IN5520 – Digital Image Analysis

REGION & EDGE BASED SEGMENTATION

Fritz Albregtsen 11.11.2020

Today

We go through sections G&W DIP4E: 10.1, 10.4, 10.7, 10.8.1

We cover the following segmentation approaches:

1. Edge-based segmentation
2. Region growing
3. Region split and merge
4. Watershed segmentation
5. Segmentation by motion

Assumed known:

1. Edge detection, point and line detection (10.2)
2. Segmentation by thresholding (10.3). *I'll give you a hint!*
3. Basic mathematical morphology (9.1-9.4)

Need repetition of INF2310-stuff?

- Edge detection:
  - Read sections 10.2 and 3.6 in Gonzalez and Woods.
  - Look at INF2310 lecture on ”Filtering II” (in Norwegian).
- Thresholding:
  - Read section 10.3 in Gonzalez and Woods.
  - Go through INF2310 lecture on ”Segmentation” (in Norwegian)
- Morphology:
  - Read section 9.1-9.4 in Gonzalez and Woods.
  - Go through INF2310 lecture on ”Morphology” (in Norwegian)

What is segmentation

- Segmentation is the separation of one or more regions or objects in an image, based on **discontinuity** or **similarity** criteria.
- A region in an image can be defined by its border (edge) or its interior, and the two representations should give same result.
- If you know the interior, you can always define the border, and vice versa.
- Because of this, image segmenation approaches can typically be divided into two categories; **edge and region based methods**.

Visual segmentation is easy (?)

What if you wanted to group all the pixels corresponding to the car in this (single) image into one group. Would that be easy?

What is a region?

- A region *R* of an image *f* is defined as a connected homogenous subset of the image with respect to some criterion such as gray level or texture.
- A segmentation of an image *f* is a partition of *f* into several homogeneous regions R_i, *i=1,….m.* Each region is assumed to be a *connected* set (see G&W 2.5 – 2.5.2, pp. 68-70).
- An image *f* can be segmented into regions R_i such that:

$$1)\ f = \bigcup_{i=1}^{m} R_i$$

$$2)\ R_i \bigcap R_j = \emptyset,\ 1 \leq i, j \leq m \ and \ i \neq j$$

$$3)\ P(R_i) = \textbf{true} \ for \ all \ i$$

$$4)\ P(R_i \bigcup R_j) = \textbf{false},\ 1 \leq i, j \leq m \ and \ i \neq j \ and$$

$$R_i, R_j \ are \ adjacent$$

$P(R_i)$ is a logical predicate defined over all points in R_i. It must be true for all pixels inside the region and false for pixels in other regions.

Regions R_i and R_j are neighbors if their union forms a connected component.

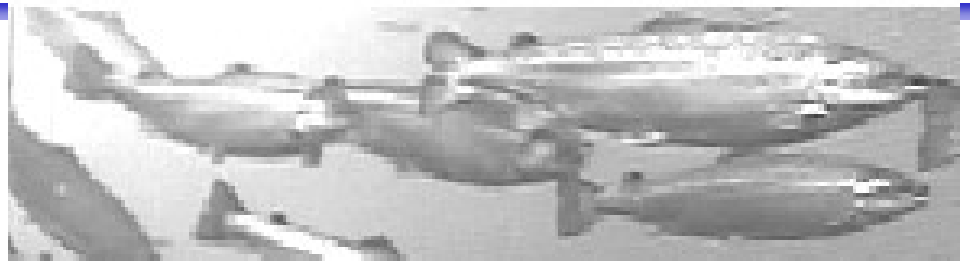
Segmentation approaches

- Pixel-based segmentation:
  each pixel is segmented based on gray-level values, no contextual information, only first order histogram.
  - Example: thresholding.

- Region-based segmentation:
  consider gray-levels from neighboring pixels by
  - including similar neighboring pixels (region growing),
  - split-and-merge,
  - or watershed segmentation.

- Edge-based segmentation:
  detect and links edge pixels to form contours.

Region vs. edge-based approaches



- Region based methods are robust because:
  - Regions cover more pixels than edges and thus you have more information available in order to characterize your region
  - When detecting a region you could for instance use texture, which is not easy when dealing with edges
  - Region growing techniques are generally better in noisy images where edges are difficult to detect
- The edge based method can be preferable because:
  - Algorithms are usually less complex
  - Edges are important features in an image to separate regions
- The edge of a region can often be hard to find because of noise or occlusions
- Combination of approaches and of results may often be a good idea !

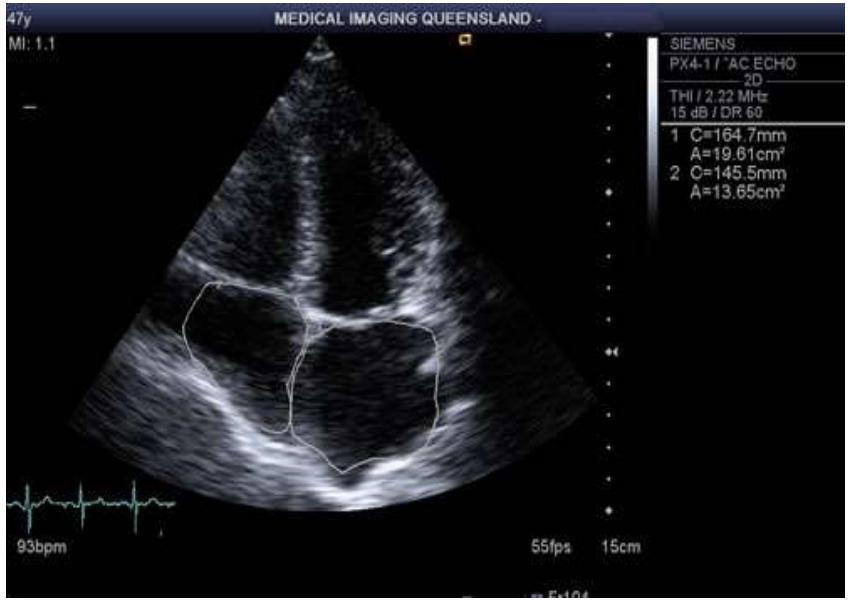
Edge-based segmentation

Two steps are needed:

1. Edge detection (to identify "edgels" - edge pixels)
   - (Gradient, Laplacian, LoG, Canny filtering)

2. Edge linking – linking adjacent "edgels" into edges
   - *Local Processing*
     - *magnitude* of the gradient
     - *direction* of the gradient vector
     - edges in a predefined neighborhood are linked if both magnitude and direction criteria is satisfied
   - *Global Processing*
     - *Hough Transform*

Edge detection

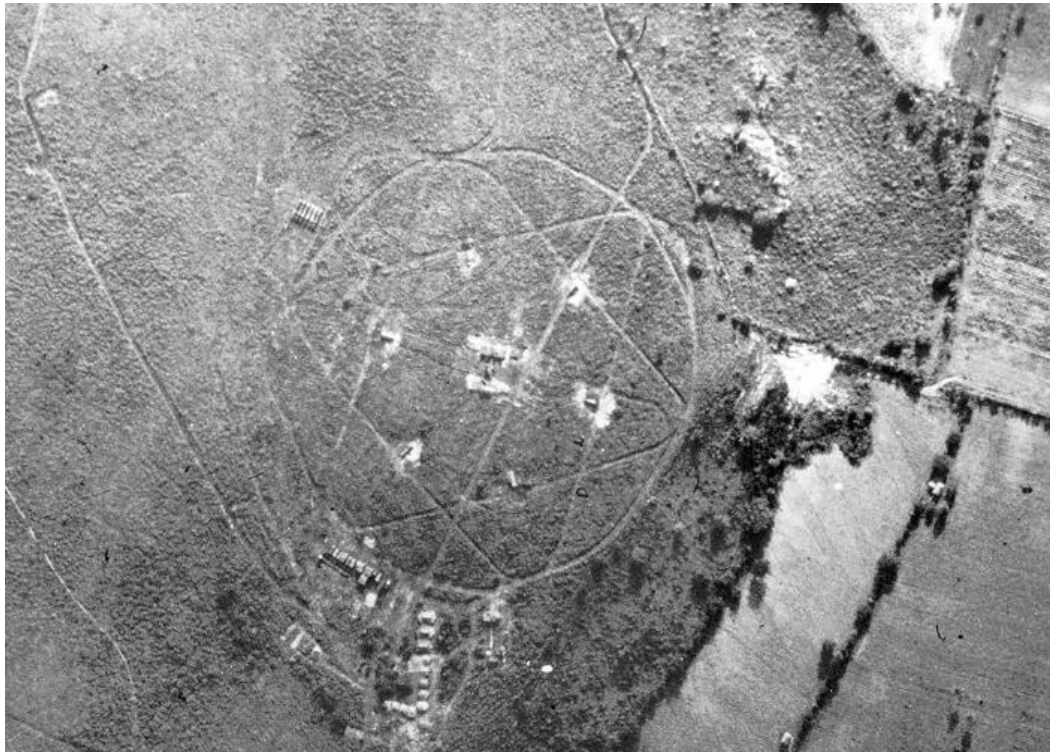


- A large group of methods.
- Operators detecting discontinuities
  - in gray level, color, texture, etc.
- Results can seldom be used directly.
- Post processing steps must follow, combining edges into edge chains.
- The more prior information used in the segmentation process, the better the segmentation results can be obtained
- The most common problems of edge-based segmentation are:
  - edge presence in locations where there is no border
  - no edge presence where a real border exists

Why is a gradient operator not enough?

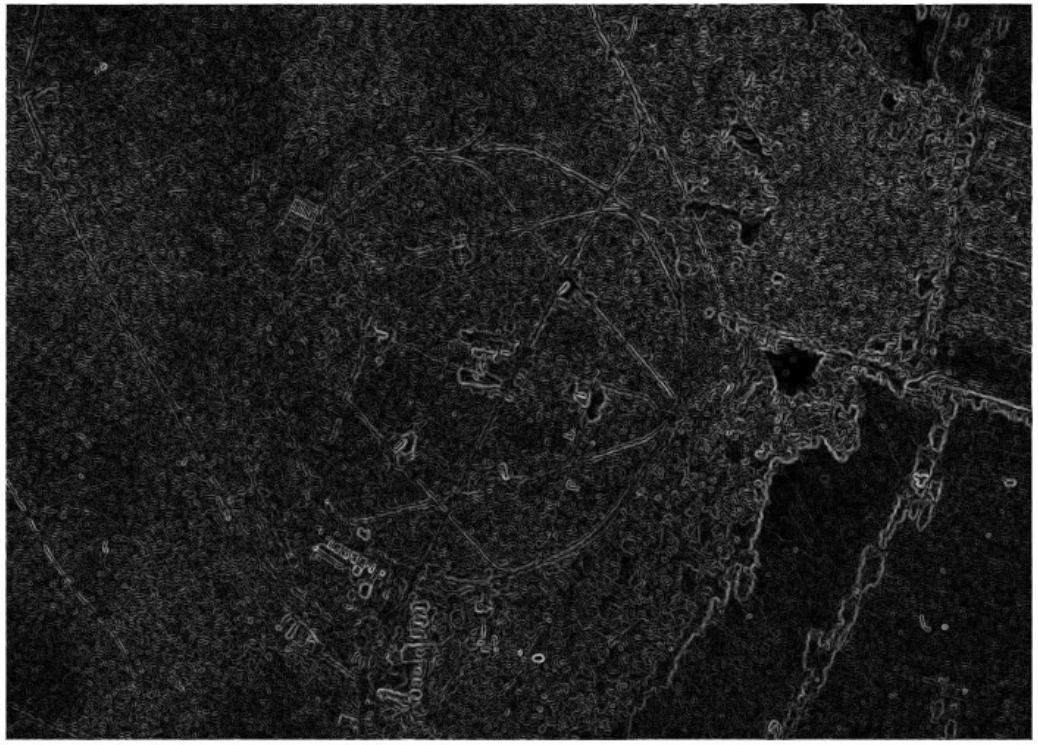
- Some images produce nice gradient maps.
- Most interesting/challenging images produce a very complicated gradient map
  - e.g. from a Sobel filter.
- Only rarely will the gradient magnitude be zero, … why?
- Distribution of gradient magnitude is often skewed, … why?
- How do we determine gradient magnitude threshold, … ?

A challenging graylevel image



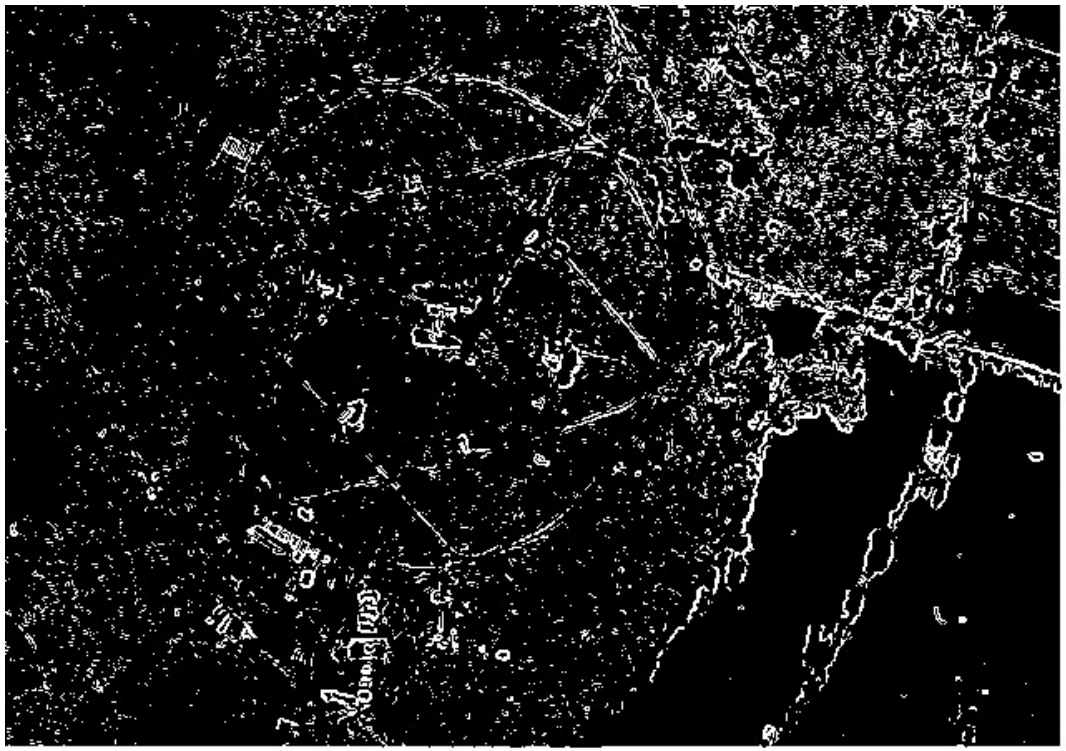
Ground to air missiles near Havana, Cuba.
Imaged by U2 aircraft 29.08.1962.

The gradient magnitude image



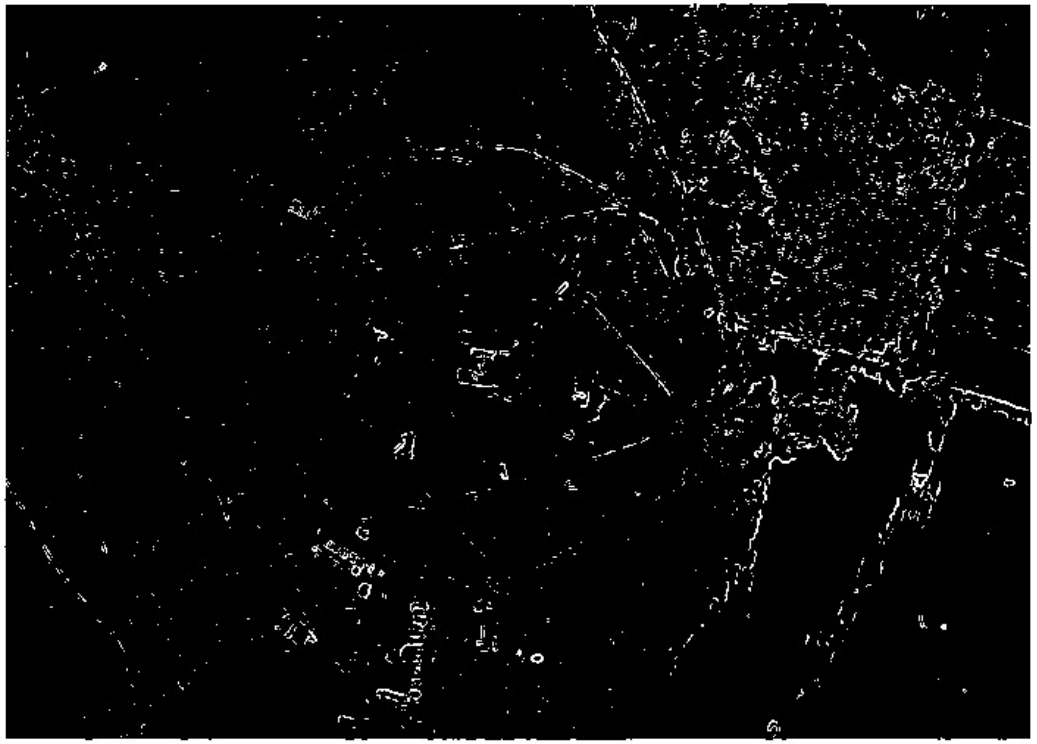
Magnitude of gradient image, min=0, max=1011

Edges from thresholding - I



Gradient magnitude thresholded at 250

Edges from thresholding - II



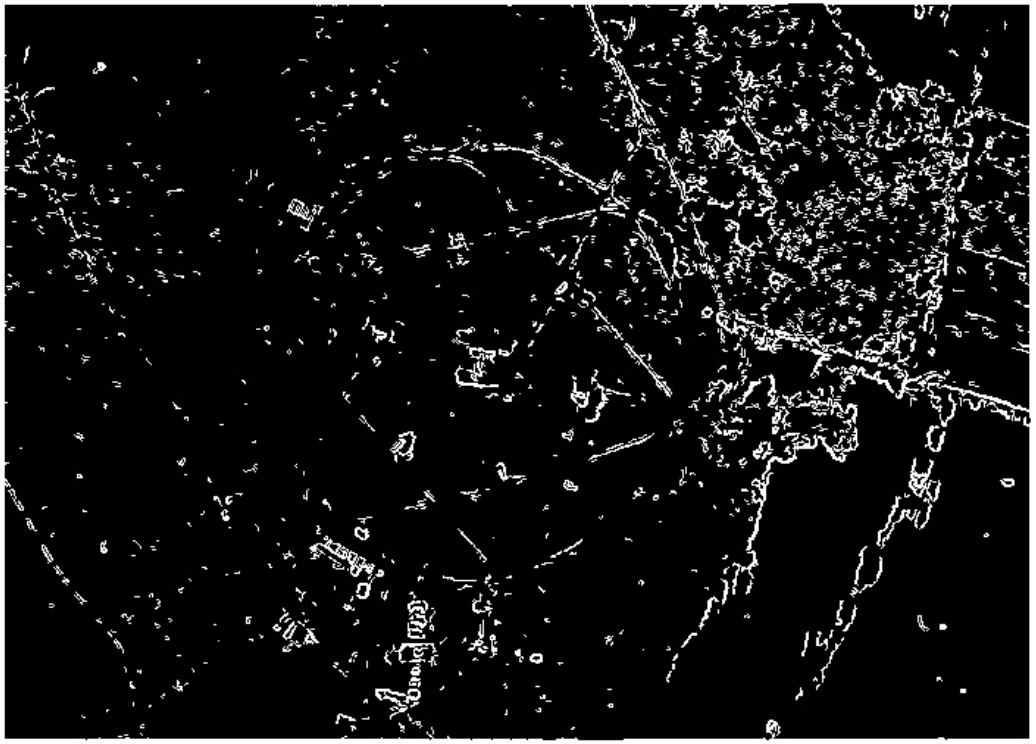
Gradient magnitude thresholded at 350

Improved edge detection

- What if we assume the following:
  - All gradient magnitudes above a high *strict* threshold are assumed to belong to a *bona fide* edge.
  - All gradient magnitudes above a lower *unstrict* threshold and connected to a pixel resulting from the strict threshold are also assumed to belong to real edges.

- This is "hysteresis thresholding"
  - Used in e.g., Canny's edge detection (see INF 2310).

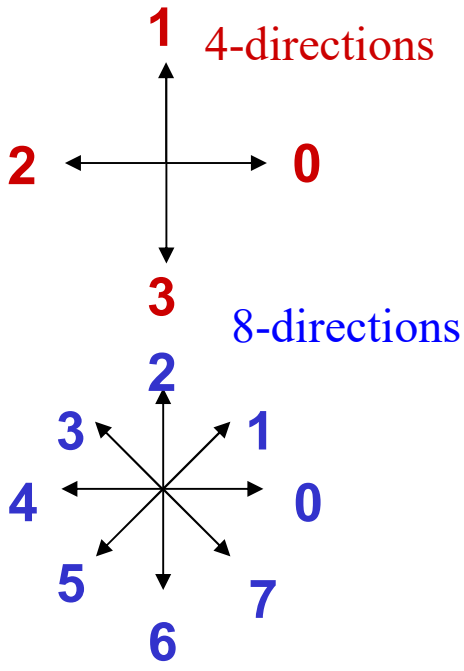
Edges from hysteresis thresholding



Result of hysteresis, are we really impressed?

Thinning of edges

1. Quantize the edge directions into four (or eight) directions.
2. For all nonzero gradient magnitude pixels, inspect the two neighboring pixels in the four (or eight) directions.
3. If the edge magnitude of any of these neighbors is higher than that under consideration, mark the pixel.
4. When all pixels have been scanned, delete or suppress the marked pixels.

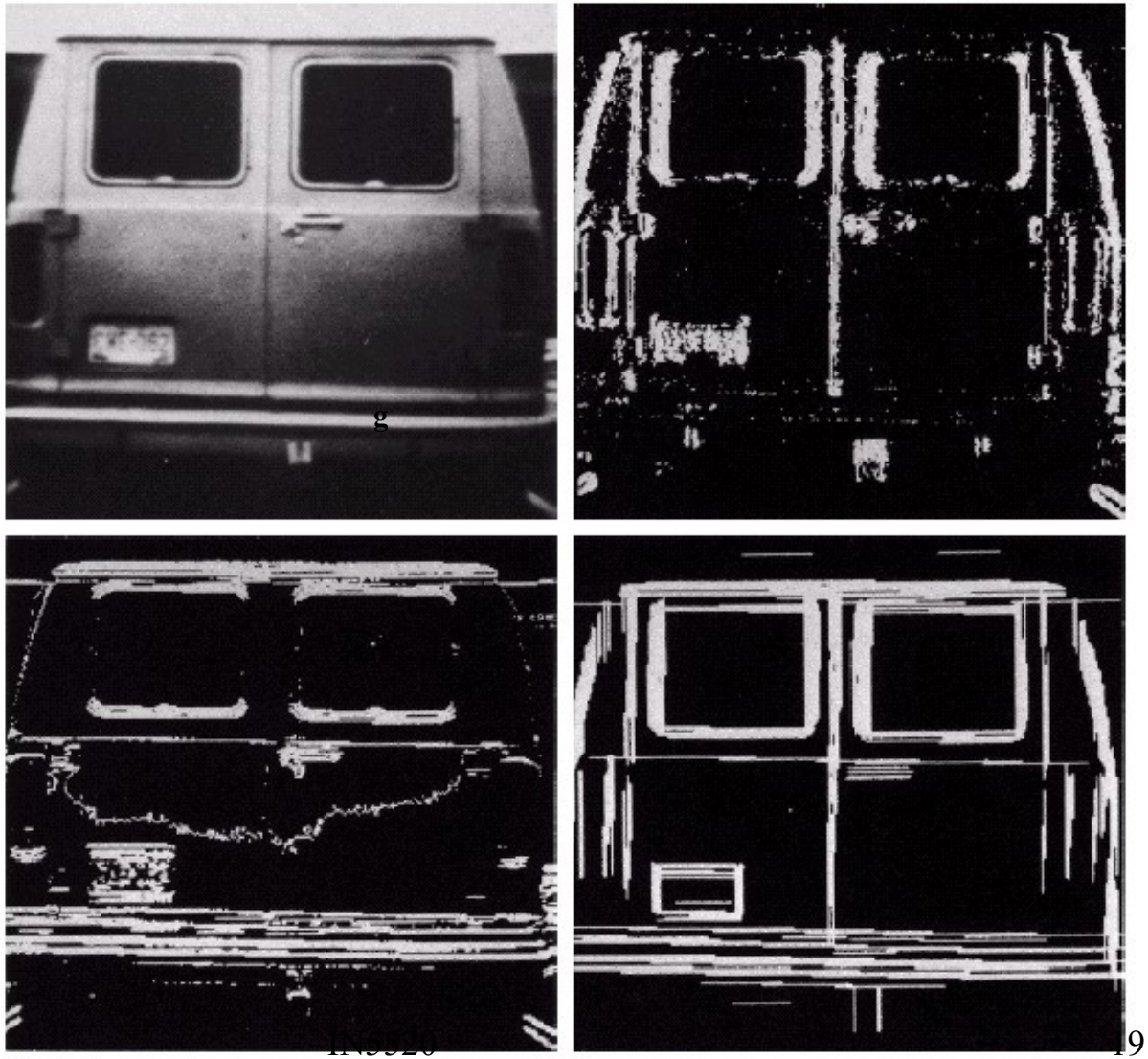


Used iteratively in "nonmaxima suppression" (e.g. Canny)

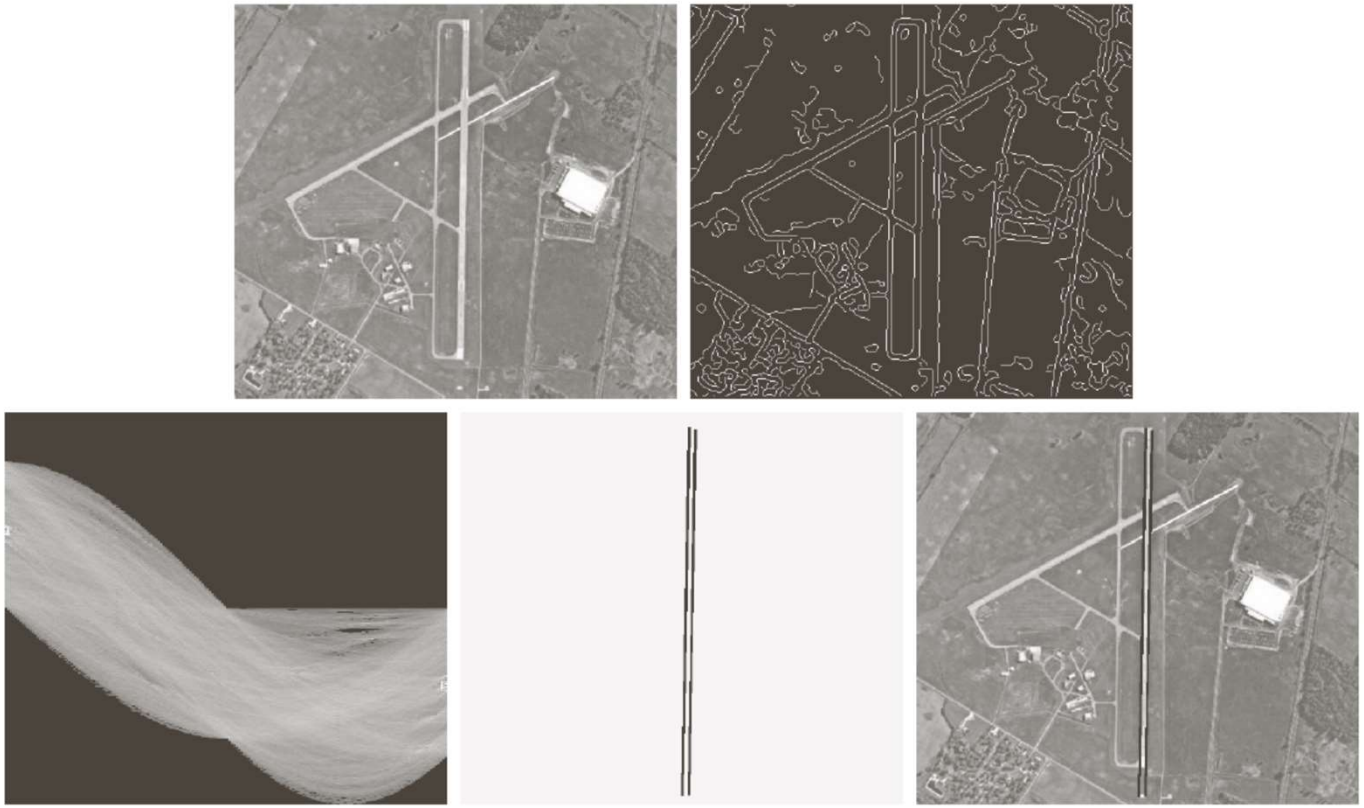
Local edge linking

Left: Input image
Right: G_y

Left: G_x
Right: Result after edge linking



Using HT for edge linking



a b
c d e

FIGURE 10.34 (a) A 502 × 564 aerial image of an airport. (b) Edge image obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes). (e) Lines superimposed on the original image.

Σ: Edge based segmentation

- **Advantages :**
  - An approach similar to how humans segment images.
  - Works well in images with good contrast between object and background

- **Disadvantages :**
  - Does not work well on images with smooth transitions and low contrast
  - Sensitive to noise
  - Robust edge linking is not trivial

Region growing (G&W:10.4.1)

- Grow regions by recursively including the neighboring pixels that are similar and connected to the seed pixel.
- Connectivity is needed not to connect pixels in different parts of the image.
- Similarity criterion - examples:
  - Use difference in gray levels for regions with homogeneous gray levels
  - Use texture features for textured regions

Similarity: gray level difference

Cannot be segmented using gray level similarity.

Region growing

- Starts with a set of seeds (starting pixels)
  - Predefined seeds
  - All pixels as seeds
  - Randomly chosen seeds

- Region growing steps (bottom-up method)
  - Find starting points
  - Include neighboring pixels with similar features (graylevel, texture, color) A similarity criterion must be selected.

Two variants:

1. Select seeds from the whole range of grey levels in the image. Grow regions until all pixels in image belong to a region.
2. Select seed only from objects of interest (e.g. bright structures). Grow regions only as long as the similarity criterion is fulfilled.

- Problems:
  - Not trivial to find good starting points
  - Need good criteria for similarity

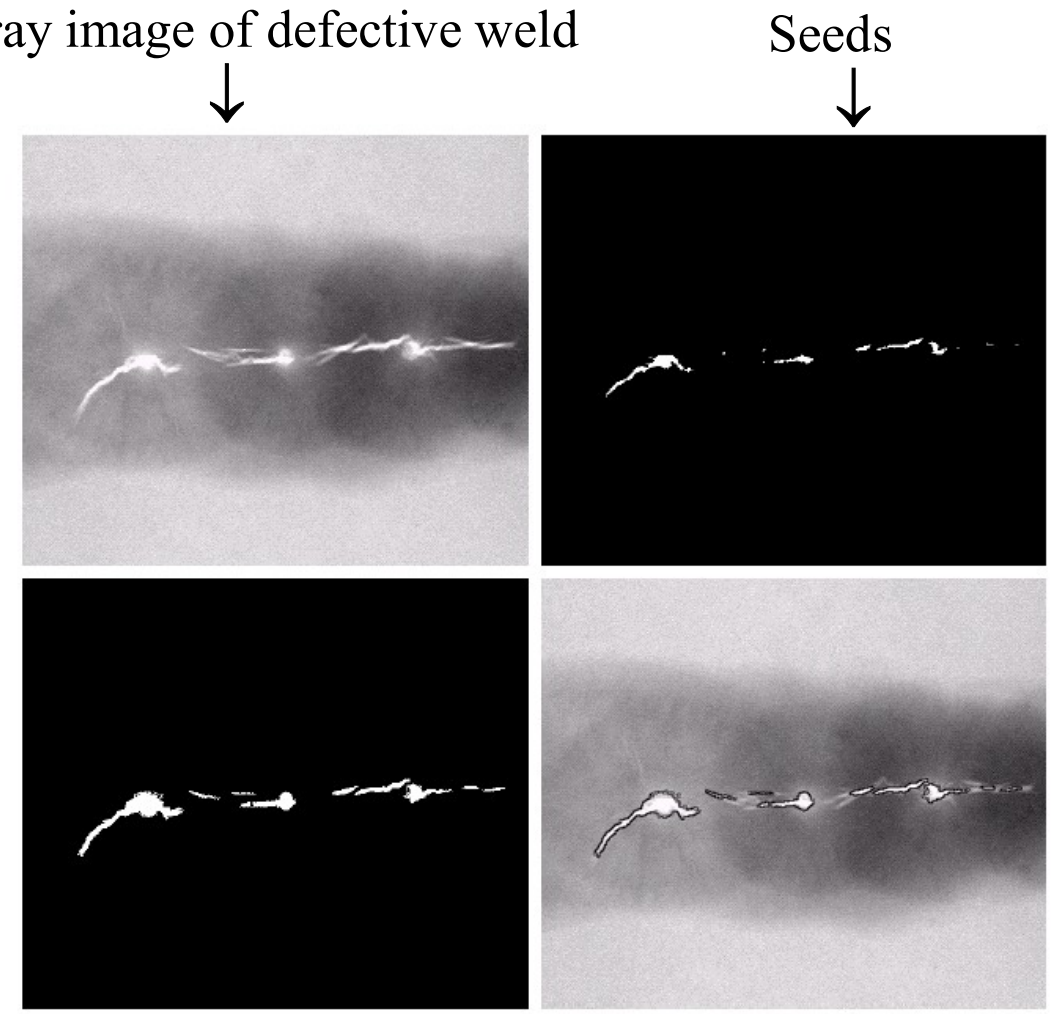
Region growing example

- Seeds: f(x,y) = 255
- P(R) = TRUE if
  | seed gray level – new pixel gray level | < 65

  and

  – New pixel must be 8-connected with at least one pixel in the region

Result of region growing →



Similarity measures

- Graylevel or color?
- Graylevel and spatial properties, e.g., texture, shape, …
- Intensity difference within a region
  (from pixel to seed, or from pixel to region average so far?)
- Within a value range (min, max)
- Distance between mean value of the regions
  (specially for region merging or splitting)
- Variance or standard deviation within a region.

Region merging techniques

- Initialize by giving all the pixels a unique label
  - All pixels in the image are assigned to a region.

- The rest of the algorithm is as follows:
  - In some predefined order, examine the neighbor regions of all regions and decide if the predicate you have chosen evaluates to true, and do this for all pairs of neighboring regions.
  - If the predicate evaluates to true for a given pair of neighboring regions then give these neighbors the same label.
  - The predicate is the similarity measure (can be defined based on e.g. region mean values or region min/max etc.).
  - Continue until no more mergings are possible.
  - Upon termination all region criteria will be satisfied.

Region merging techniques

- The aim is to separate the apples from the background.
- This image poses more of a challenge than you might think.

Region merging techniques

- We run a standard region merging procedure where all pixels initially are given a unique label.
- If neighboring regions have mean values within 10 gray levels they are fused.
- Regions are considered neighbors in 8-connectivity.
- Q: Typical errors ?

Did Andy merge Marilyn?

Region merging techniques

- Initialization is critical; results will in general depend on the initialization.
- The order in which the regions are treated will also influence the result:
  - The top image was flipped upside down before it was fed to the merging algorithm.
- Notice the differences & similarities, not in color, but in shape and area!

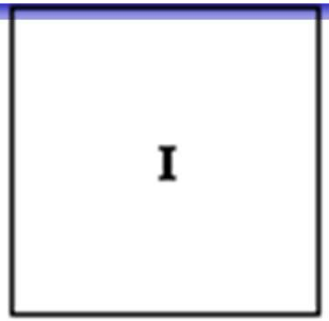
Split and merge (G&W:10.4.2)

- Separate the image into regions based on a given similarity measure. Then merge regions based on the same or a different similarity measure.

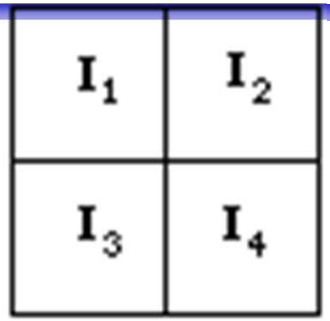
- The method is also called "quad tree" division:

  Set up some criteria for what is a uniform area

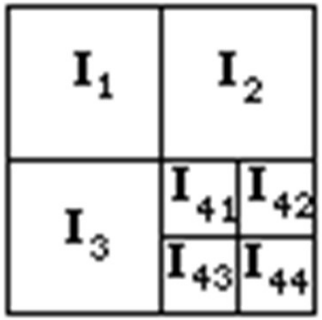
  1. Start with the full image and **split** it in to 4 sub-images.
  2. Check each sub-image.
     If not uniform, divide into 4 new sub-images
  3. After each iteration, compare neighboring regions and **merge** if uniform according to the similarity measure.



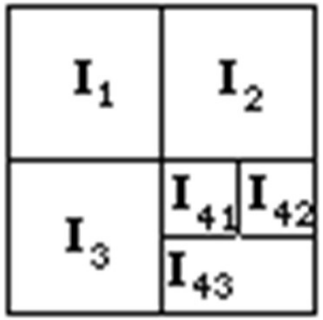
(a) Whole Image



(b) First Split

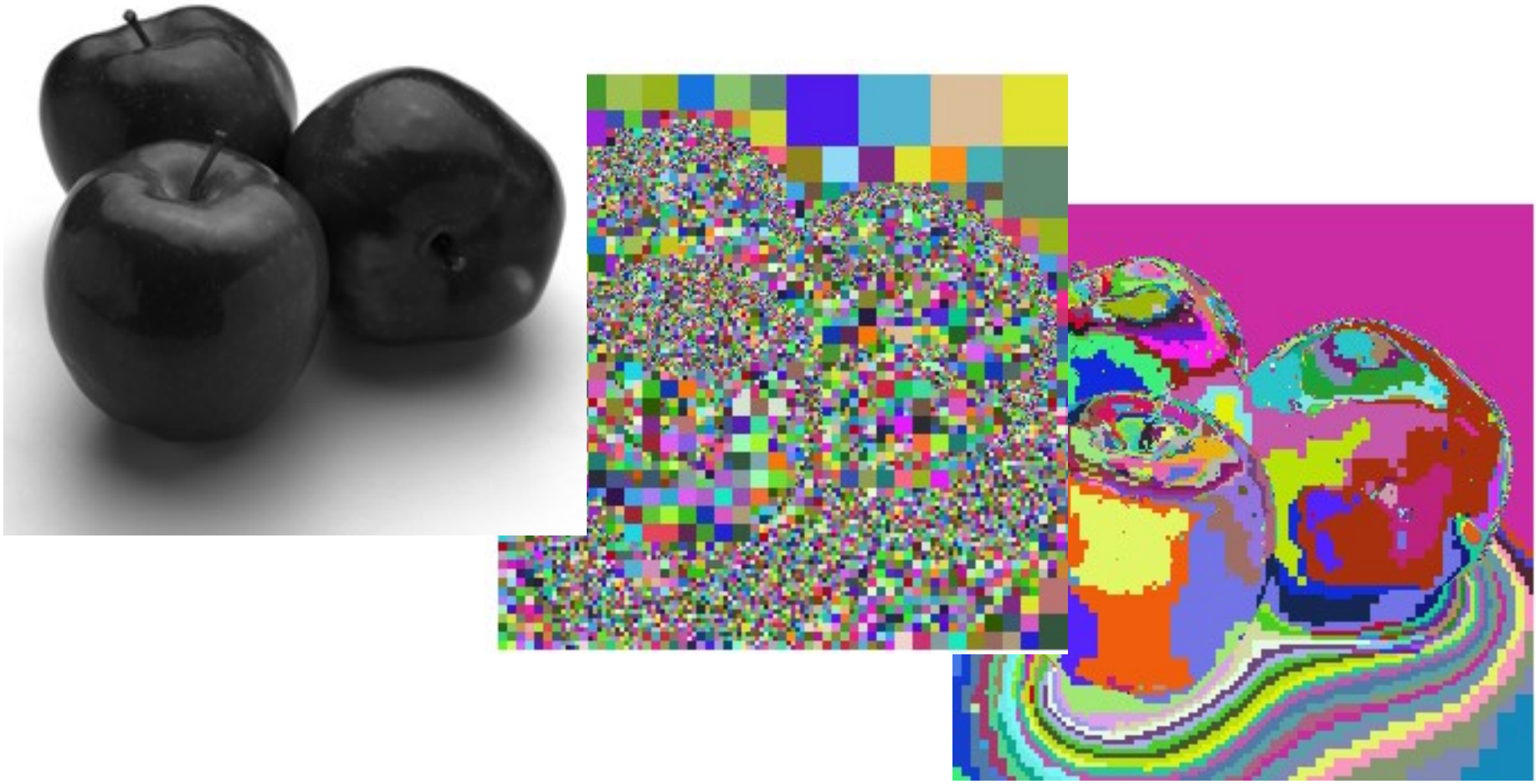


(c) Second Split



(d) Merge

Split and merge example



«Default» thresholding: Otsu's method

- Given a NxM pixel image with G graylevels.
- Find image histogram, h(k), k= 0,1,2,..,G-1.
- Find normalised histogram:

$$p(k) = \frac{h(k)}{MN}, \quad k = 0,1,2,...,G-1$$

- Compute cumulative normalised histogram:

$$P_1(k) = \sum_{i=0}^{k} p(i), \quad k = 0,1,2,...,G-1$$

- Compute cumulative mean value, μ(k):

$$\mu(k) \equiv \sum_{i=0}^{k} ip(i), \quad k = 0,1,2,...,G-1$$

- Compute global mean value, μ:

$$\mu \equiv \sum_{i=0}^{G-1} ip(i)$$

- Compute variance between classes, $\sigma_B^2(k)$:

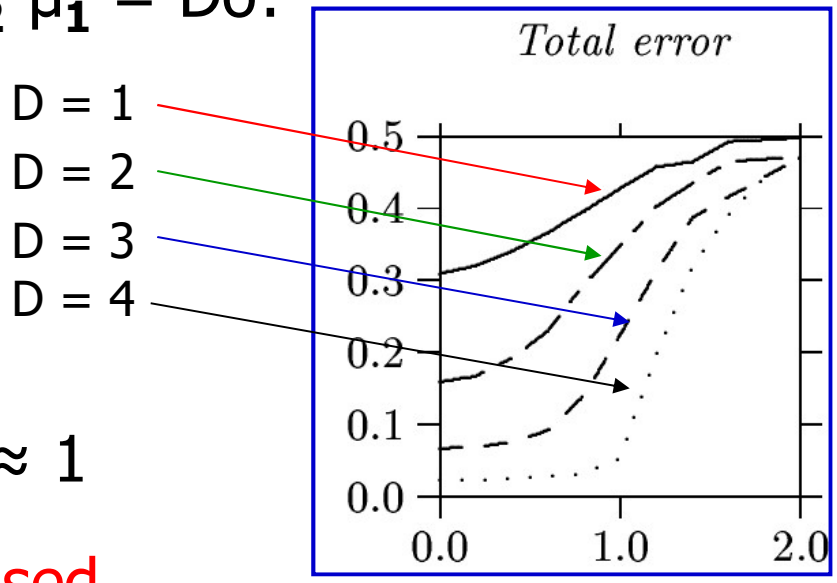
$$\sigma_B^2(t) = \frac{[\mu(t) - \mu P_1(t)]^2}{P_1(t)(1 - P_1(t))}$$

- Find threshold where $\sigma_B^2(k)$ has its maximum.
- Compute separability measure, η(t):

$$\eta(t) = \frac{\sigma_B^2(t)}{\sigma_{Tot}^2}, \quad 0 \le \eta(t) \le 1$$

Histogram-based thresholding (Otsu)

- Total classification error vs $\log_{10}(P_1/P_2)$ for four values of $\mu_2-\mu_1 = D\sigma$:



- Error increases around $\log_{10}(P_1/P_2) \approx 1$
- => Otsu's method only to be used when $0.1<P_1/P_2<10$.
- Better alternatives exist !

”Minimum error” thresholding

- Kittler and Illingworth (1985) assume a mixture of two Gaussians.
- Find t that minimizes the Kullback-Leibler (KL) distance between observed histogram and model distribution :

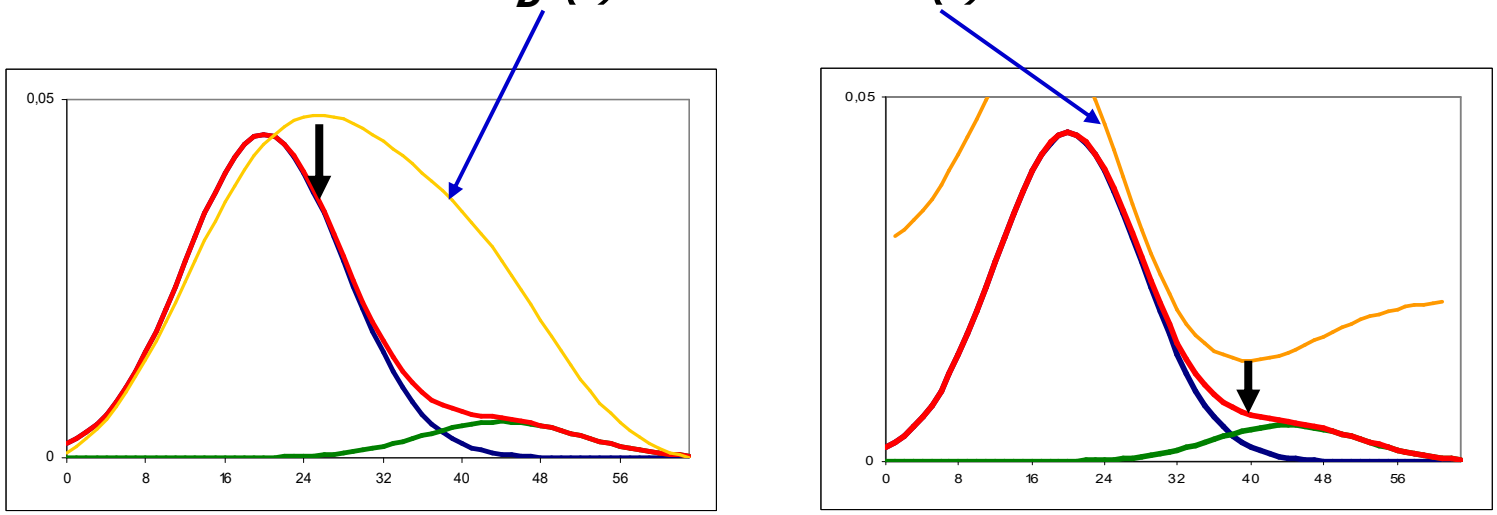
$$J(t) = 1 + 2\left[P_1(t)\ln\sigma_1(t) + P_2(t)\ln\sigma_2(t)\right] - 2\left[P_1(t)\ln P_1(t) + P_2(t)\ln P_2(t)\right]$$

- As t varies, all five Gaussian model parameters change.
- Compute J(t) for all t; then find minimum.
- Criterion function may have minima at boundaries of the gray scale.
  - An unfortunate starting value for an iterative search may cause the iteration to terminate at a nonsensical threshold value.
  - Use Otsu’s threshold as starting value for iterative search.

A comparison example

- For B =0.9, F =0.1, μ_B = 20, μ_F = 44, σ_F = σ_B = 8:
- Otsu's threshold (left) gives significant error in t.
- Kittler and Illingworth's threshold (right) is OK.

Criterion functions: $\sigma_B^2(t)$ and $J(t)$



Watershed thresholding – the idea

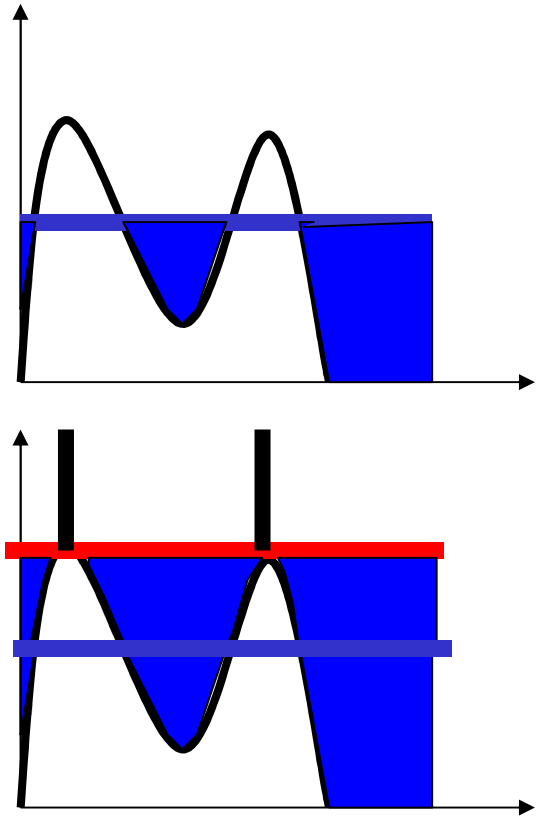
- A gray level image
  (or a gradient magnitude image
  or some other feature image)
  may be seen as a topographic relief,
  where increasing pixel value
  is interpreted as increasing height.

- Drops of water falling on a topographic relief
  will flow along paths to end up in local minima.

- The watersheds of a relief correspond to the limits of adjacent catchment basins of all the drops of water.

Watershed segmentation (G&W:10.5)

- Look at the image as a 3D topographic surface, (x,y,intensity), with both local valleys and mountains.
- Assume that there is a hole at each minimum, and that the surface is immersed into a lake.
- The water will enter through the holes at the minima and flood the surface.
- To avoid that two different basins merge, a dam is built.
- Final step: the only thing visible would be the dams.
- The connected dam boundaries correspond to the watershed lines.



Watershed segmentation

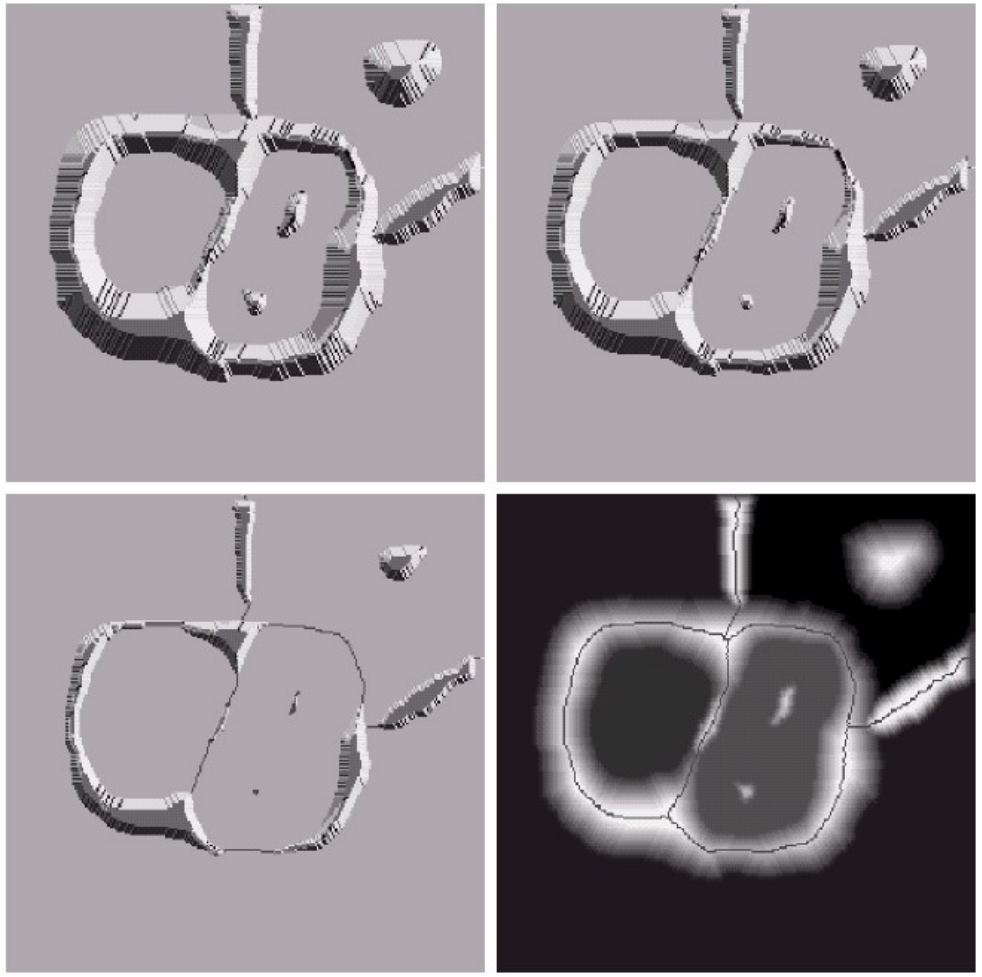
Original image

Original, topographic view

Water rises and fills the dark background

Water now fills one of the dark regions

Watershed segmentation



The two basins are about to meet, dam construction starts

Final segmentation result overlaid on original image.

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

Watershed segmentation

- Can be used on images derived from:
  - The intensity image
  - Edge enhanced image
  - Distance transformed image (e.g. distance from object edge)
    - Thresholded image.
    - From each foreground pixel,
      compute the distance to a background pixel.
  - Gradient of the image
- Most common basis of WS: gradient image.

Watershed algorithm

- Let g(x,y) be the input image (often a gradient image).
- Let $M_1,...M_R$ be the coordinates of the regional minima.
- Let $C(M_i)$ be a set consisting of the coordinates of all points belonging to the catchment basin associated with the regional mimimum M_i.
- Let T[n] be the set of coordinates (s,t) where g(s,t)<n

$$T[n] = \{(s,t) \mid g(s,t) < n\}$$

This is the set of coordinates lying below the plane g(x,y)=n

These are candidate pixels for inclusion into the catchment basin, but we must take care that the pixels do not belong to a different catchment basin.

Watershed algorithm cont.

- The topography will be flooded with integer flood increments from n=min-1 to n=max+1.

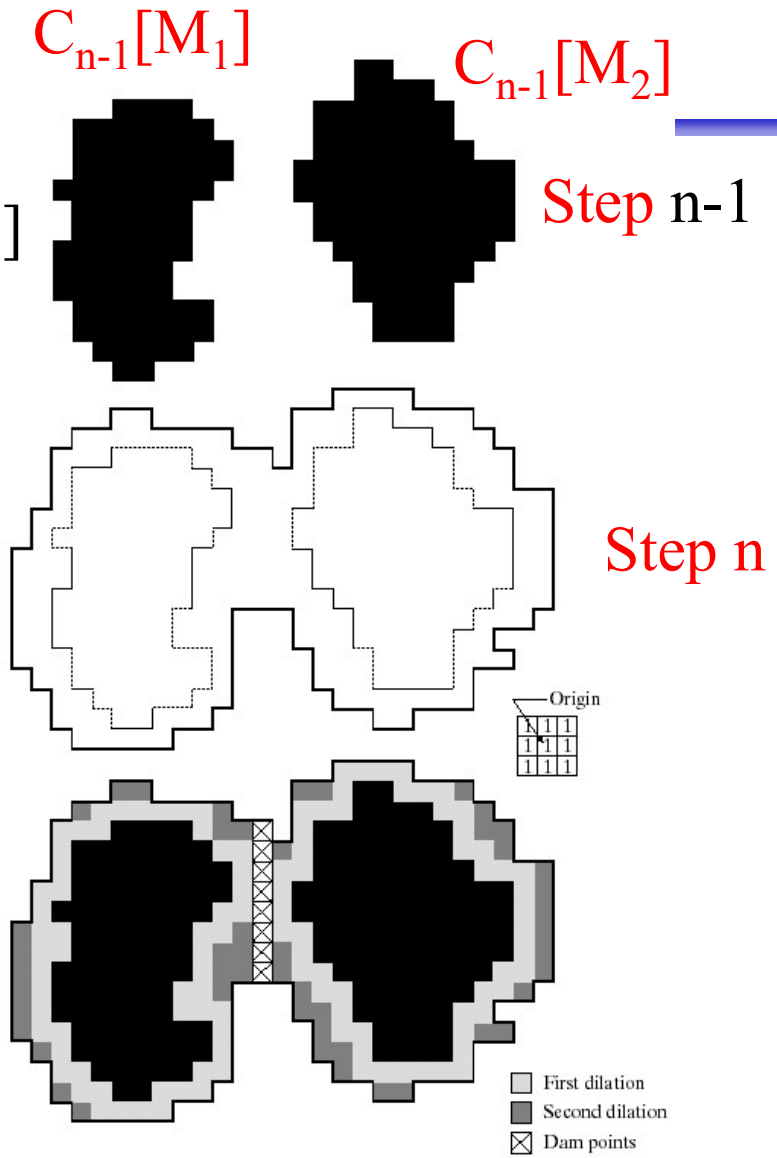
- Let $C_n(M_i)$ be the set of coordinates of points in the catchment basin associated with M_i, flooded at stage n.
- This must be a connected component and can be expressed as $C_n(M_i) = C(M_i) \cap T[n]$ (only the portion of T[n] associated with basin M_i)

- Let C[n] be union of all flooded catchments at stage n:

$$C[n] = \bigcup_{i=1}^{R} C_n(M_i) \quad \text{and} \quad C[\max+1] = \bigcup_{i=1}^{R} C(M_i)$$

Dam construction

- Stage n-1: two basins forming separate connected components.
- To consider pixels for inclusion in basin k in the next step (after flooding), they must be part of T[n], and also be part of the connected component **q** of T[n] that $C_{n-1}[k]$ is included in.
- Use morphological dilation iteratively.
- Dilation of C[n-1] is constrained to q.
- The dilation can not be performed on pixels that would cause two basins to be merged (form a single connected component)



Watershed algorithm cont.

- Initialization: let C[min+1]=T[min+1]

- Then recursively compute C[n] from C[n-1]:
  - Let Q be the set of connected components in T[n].
  - For each component q in Q, there are three possibilities:

  1. $q \cap C[n-1]$ is empty – new minimum
     Combine q with C[n-1] to form C[n].

  2. $q \cap C[n-1]$ contains one connected component of C[n-1]
     q lies in the catchment basin of a regional minimum
     Combine q with C[n-1] to form C[n].

Watershed alg.

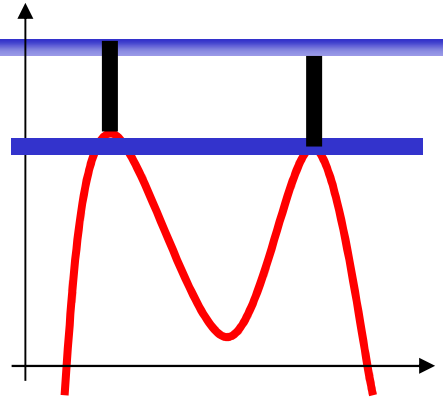
3. $q \cap C[n-1]$ contains more than one connected component of $C[n-1]$

   q is then on a ridge between catchment basins, and a dam must be built to prevent overflow.

   Construct a one-pixel dam by dilating $q \cap C[n-1]$ with a 3x3 structuring element, and constrain the dilation to q.

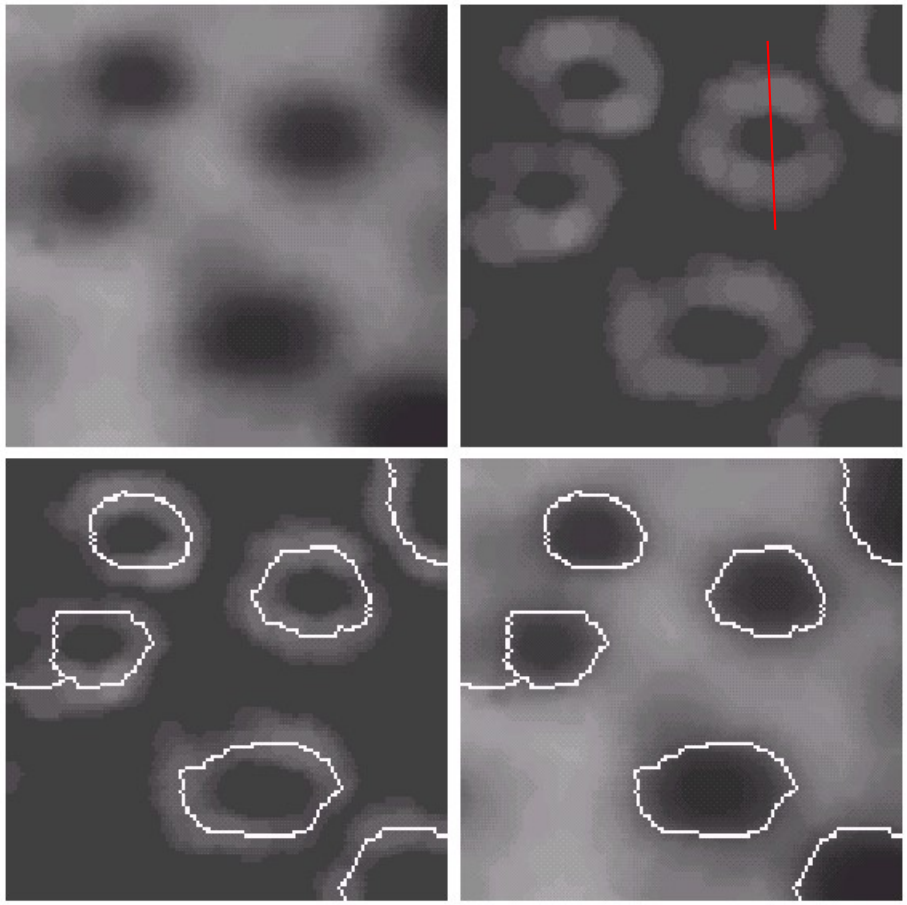
- Constrain n to existing intensity values of g(x,y) (obtain them from the histogram).

Watershed example



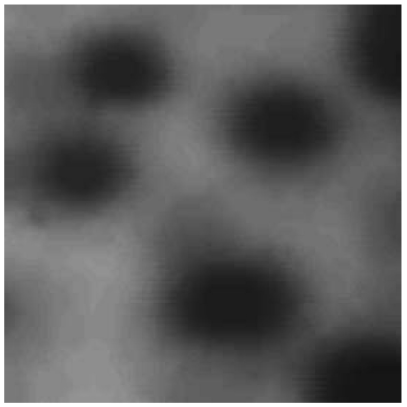
a b
c d

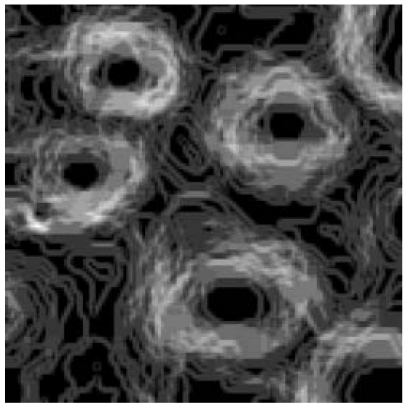
FIGURE 10.46
(a) Image of blobs. (b) Image gradient. (c) Watershed lines. (d) Watershed lines superimposed on original image. (Courtesy of Dr. S. Beucher, CMM/Ecole des Mines de Paris.)



"Over-segmentation" or fragmentation

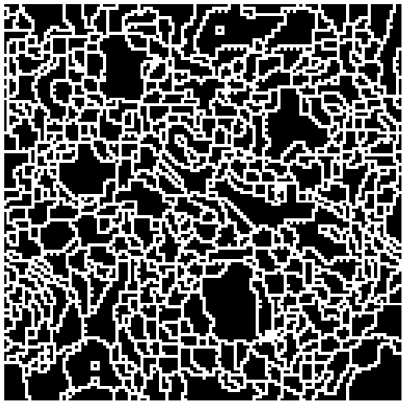
Image I

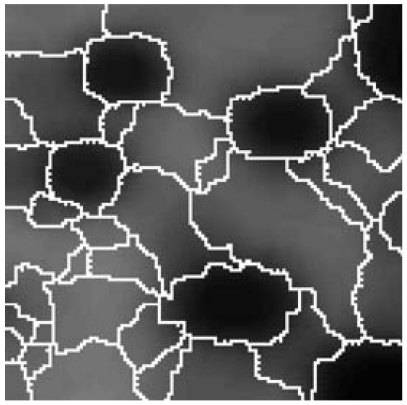




Gradient magnitude image (g)

Watershed of g

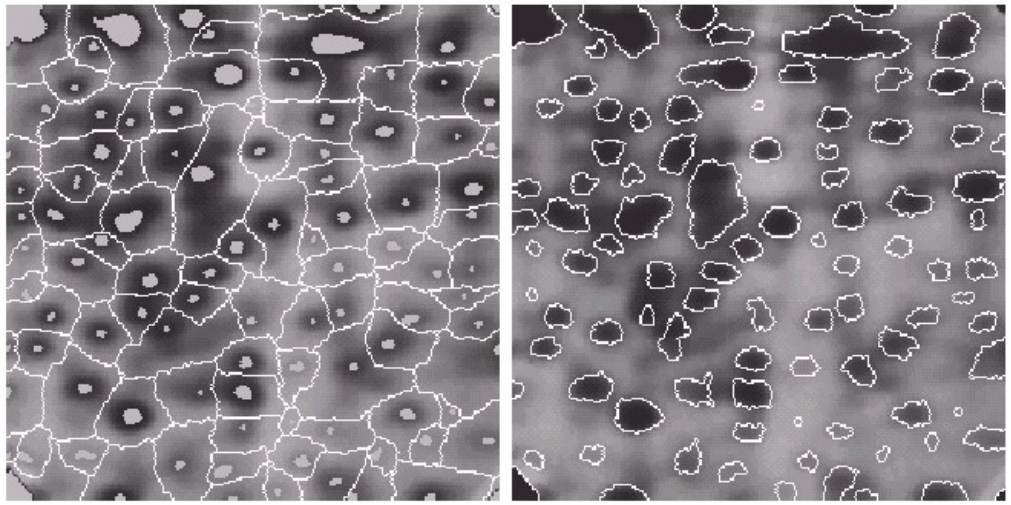




Watershed of smoothed g

- Using the gradient image directly can cause fragmentation because of noise and small irrelevant intensity changes
- Improved by smoothing the gradient image or using markers

Solution: Watershed with markers



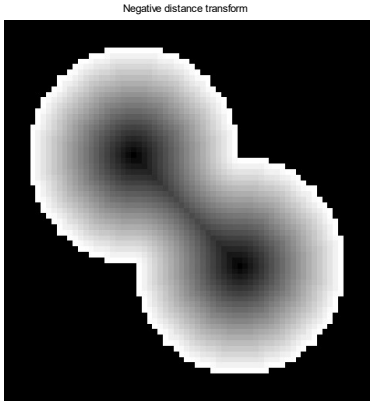
- A marker is an extended connected component in the image
- Can be found by intensity, size, shape, texture etc
- Internal markers are associated with the object
  (a region surrounded by bright point (of higher altitude))
- External markers are associated with the background
  (watershed lines)
- Segment each sub-region by some segmentation algorithm

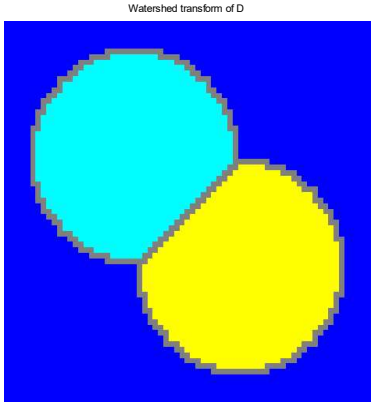
How to find markers

- Apply filtering to get a smoothed image
- Segment the smooth image to find the internal markers.
  - Look for a set of point surrounded by bright pixels.
  - How this segmentation should be done is not well defined.
  - Many methods can be used.
- Segment smooth image using watershed to find external markers, with the restriction that the internal markers are the only allowed regional minima.
  The resulting watershed lines are then used as external markers.
- We now know that each region inside an external marker consists of a single object and its background.
- Apply a segmentation algorithm (watershed, region growing, threshold etc.) only inside each watershed.

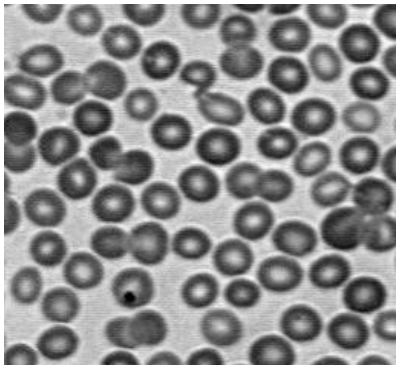
Splitting objects by watershed

- Example: Splitting touching or overlapping objects.
  - Given graylevel (or color) image
  - Perform first stage segmentation
    - (edge detection, thresholding, classification,…)
  - Now you have a labeled image, e.g. foreground / background
  - Obtain distance transform image
    - From each foreground pixel, compute distance to background.
  - Use watershed algorithm on *inverse* of distance image.

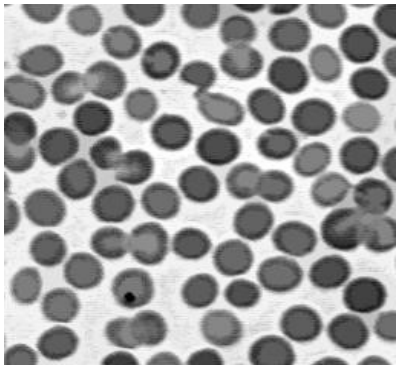




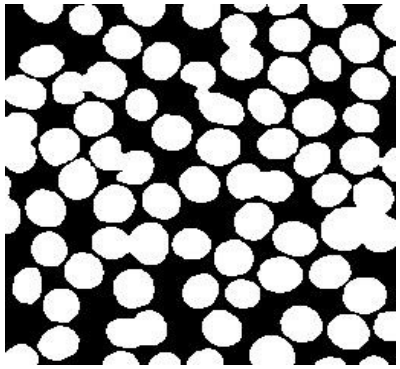
Watershed – advanced example



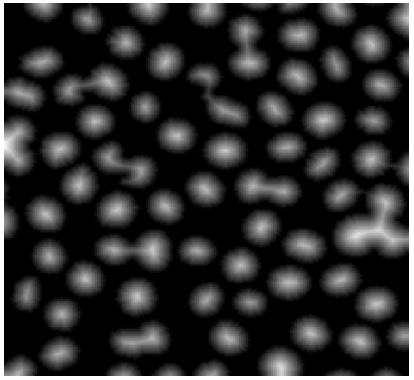
Original



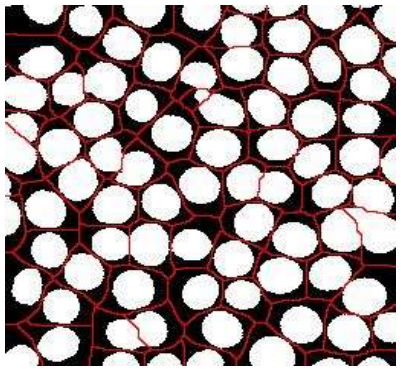
Opening



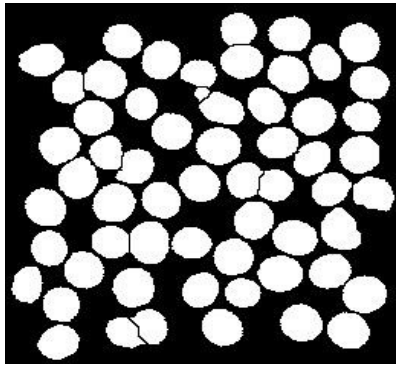
Threshold, global



Distance transform
- Distance from a point in a region to the border of the region



Watershed of inverse of distance transform

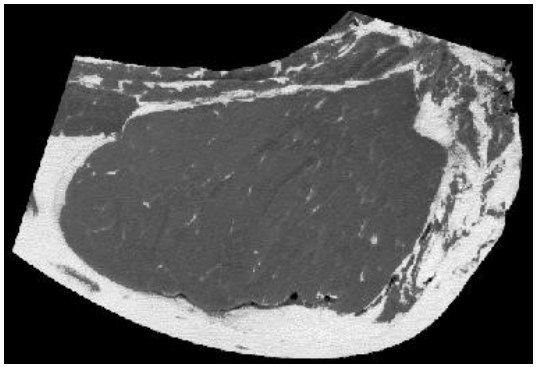


Second watershed to split cells

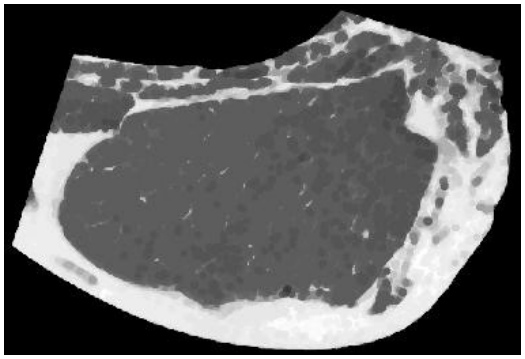
Matlab:

Imopen
bwdist
imimposemin
imregionalmax
watershed

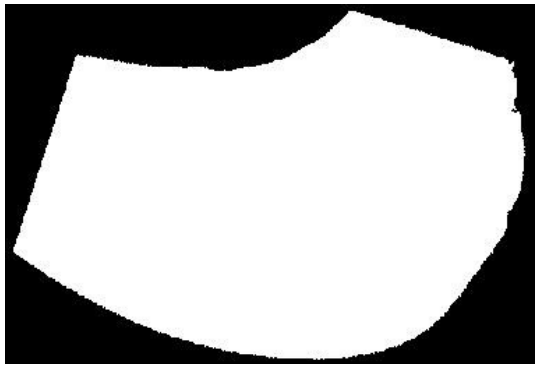
Watershed – example 3



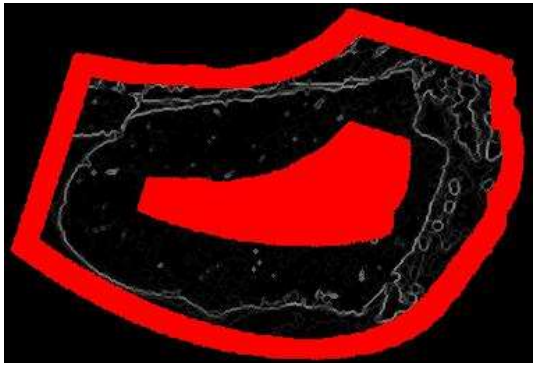
Original



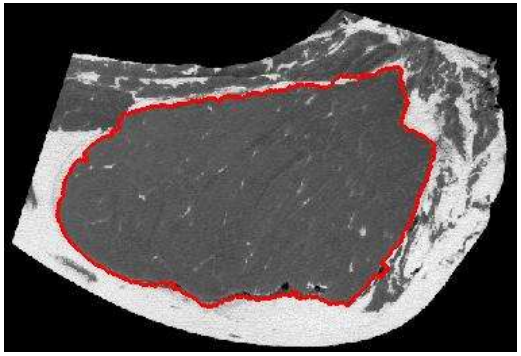
Opening



Threshold



Find internal and external markers from gradient image



Watershed

Matlab:
- imopen
- imimposemin
- bwmorph
- watershed

Σ: Watershed

- **Advantages :**
  - Gives connected components
  - A priori information can be implemented in the method using markers

- **Disadvantages :**
  - Often needs preprocessing to work well
  - Fragmentation or "over-segmentation" can be a problem

Segmentation by motion

- In video surveillance one of the main tasks is to segment foreground objects from the background scene to detect moving objects.
- The background may have a lot of natural movements and changes
  - (moving trees, snow, sun, shadows etc)
- Motion detection may be restricted to given ROI
  - to avoid moving trees etc.

Difference images

- Thresholding intensity differences between frames:

$$d_{ij}(x,y) = \begin{cases} 1 & if \left| f(x,y,t_i) - f(x,y,t_j) \right| > T \\ 0 & otherwise \end{cases}$$

  - Images must be of same size, co-registered, same lighting, …
  - Clean up small objects in d_{ij}, caused by noise.
  - Possible to accumulate counts of differences per pixel (ADI's).
- Background subtraction:
  - Estimate the background image (per pixel) by computing the mean and variance of the n previous time frames at pixel (x,y).
  - Subtract background image from current frame
  - Note that the background is updated by only considering the *n* previous frames. This eliminates slow trends.

Learning goals - segmentation

- Fundamentals of segmentation (10.1) is important.
- Basic edge and line detection (10.2) must be known.
- Thresholding (10.3) is assumed known.
- Region growing and split-and-merge is not trivial.
- Morphological watersheds may be very effective.
- Combining region- and edge-based methods is a good idea.
- Do the exercises (“learning by programming”) !