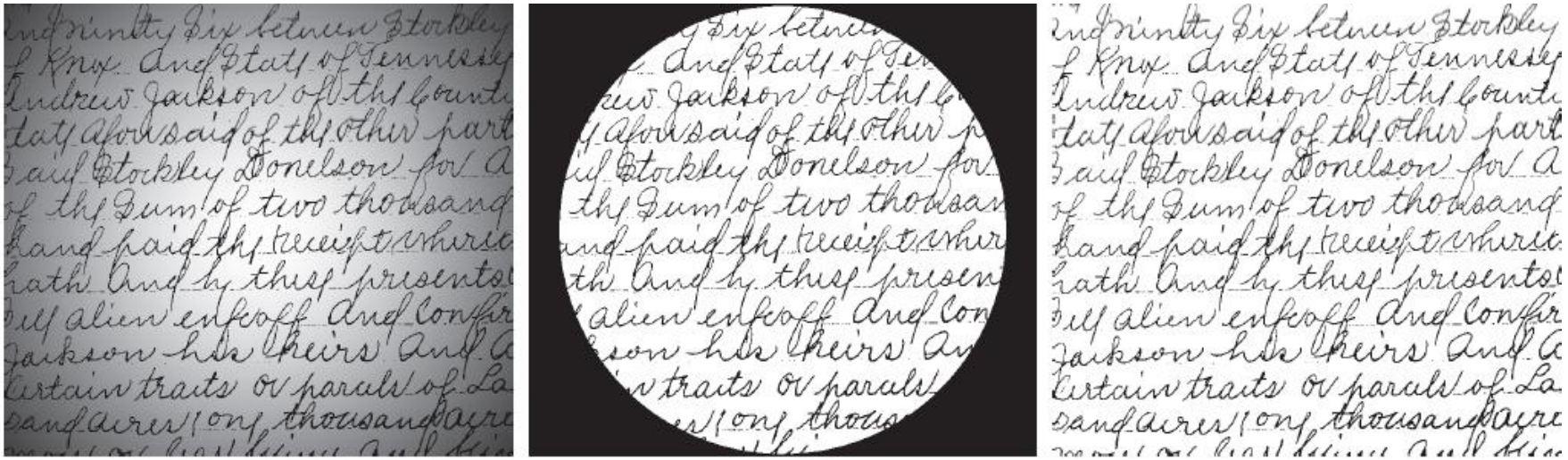
$$q_1(t+1) = q_1(t) + P(t+1)$$

$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)P(t+1)}{q_1(t+1)}$$

$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\mu_1(t+1)}{1 - q_1(t+1)}$$

Gonzalez & Woods Figure 10.44

(a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.



a b c

2. Adaptive/Local Thresholding Based on Local Image Properties

- Compute a threshold at every point (x,y) based on one or more specified properties in a neighborhood
- Thresholding based on local mean and variance

Local standard deviation for neighborhood centered at (x,y)

Local mean for neighborhood centered at (x,y)

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

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

global mean

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

Significant power (with a modest increase in computation) can be added to variable thresholding by using predicates (Q) based on the parameters computed in the neighborhood of a point (x, y):

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

Example predicate

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

3. Segmentation by Region Growing

Start from selected seed points in the image and to iteratively add connected points to form labeled regions. Assumes (and suffers from) a similar image model as in the case of intensity thresholding.

Ordinary region growing:

- The most straightforward implementation: ordinary region growing,

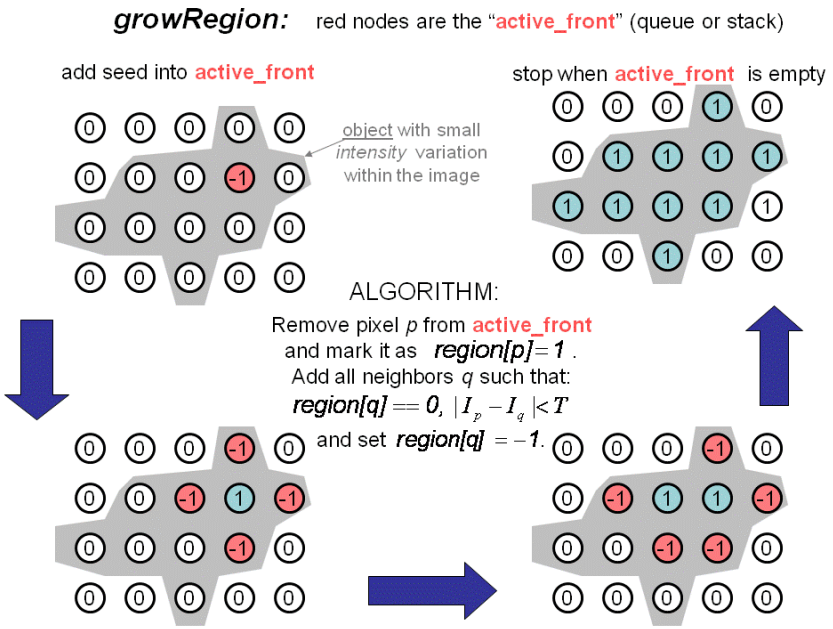


Figure: Ulas Bagci, UCF

Cell Segmentation: 50 Years Down the Road, Erik Meijering
IEEE Signal Processing Magazine, vol. 29, no. 5, September 2012, pp. 140–145

3. Segmentation by Region Growing

Region growing is a procedure that groups pixels or subregions into larger regions based on predefined criteria for growth.

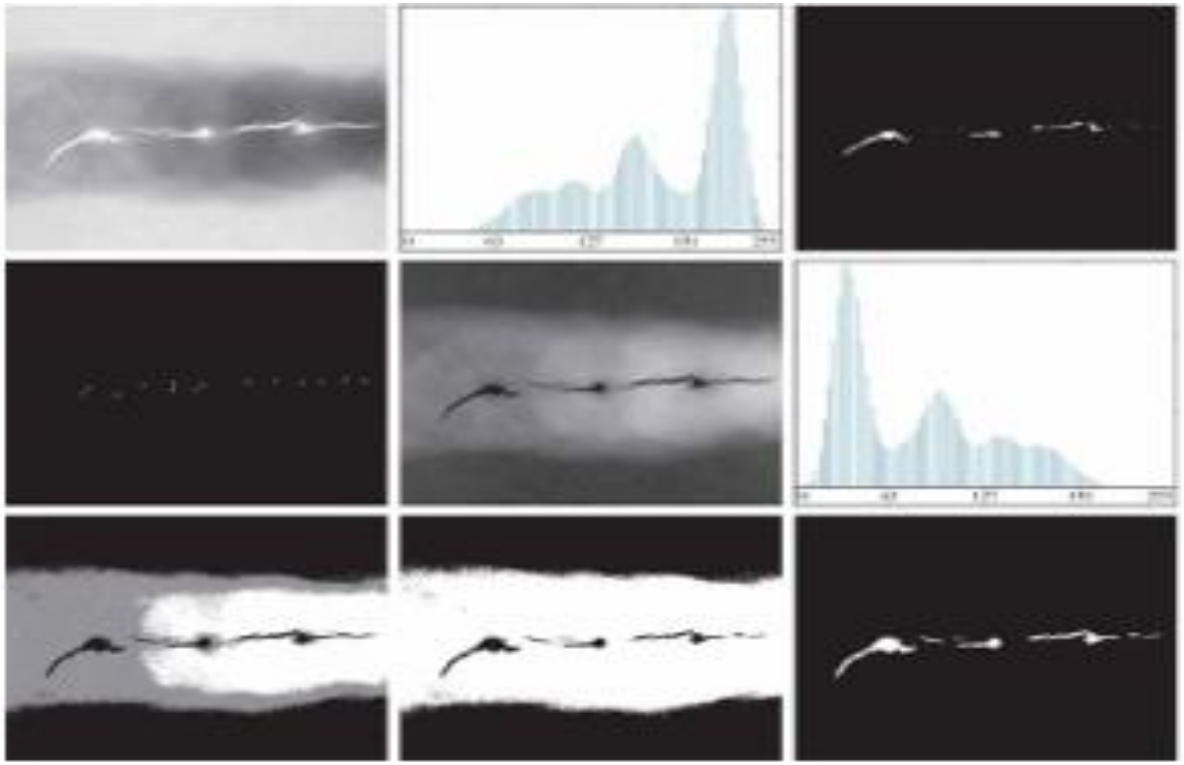
The basic approach:

- Start with a set of "seed" points,
- From the seeds grow regions by appending to each seed those neighboring pixels that have predefined properties similar to the seed (such as ranges of intensity or color).

1. Find all connected components in $S(x, y)$ and reduce each connected component to one pixel; label all such pixels found as 1. All other pixels in S are labeled 0.
2. Form an image f_Q such that, at each point (x, y), $f_Q(x, y) = 1$ if the input image satisfies a given predicate, Q, at those coordinates, and $f_Q(x, y) = 0$ otherwise.
3. Let g be an image formed by appending to each seed point in S all the 1-valued points in f_Q that are 8-connected to that seed point.
4. Label each connected component in g with a different region label (e.g.,integers or letters). This is the segmented image obtained by region growing.

3. Segmentation by Region Growing

(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 the seed value (255) and (a).
(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.)



a b c
d e f
g h i

4. Segmentation by Region Splitting and Merging

- Subdivide an image initially into a set of disjoint regions and then merge and/or split the regions in an attempt to satisfy the conditions of segmentation
- **Hierarchical split-and-merge schemes**: operating per resolution layer and using some uniformity predicate.

The preceding discussion can be summarized by the following procedure in which, at any step, we

1. Split into four disjoint quadrants any region R_i for which $Q(R_i) = \text{FALSE}$.
2. When no further splitting is possible, merge any adjacent regions R_j and R_k for which $Q(R_j \cup R_k) = \text{TRUE}$.
3. Stop when no further merging is possible.

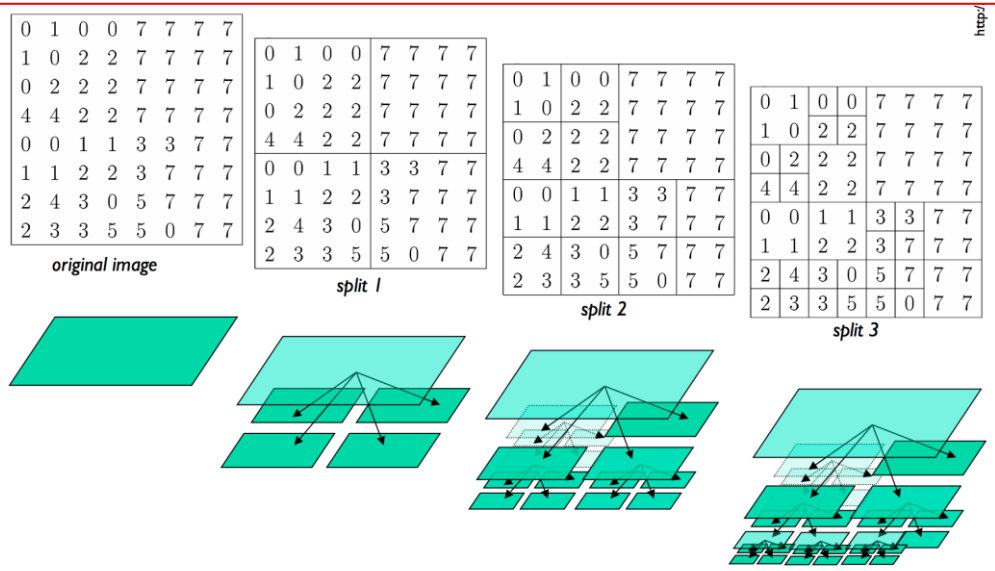
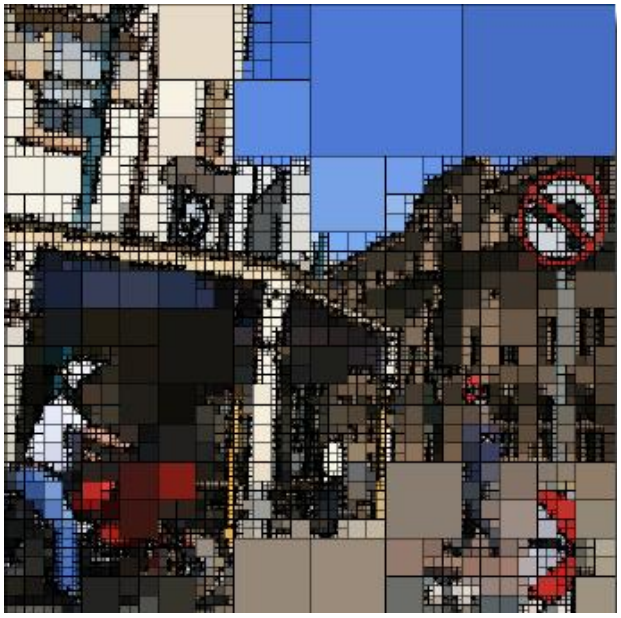


Figure: Ulas Bagci, UCF

Cell Segmentation: 50 Years Down the Road, Erik Meijering
IEEE Signal Processing Magazine, vol. 29, no. 5, September 2012, pp. 140–145

Quadtree


http://devmag.org.za/2011/02/23/quadtrees-implementation/

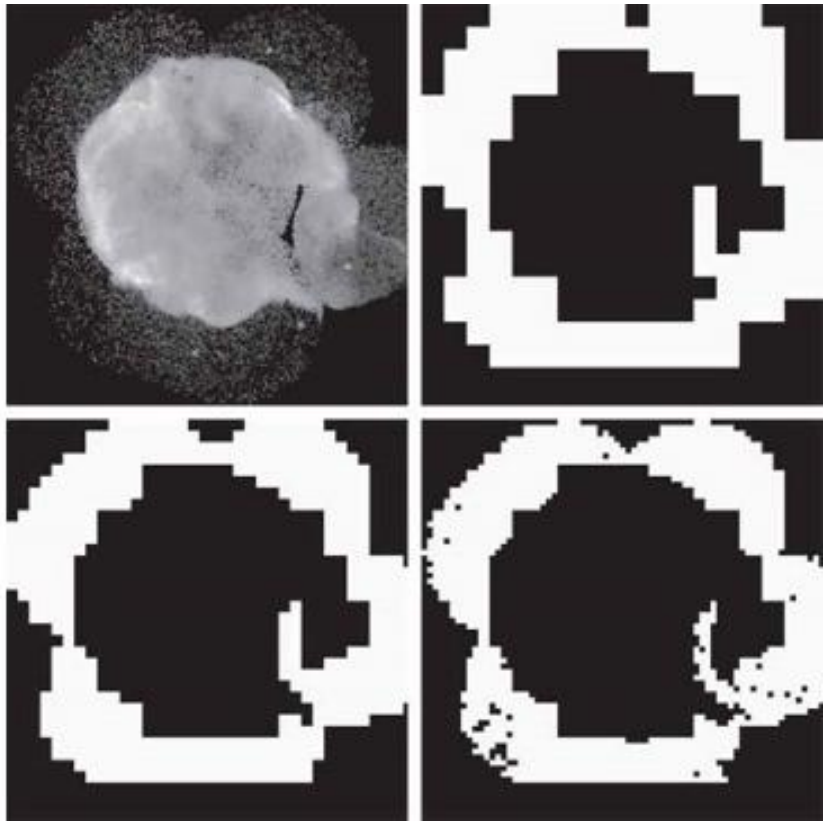
Geometry in Action: https://www.ics.uci.edu/~eppstein/gina/quadtree.html

Gonzalez & Woods Figure 10.48

(a) Image of the Cygnus Loop supernova, taken in the X-ray band by NASA's Hubble Telescope.

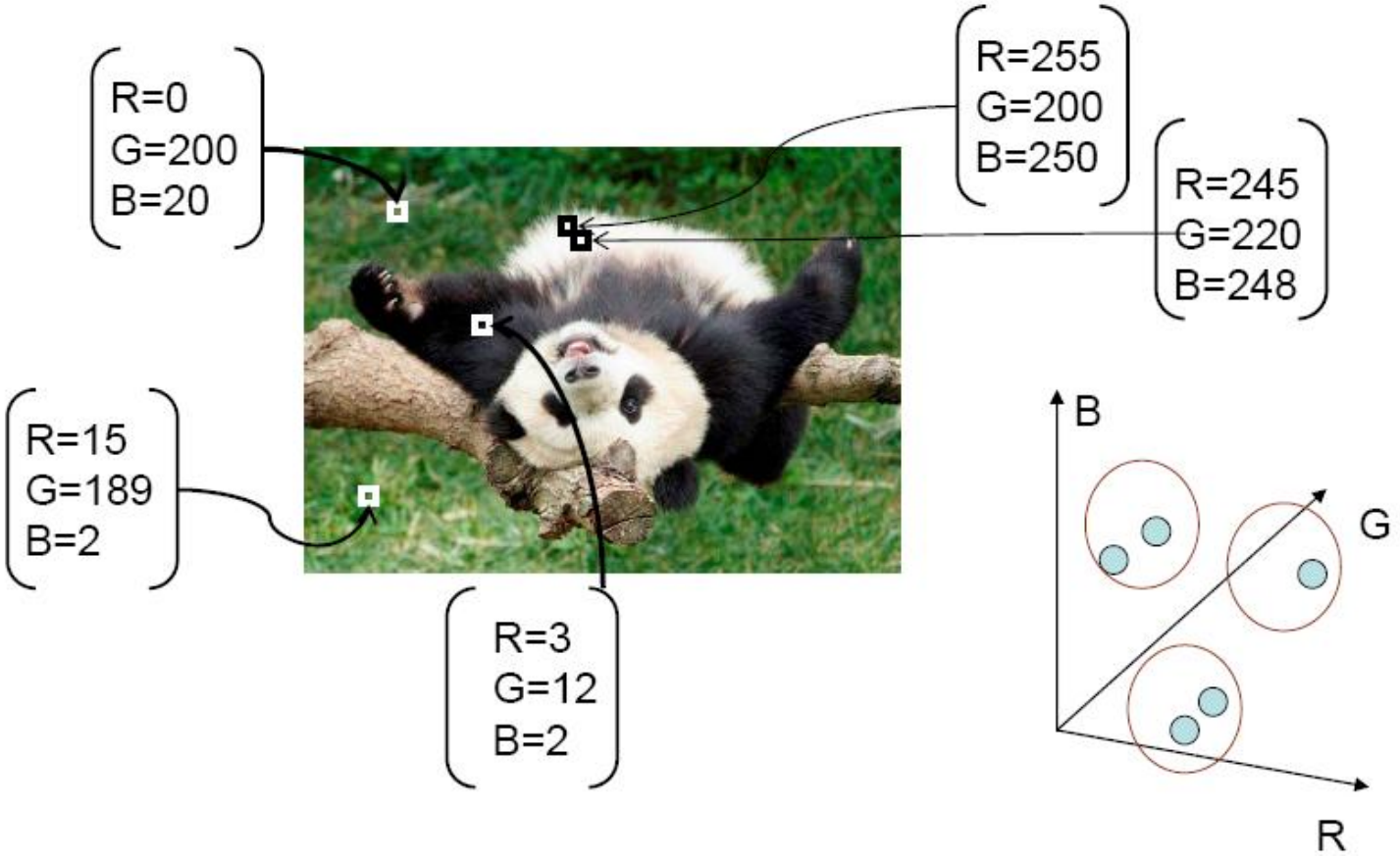
(b) (b) through (d) Results of limiting the smallest allowed quadregion to be of sizes of 32x32, 16x16, and 8x8 pixels, respectively. (Original image courtesy of NASA.)

a b
c d



5. Segmentation as clustering

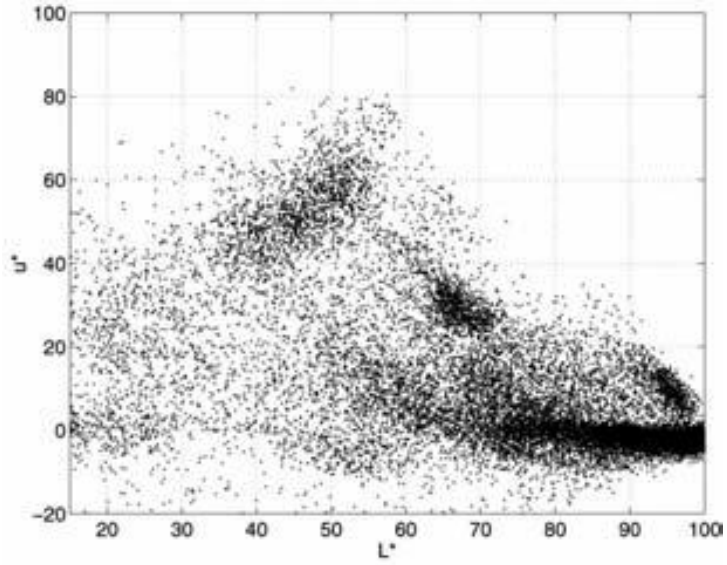
- Cluster similar pixels (features) together

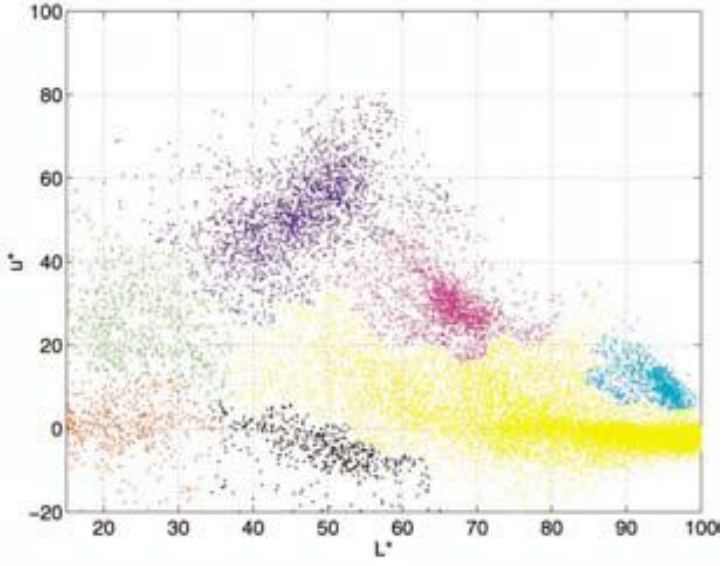


Source: K. Grauman

Segmentation as clustering

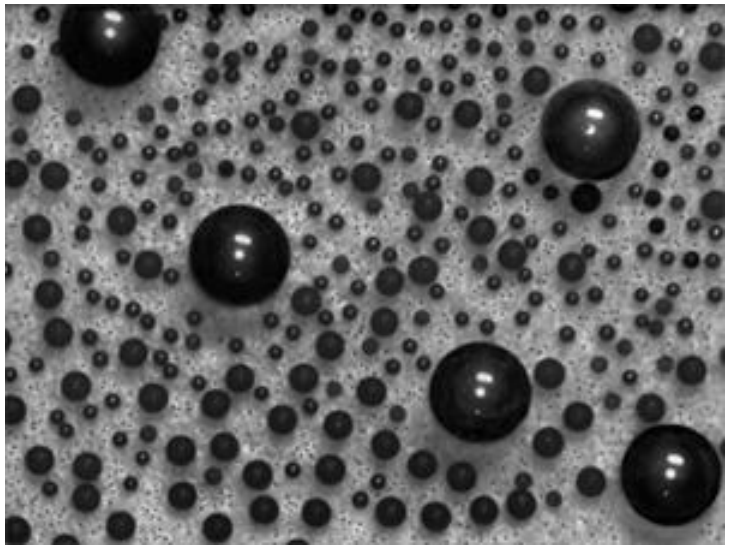
How to choose the representative colors?





Histogram-based segmentation

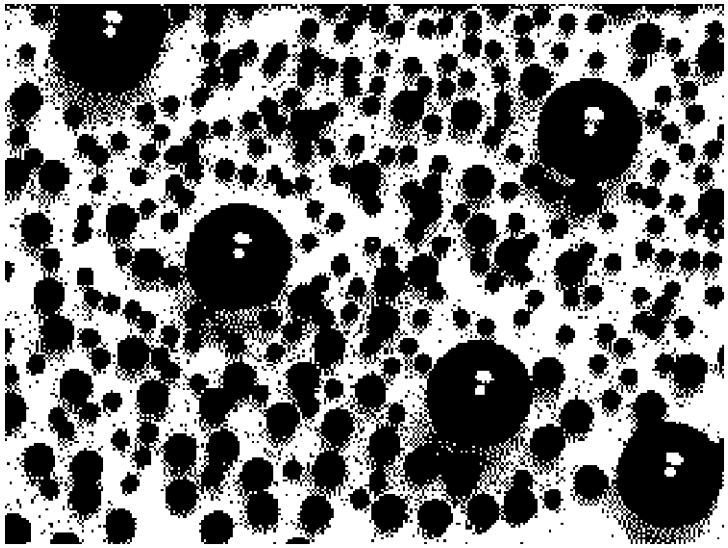
- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color

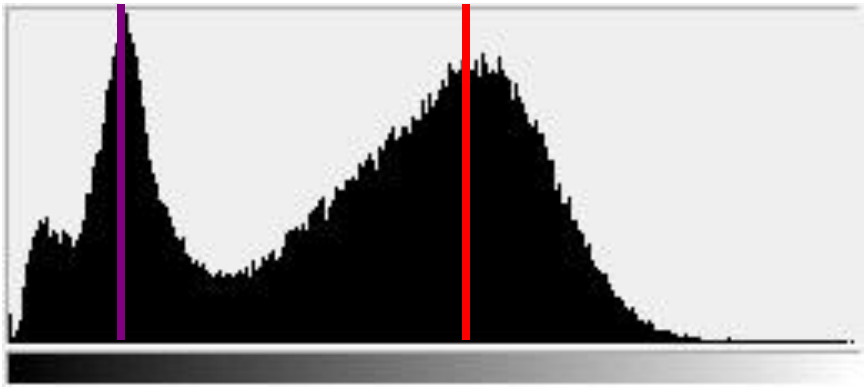


Simon-Seitz

Histogram-based segmentation

- Goal
  - Break the image into K regions (segments)
  - Solve this by reducing the number of colors to K and mapping each pixel to the closest color



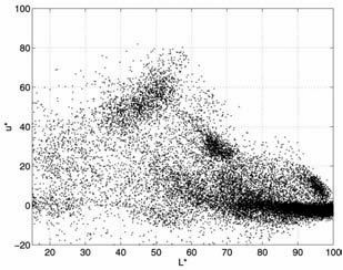


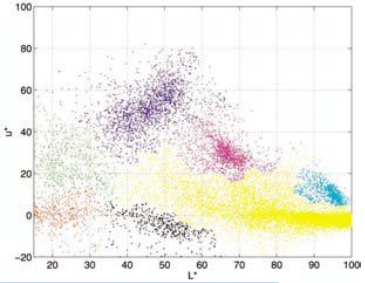
Here's what it looks like if we use two colors

Simon-Seitz

Meanshift + Mode Finding (K-means, Mixture of Gaussians)

1. Pixel $\rightarrow$ color, position, texture….
   $\rightarrow$ Feature vector
   - Feature vector : Samples from unknown probability density function
2. Try to find clusters (modes) in this distribution





K-Means & MoG

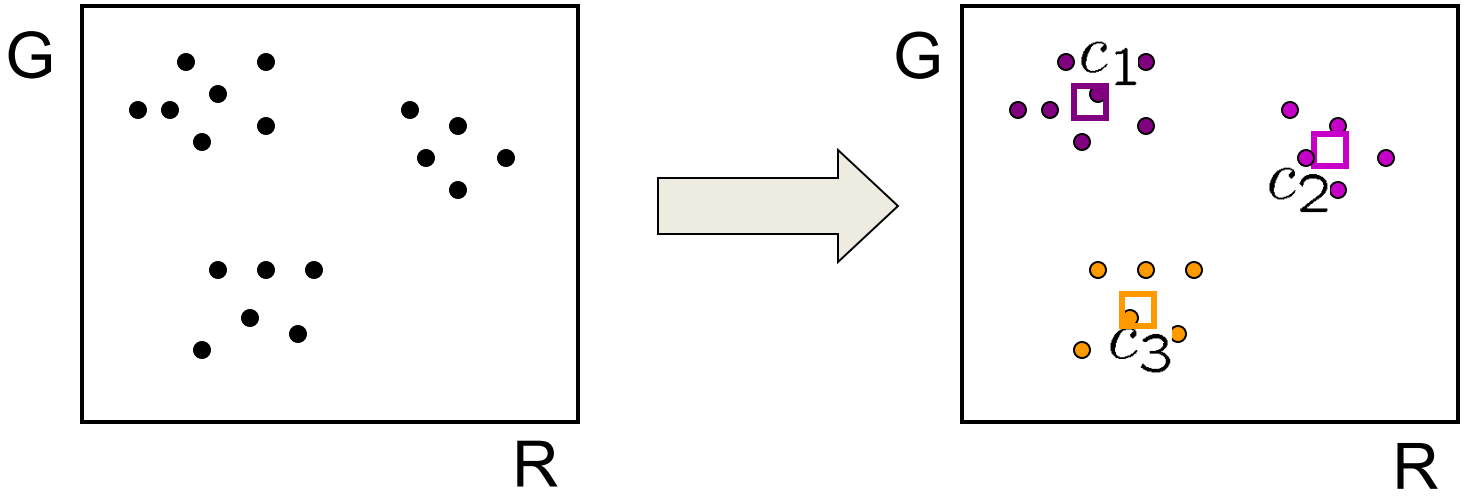
- Parametric model
- Superposition of simpler distributions (Gaussians)
- Center & Shape (covariance) can be estimated

Meanshift

- Non-parametric
- Smooth the distribution
- Find peaks & regions corresponding to peaks

Clustering

- How to choose the representative colors?
  - This is a clustering problem!



Objective

- Each point should be as close as possible to a cluster center
  - Minimize sum squared distance of each point to closest center

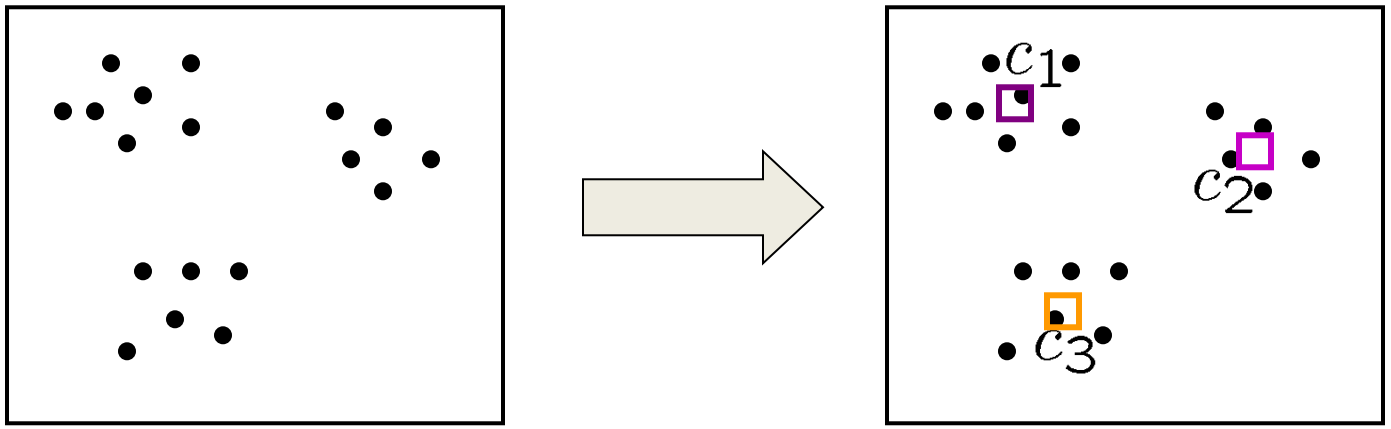
$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Simon-Seitz

Break it down into subproblems

1) Suppose I tell you the cluster centers c_i

- Q: how to determine which points to associate with each c_i?
- A: for each point p, choose closest c_i



2) Suppose I tell you the points in each cluster

- Q: how to determine the cluster centers?
- A: choose c_i to be the mean of all points in the cluster

Simon-Seitz

K-means clustering

K-means clustering algorithm

1. Randomly initialize the cluster centers, c_1, ..., c_K
2. Given cluster centers, determine points in each cluster
   - For each point p, find the closest c_i. Put p into cluster i
3. Given points in each cluster, solve for c_i
   - Set c_i to be the mean of points in cluster i
4. If c_i have changed, repeat Step 2

Properties

- Probability density : superposition of spherically symmetric distributions
- Will always converge to *some* solution
- Can be a "local minimum"
  - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Simon-Seitz

Gonzalez & Woods Figure 10.49

(a) Image of size 688x688 pixels
(b) Image segmented using k-means algorithm with k=3

a b

K-Means clustering

- K-means clustering based on intensity or color is essentially vector quantization of the image attributes
  - Clusters don't have to be spatially coherent

Image

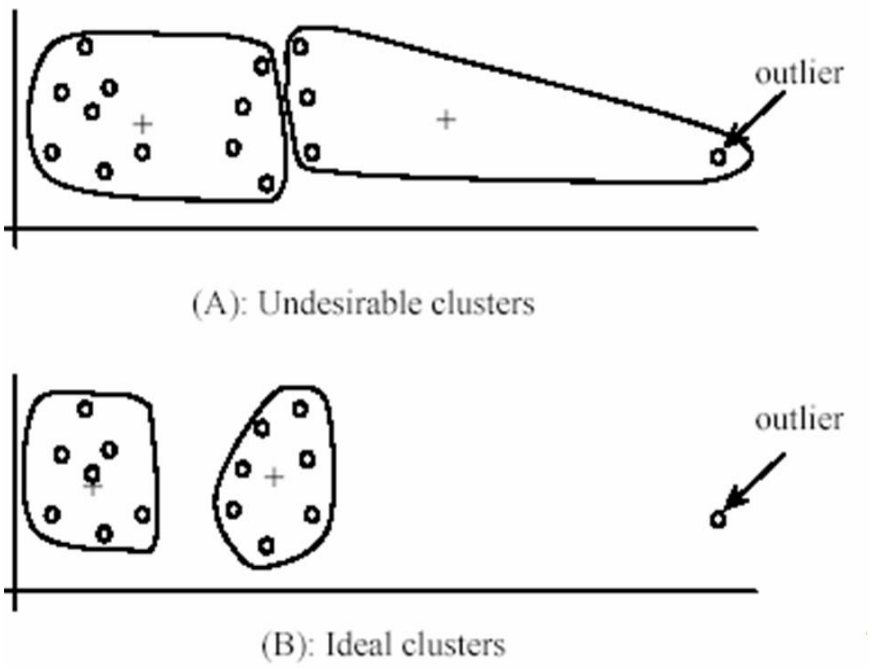
Intensity-based clusters

Color-based clusters

Grauman

K-Means pros and cons

- Pros
  - Simple and fast
  - Converges to a local min
- Cons
  - Need to pick K
  - Sensitive to initialization
  - Sensitive to outliers
  - Only finds “spherical” clusters



Some Improvements

- Split / Merge cluster centers based on their statistics
- Better center initialization

K-Means++

- Can we prevent arbitrarily bad local minima?

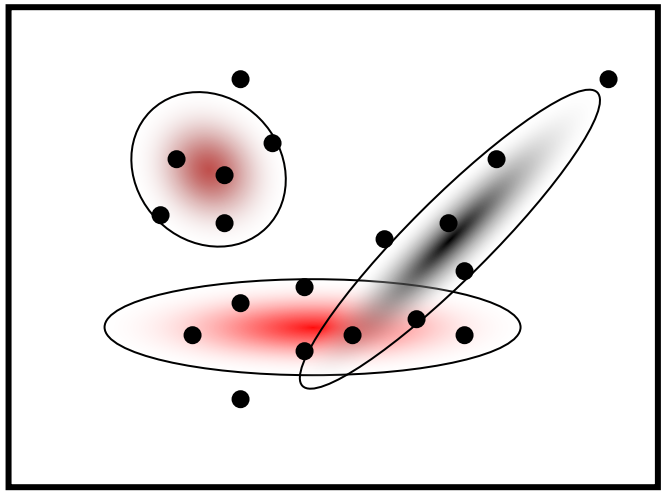
1. Randomly choose first center.
2. Pick new center with prob. proportional to: $||p - c_i||^2$
   (contribution of *p* to total error)
3. Repeat until *k* centers.

- expected error = $O(\log k)$ * optimal

- Arthur & Vassilvitskii 2007

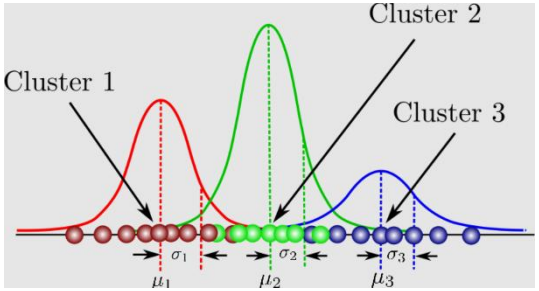
Mixture of Gaussians

K-Means	Mixture of Gaussians
Samples from Probability Density Function	Samples from Probability Density Function
Parametric model	Parametric model
Spherically symmetric distributions	Center + shape (covariance matrix)
Point association: Nearest-neighbor	Point association: Mahalanobis distance (center + shape (covariance)



Mahalanobis distance: $d(\boldsymbol{x}_i, \boldsymbol{\mu}_k; \boldsymbol{\Sigma}_k) = (\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu}_k)$

Mixture of Gaussians



Represent a dataset using a mixture of Gaussian density functions.Dataset can be

- All the intensity values in an image,
- All the color vector in an image,
- Intensity/color at a certain position (x,y) in a video
- Etc.

Mixture of Gaussians density function:

Data sample

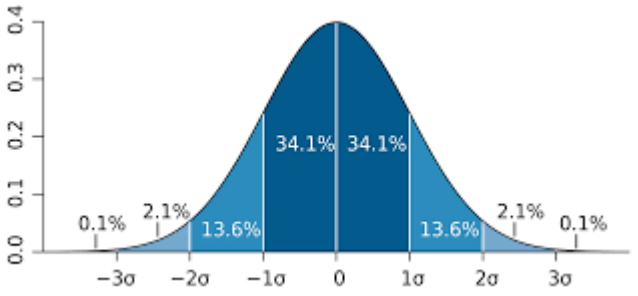
$$p(\boldsymbol{x}|\{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}) = \sum_k \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

Mixing coefficients

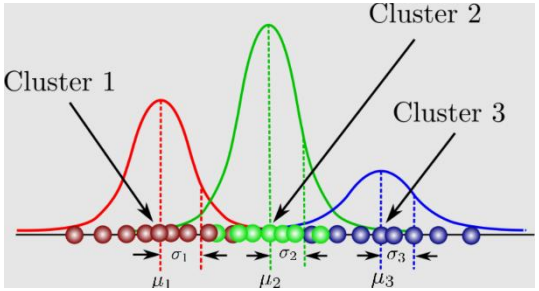
Gaussian means and covariances

Normal (Gaussian) distribution:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) = \frac{1}{|\boldsymbol{\Sigma}_k|} e^{-d(\boldsymbol{x}, \boldsymbol{\mu}_k; \boldsymbol{\Sigma}_k)}$$



Expectation Maximization (EM) algorithm



Iteratively compute (a local) maximum likely estimate for the unknown mixture parameters $\{\pi_k, \mu_k, \Sigma_k\}$

1. Expectation Stage (E step): Estimate *responsibilities* (how likely a sample x_i was generated from the k^{th} Gaussian cluster)

$$z_{ik} = \frac{1}{Z_i}\pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \quad \text{with} \quad \sum_k z_{ik} = 1,$$

2. Maximization stage (M step): update parameter values

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_i z_{ik}\boldsymbol{x}_i, \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_i z_{ik}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^T, \\
\pi_k &= \frac{N_k}{N},
\end{aligned}$$

$$N_k = \sum_i z_{ik}.$$

Estimate of number of sample points assigned to each cluster.

N: number of sample points

Applications of EM

- Turns out this is useful for all sorts of problems
  - any clustering problem
  - any model estimation problem
  - missing data problems
  - finding outliers
  - segmentation problems
    - segmentation based on color
    - segmentation based on motion
    - foreground/background separation
  - ...

Problems with EM

1. Local minima

   k-means is NP-hard even with k=2

2. Need to know number of segments

   solutions: AIC, BIC, Dirichlet process mixture

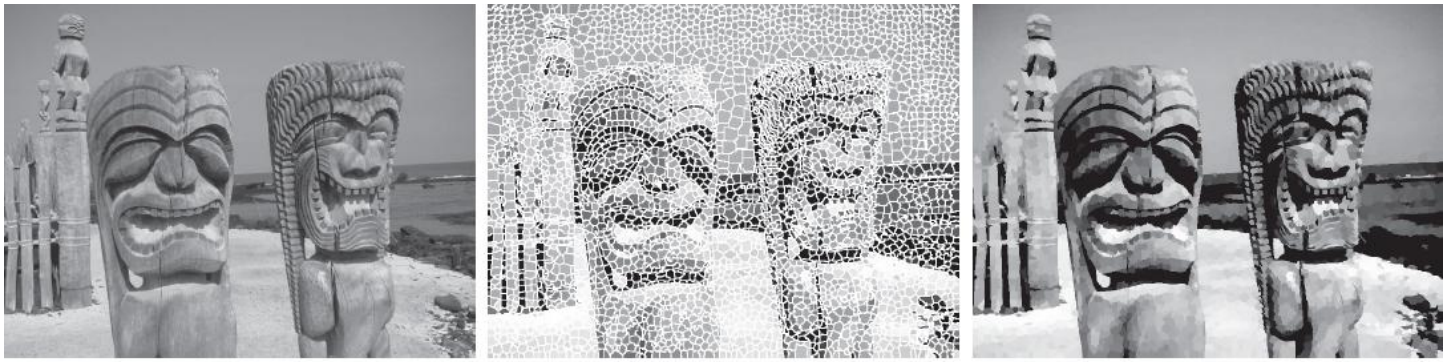
3. Need to choose generative model

Simon-Seitz

Region Segmentation using Superpixels

- The idea behind *superpixels* is to replace the standard pixel grid by grouping pixels into primitive regions that are more perceptually meaningful than individual pixels.
- The objectives are to
  - lessen computational load,
  - improve the performance of segmentation algorithms

  by reducing irrelevant detail.
- One important requirement of any superpixel representation is adherence to boundaries. This means that boundaries between regions of interest must be preserved in a superpixel image.



a b c

Gonzalez & Woods Figure 10.50

(a) Image of size 600x480 (480,000) pixels.
(b) Image composed of 4,000 superpixels (the boundaries between superpixels (in white) are superimposed on the superpixel image for reference—the boundaries are not part of the data).
(c) Superpixel image.

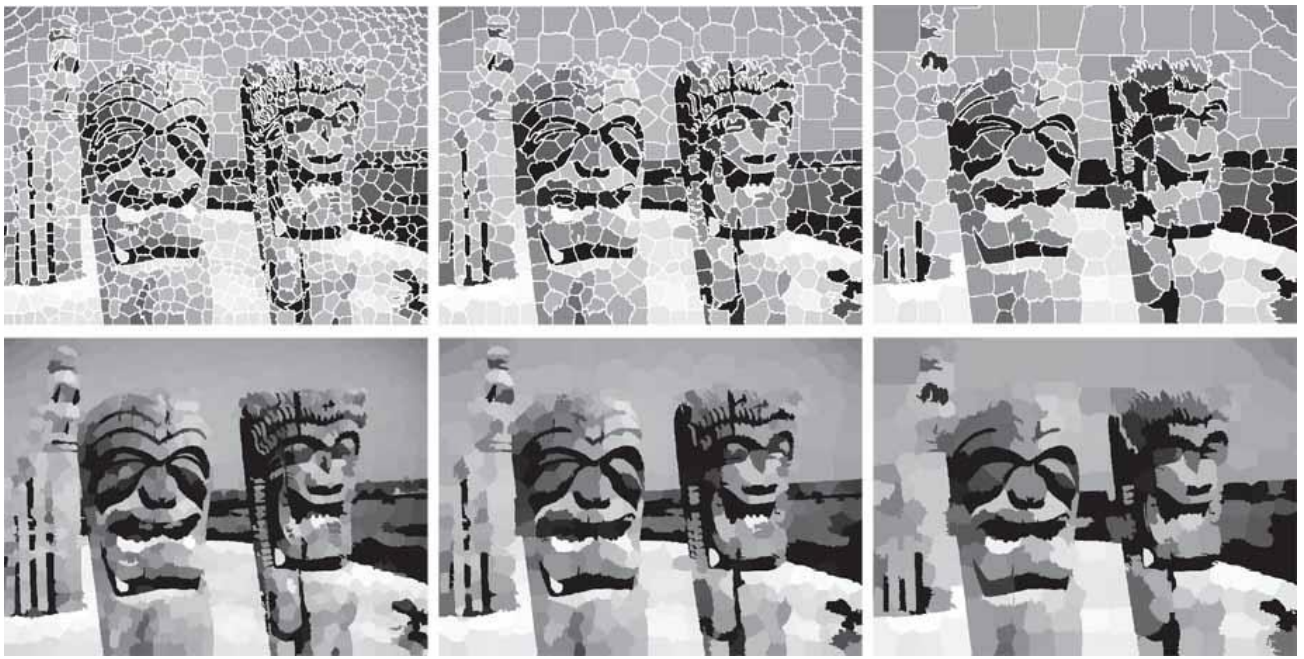
Gonzalez & Woods Figure 10.51

(a) Original image. (b) Image composed of 40,000 superpixels. (c) Difference between (a) and (b).

a b c

Gonzalez & Woods Figure 10.52

Top row: Results of using 1,000, 500, and 250 superpixels in the representation of Fig. 10.50(a). As before, the boundaries between superpixels are superimposed on the images for reference. Bottom row: Superpixel images.



SLIC (Simple Linear Iterative Clustering) Superpixel Algorithm

- Developed by **Achanta et al. [2012]**
- SLIC is a modification of the *k*-means algorithm.
- SLIC observations typically use (but are not limited to) 5-dimensional vectors containing three color components and two spatial coordinates i.e. z=[r,g,b,x,y]

$$\mathbf{z} = \begin{bmatrix} r \\ g \\ b \\ x \\ y \end{bmatrix}$$

n_{sp} denote the desired number of superpixels

n_{tp} denote the total number of pixels in the image.

initial superpixel centers, $\mathbf{m}_i = [r_i \;\; g_i \;\; b_i \;\; x_i \;\; y_i]^T$, $i = 1, 2, ..., n_{sp}$, are obtained by sampling the image on a regular grid space

$$s = [n_{tp}/n_{sp}]^{1/2}$$

1. **Initialize the algorithm:** Compute the initial superpixel cluster centers,

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

by sampling the image at regular grid steps, *s*. Move the cluster centers to the lowest gradient position in a 3×3 neighborhood. For each pixel location, *p*, in the image, set a label $L(p) = -1$ and a distance $d(p) = \infty$.

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

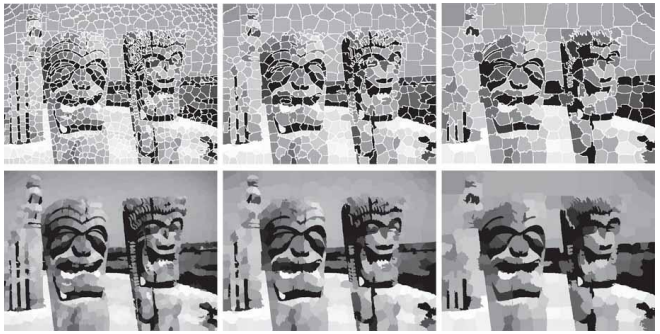
$$\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \qquad i = 1, 2, ..., n_{sp}$$

where $|C_i|$ is the number of pixels in set C_i, and the **z**'s are given by **Eq. (10-86)** .

4. **Test for convergence:** Compute the Euclidean norms of the differences between the mean vectors in the current and previous steps. Compute the residual error, *E*, as the sum of the n_{sp} norms. If $E < T$, where *T* a specified nonnegative threshold, go to Step 5. Else, go back to Step 2.
5. **Post-process the superpixel regions:** Replace all the superpixels in each region, C_i, by their average value, $\mathbf{m}_i$.

SLIC Distance measure

- Euclidean distances in CIELAB color space for more perceptually meaningful distances



$$d_{lab} = \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}$$

$$d_{xy} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

$$D_s = d_{lab} + \frac{m}{\varsigma} d_{xy} \ ,$$

Algorithm 1 Efficient superpixel segmentation

1: Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.
2: Perturb cluster centers in an $n \times n$ neighborhood, to the lowest gradient position.
3: **repeat**
4: **for** each cluster center C_k **do**
5: Assign the best matching pixels from a $2S \times 2S$ square neighborhood around the cluster center according to the distance measure (Eq. 1).
6: **end for**
7: Compute new cluster centers and residual error E {$L1$ distance between previous centers and recomputed centers}
8: **until** $E \leq$ threshold
9: Enforce connectivity.

Gonzalez & Woods Figure 10.53

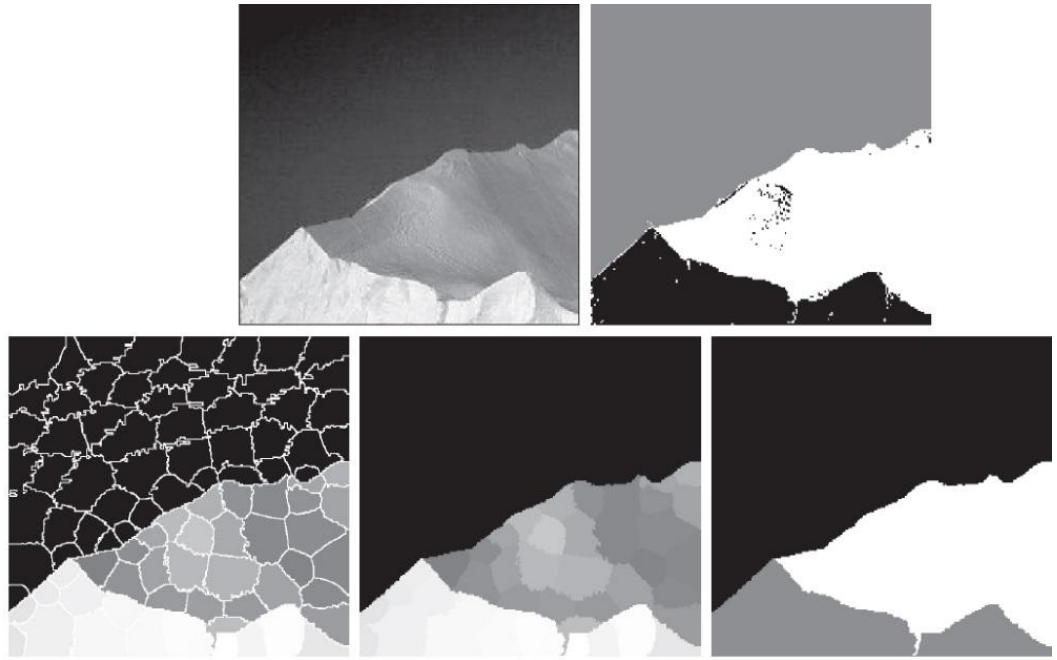
(a) Image of size 533x566 (301,678) pixels.

(b) Image segmented using the *k*-means algorithm.

(c) 100-element superpixel image showing boundaries for reference.

(d) Same image without boundaries.

(e) Superpixel image (d) segmented using the *k*-means algorithm. (Original image courtesy of NOAA.)



a b
c d e

SLIC Examples

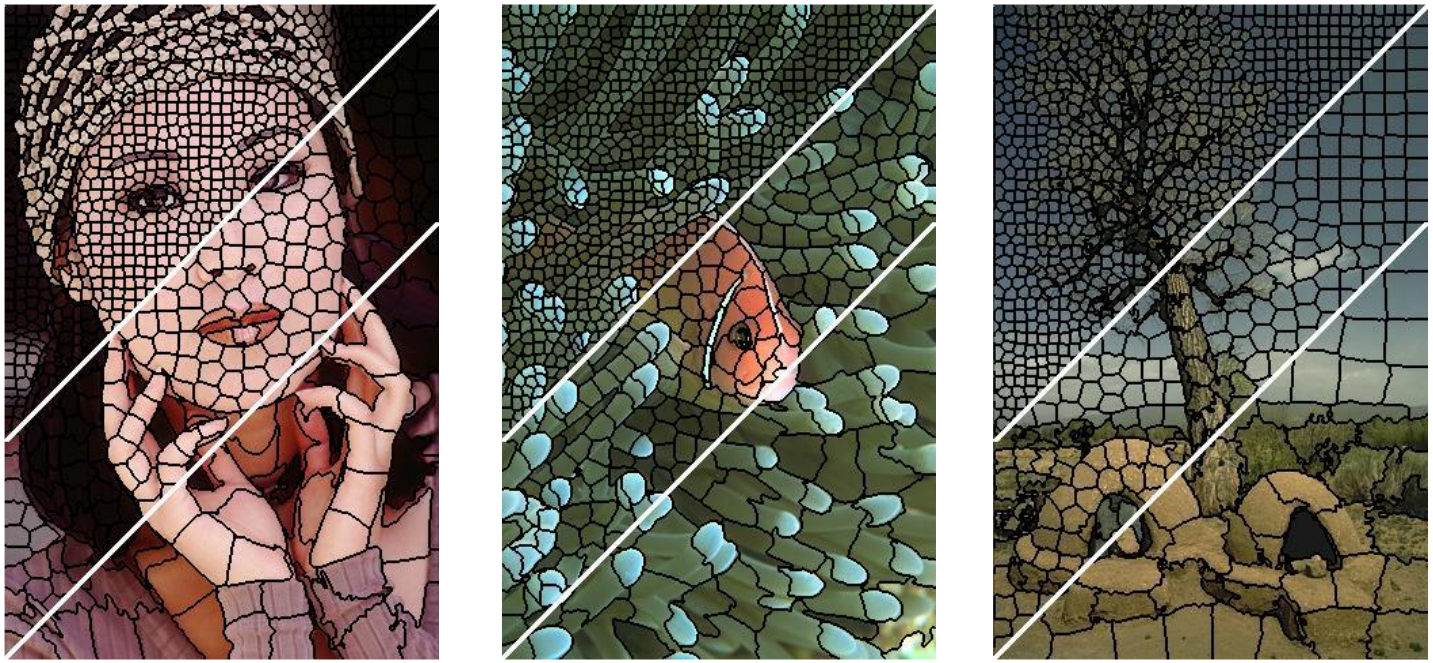


Image segmented using our algorithm into superpixels of (approximate) size 64, 256, and 1024 pixels. The superpixels are compact, uniform in size, and adhere well to region boundaries.