GNR602
Advanced Methods in Satellite Image Processing

Instructor: Prof. B. Krishna Mohan
CSRE, IIT Bombay
bkmohan@csre.iitb.ac.in

Slot 13

Lecture 23-25 Image Segmentation Techniques

Contents of the Lecture

- Definition of Image Segmentation
- Image Segmentation by Thresholding
- Optimal Thresholding
- Edge Detection
- Region Growing

Definition

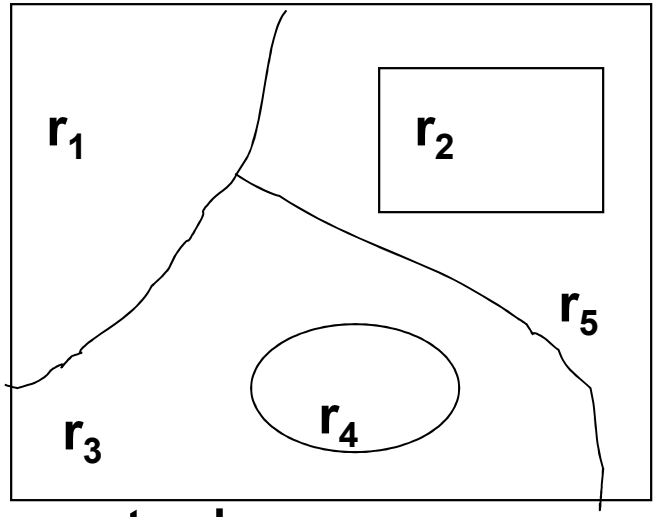
- **Given by Theodosis Pavlidis**
- An image S segmented to R regions r_i should satisfy

$$\bigcup_{i=1}^{R} r_i = S$$

$$P(r_i) = \text{True}$$

where P is a homogeneity predicate.

$$P(r_i \cap r_j) = \text{False}, i \neq j$$



Each r_i is 4-connected or 8-connected

Definition

- **Given by Theodosis Pavlidis**
- Widely accepted, since it covers the logical process of image segmentation
- If the image is decomposed into its constituent parts, each part should correspond to some object in real world, and one or more parts build up the entire object

Interpretation of Pavlidis' Definition

- $P(r_i)$ = True
- The image is made up of regions where each region is homogeneous according to some predicate or condition
- The region (or segment) should have low internal variance so that it does not appear to be made up of smaller homogeneous segments

Interpretation of Pavlidis' Definition

The second condition requires that the segments cover the entire image

$$\bigcup_{i=1}^{R} r_i = S$$

The entire image is segmented into homogeneous regions

Interpretation of Pavlidis' Definition

- 3rd condition suggests that the adjacent regions cannot be merged without flouting the homogeneity criterion

$$P(r_i \cap r_j) = \text{False}, i \neq j$$

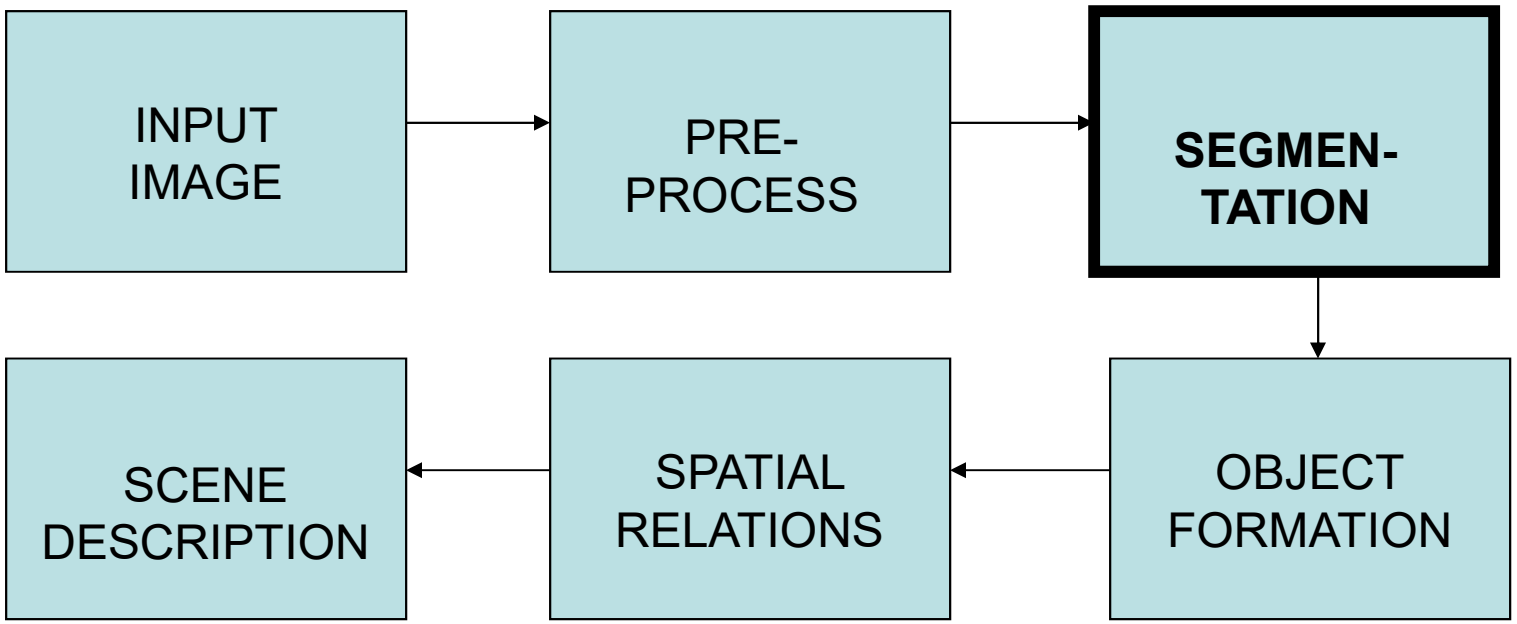
Obviously this condition does not make sense if the regions are not spatially adjacent, since the regions representing the same type of objects can be physically separated.

Interpretation of Pavlidis' Definition

- 4^{th} condition
  - Each r_i is 4-connected or 8-connected

This condition implies that each region is made up of spatially contiguous set of pixels. We do not refer to the same region or segment if there is no connected path within it joining any point to any other point

Role Of Image Segmentation In Image Understanding



Cont…

- The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.

- They differ greatly in the types of image to which they are likely to be successful and in computational costs.

- Most explored area in IP.

Image Segmentation Approaches

- **Image thresholding**
- Texture analysis
- Edge/line detection
- Region growing
- Clustering
- Pattern recognition

Image segmentation by Thresholding

- Thresholding is a popular way to segment an image of a binary scene.
- Thresholding - based on information contained in a GL histogram of a given image.
- The objective of thresholding approach is to determine the distributions representing the desired object and the background, and the threshold point in such a case will be the valley (lowest frequency point) of the histogram that, hopefully, best separates the two groups.

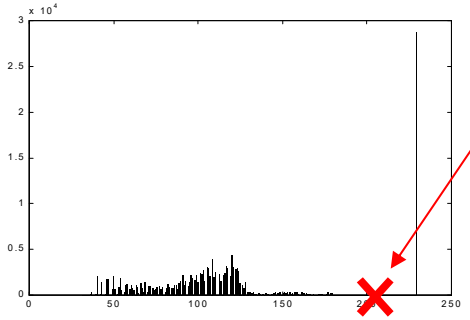
Image thresholding

- The conventional thresholding schemes :
  - Global Thresholding
  - Local Thresholding
  - Dynamic Thresholding

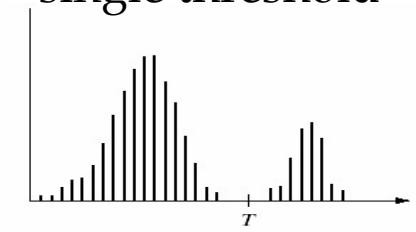
Thresholding Method

thresholding

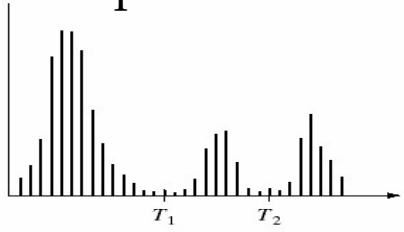
histogram



single threshold



multiple thresholds



From [Gonzalez & Woods]

Cont…

$$g(i,j) = T(f(i,j)) = B \quad \text{if } f(i,j) < P$$
$$= W \text{ otherwise}$$

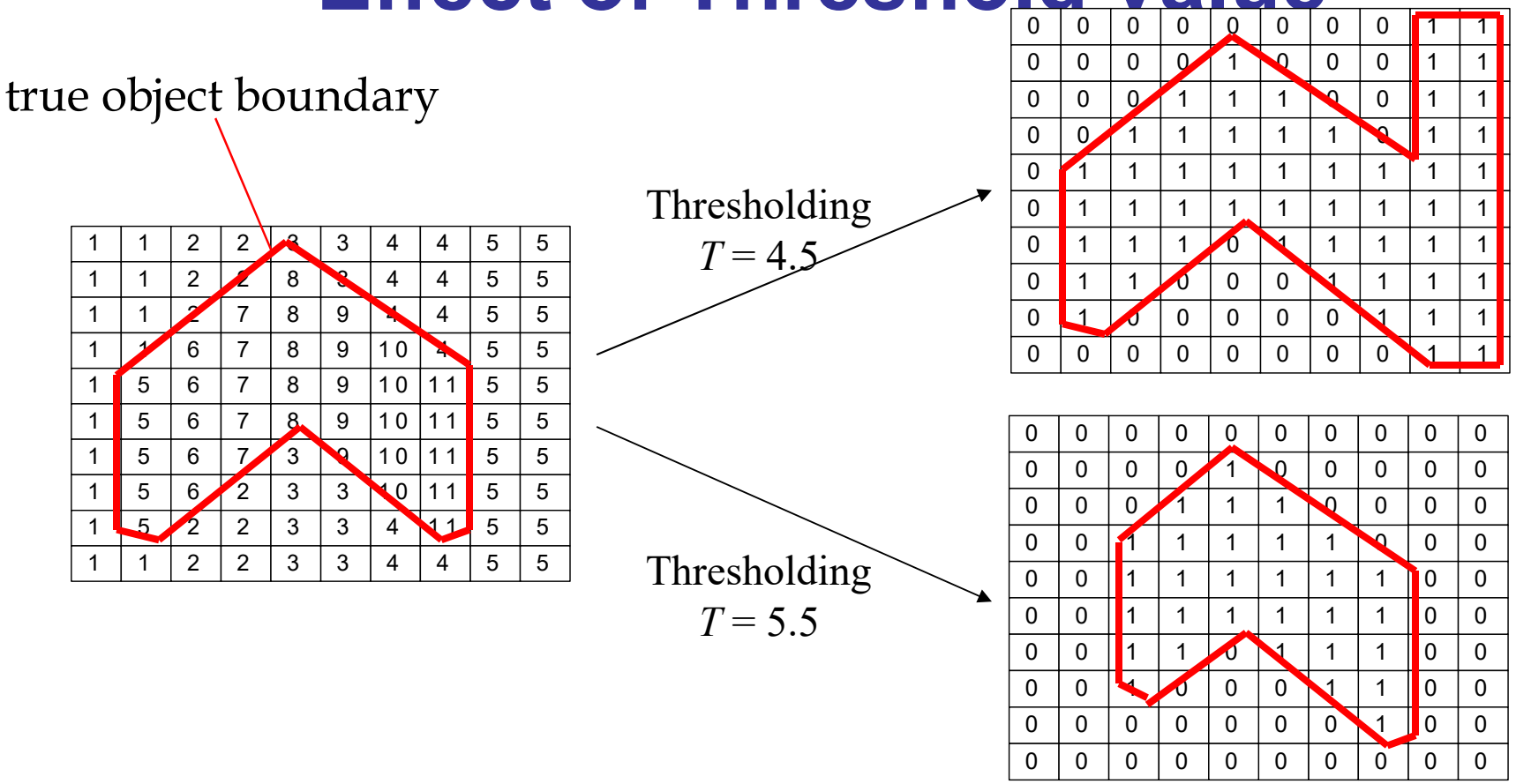
If several thresholds T1, T2 , ... ,Tk are chosen, then the output image contains k+1 levels.

$$g(i,j) = T(f(i,j)) = g1 \quad \text{if } 0 \le f(.,.) < T1$$
$$= g2 \quad \text{if } T1 \le f(.,.) < T2$$
$$= .....$$
$$= gk+1 \quad \text{if } Tk \le f(.,.)$$

Global Thresholding

- Identification of a level P such that all pixels in the image with gray level below P to be represented by a value B and all pixels with gray level equal to or above P to be represented by a value W.
- $g(i,j) = T(f(i,j)) = B \quad \text{if } f(i,j) < P$
  $= W \text{ otherwise}$
- T(.) is the thresholding function.
- This operation applies ***globally*** to all pixels in the image irrespective of their ***position*** or the gray levels of their neighbors. This is performed by a simple table-lookup operation in real time

Effect of Threshold value



Local Thresholding

- Local thresholding varies the threshold based on the region in which the pixel is located

B11	B12	B13
B21	B22	B23
B31	B32	B33

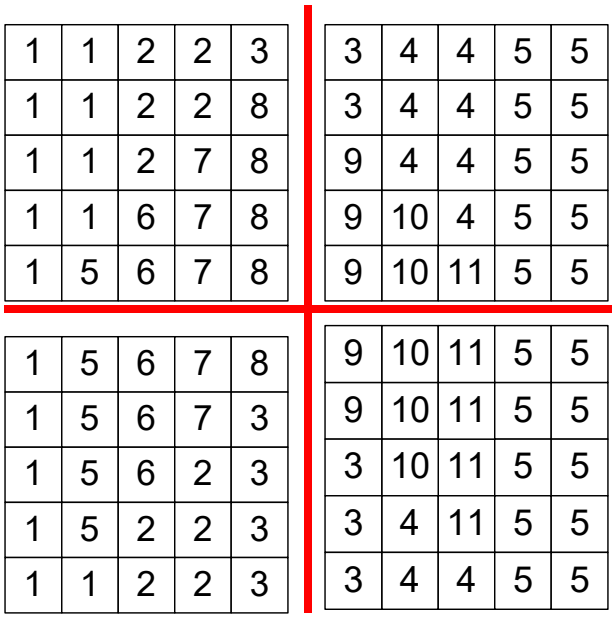
Each block b(i,j) has a separate threshold P(i,j)
This approach can better adapt to local intensity variations, but creates discontinuities at the boundaries of adjacent blocks

Local Thresholding Method

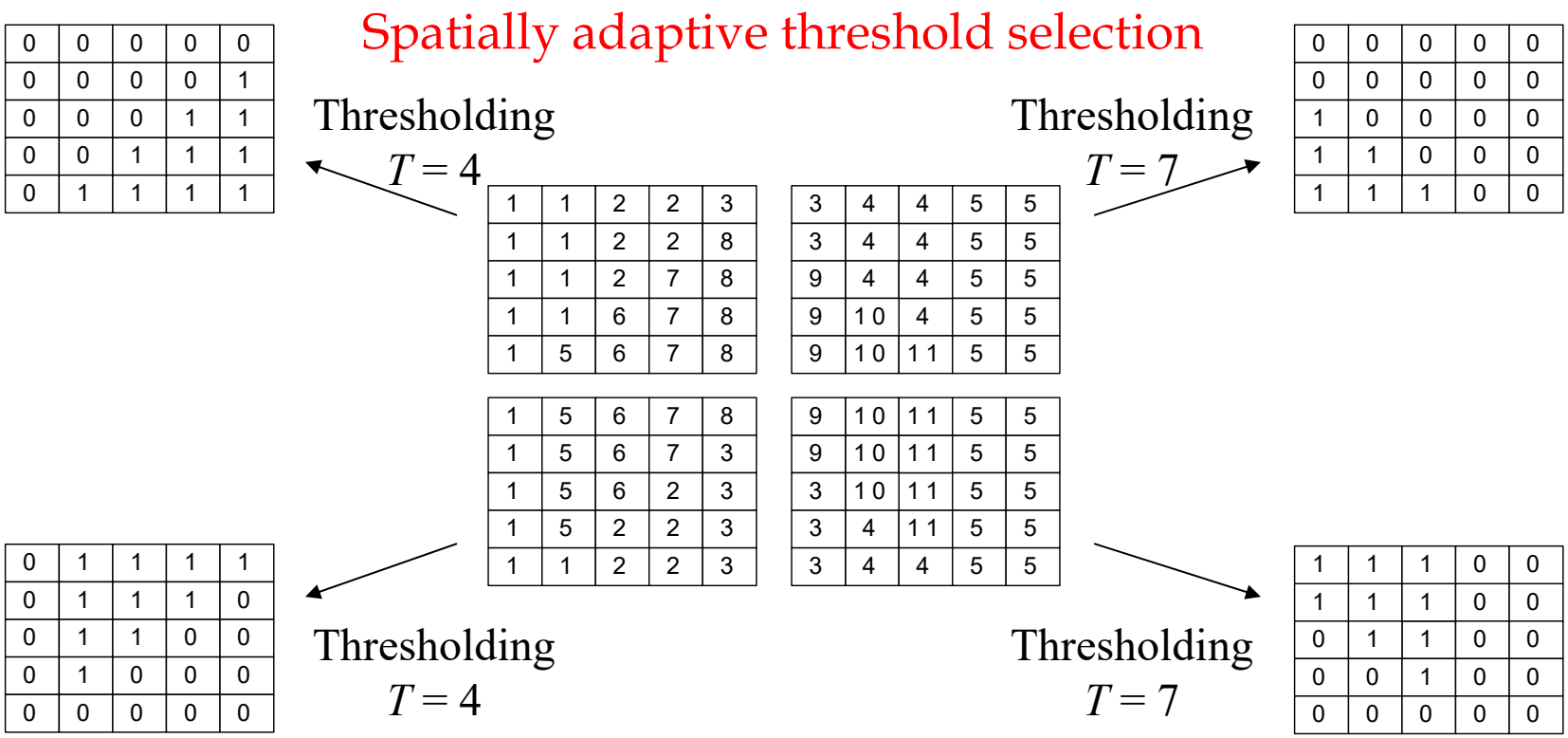
- Spatially adaptive thresholding
- Localized processing

1	1	2	2	3	3	4	4	5	5
1	1	2	2	8	3	4	4	5	5
1	1	2	7	8	9	4	4	5	5
1	1	6	7	8	9	10	4	5	5
1	5	6	7	8	9	10	11	5	5
1	5	6	7	8	9	10	11	5	5
1	5	6	7	3	9	10	11	5	5
1	5	6	2	3	3	10	11	5	5
1	5	2	2	3	3	4	11	5	5
1	1	2	2	3	3	4	4	5	5

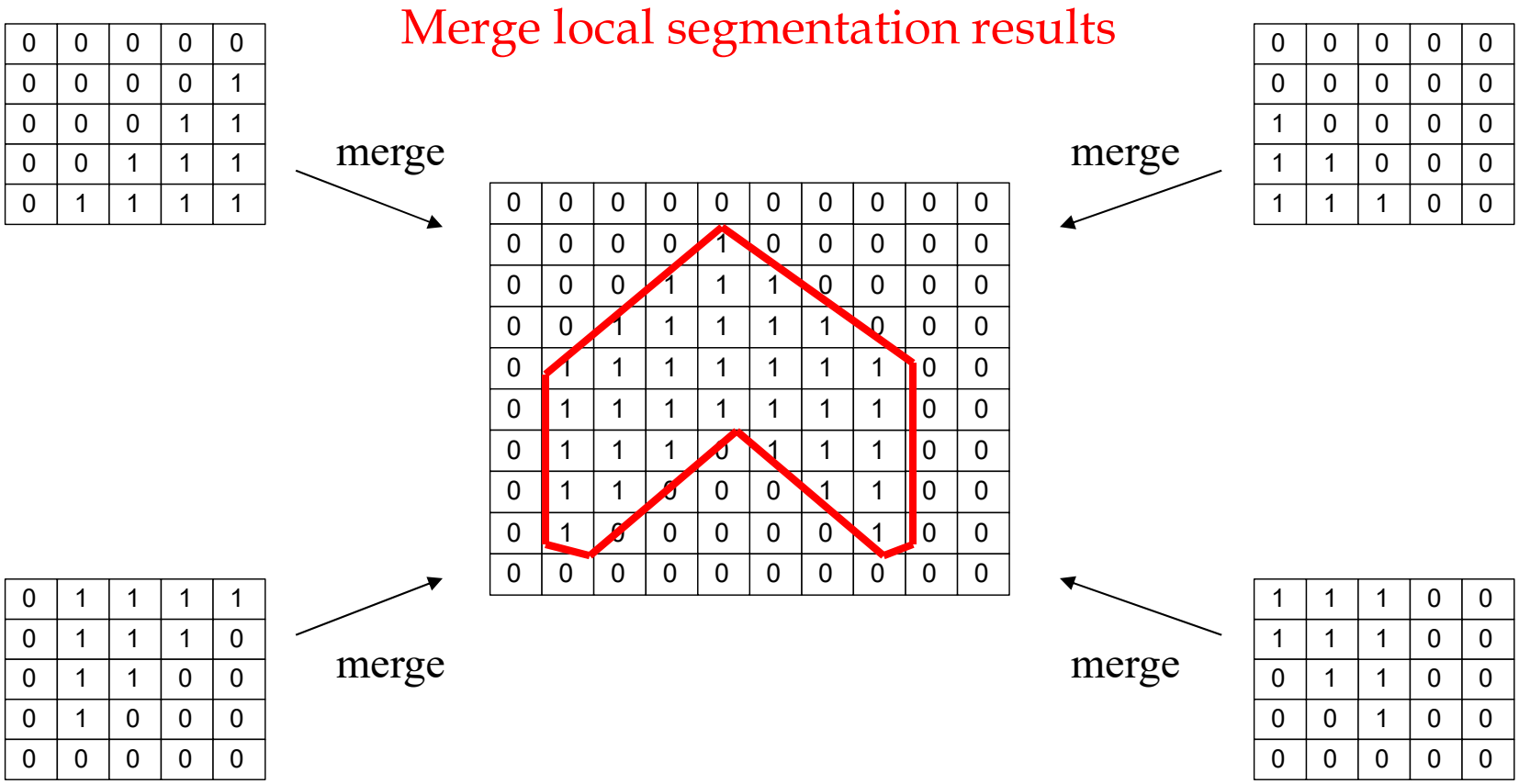
Split



Local Thresholding Method



Local Thresholding Method



Dynamic Thresholding

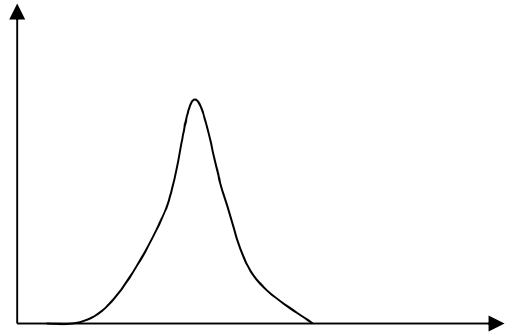
- The threshold is allowed to vary from pixel to pixel in this case.
- The procedure is similar to local thresholding. The additional step here is the generation of a *threshold surface* by interpolating the threshold values computed for different blocks.
- The *threshold surface* provides a threshold for each pixel of the image.
- This overcomes the limitation of discontinuities at the block boundaries

Classical Automatic Thresholding Algorithm

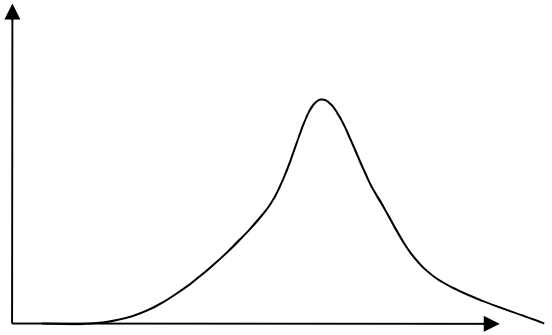
- Select an initial estimate for T
- Segment the image using T, producing 2 groups: G1, pixels with value >T and G2 , with value <T
- Compute μ1 and μ2, average pixel value of G1 and G2
- New threshold: T=1/2(μ1+μ2)
- Repeat steps 2 to 4 until T stabilizes.

- Very easy + very fast
- Assumptions: normal dist. + low noise
- This is the well known ------- algorithm!

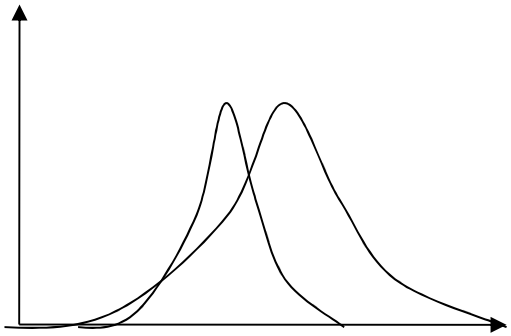
- **Optimal Thresholding** is based on the shape of the current image histogram. Search for valleys, Gaussian distributions etc.



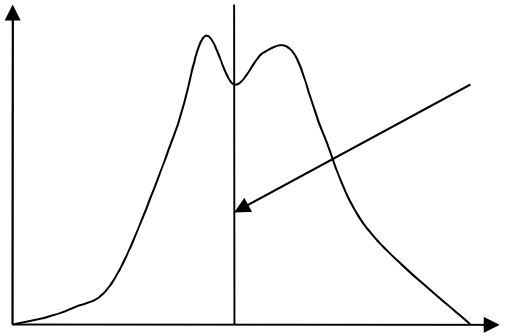
Foreground



Background

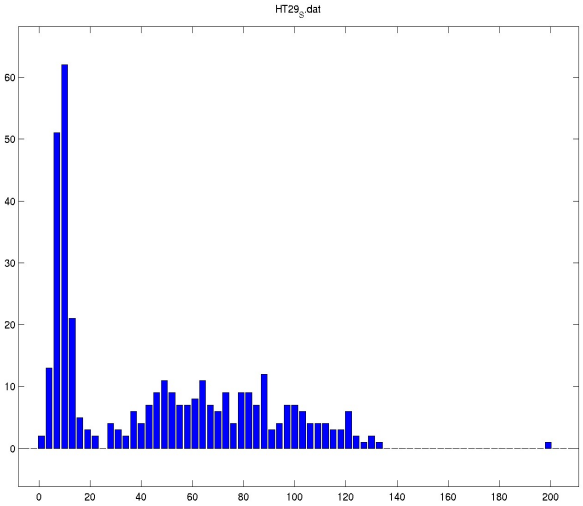


Both

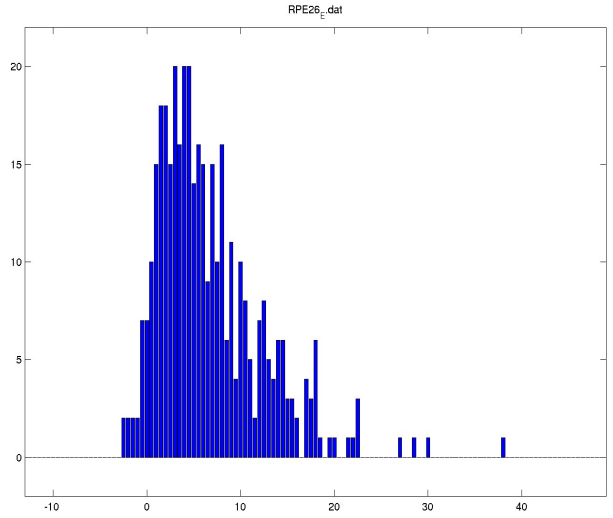


Real histogram

Histograms

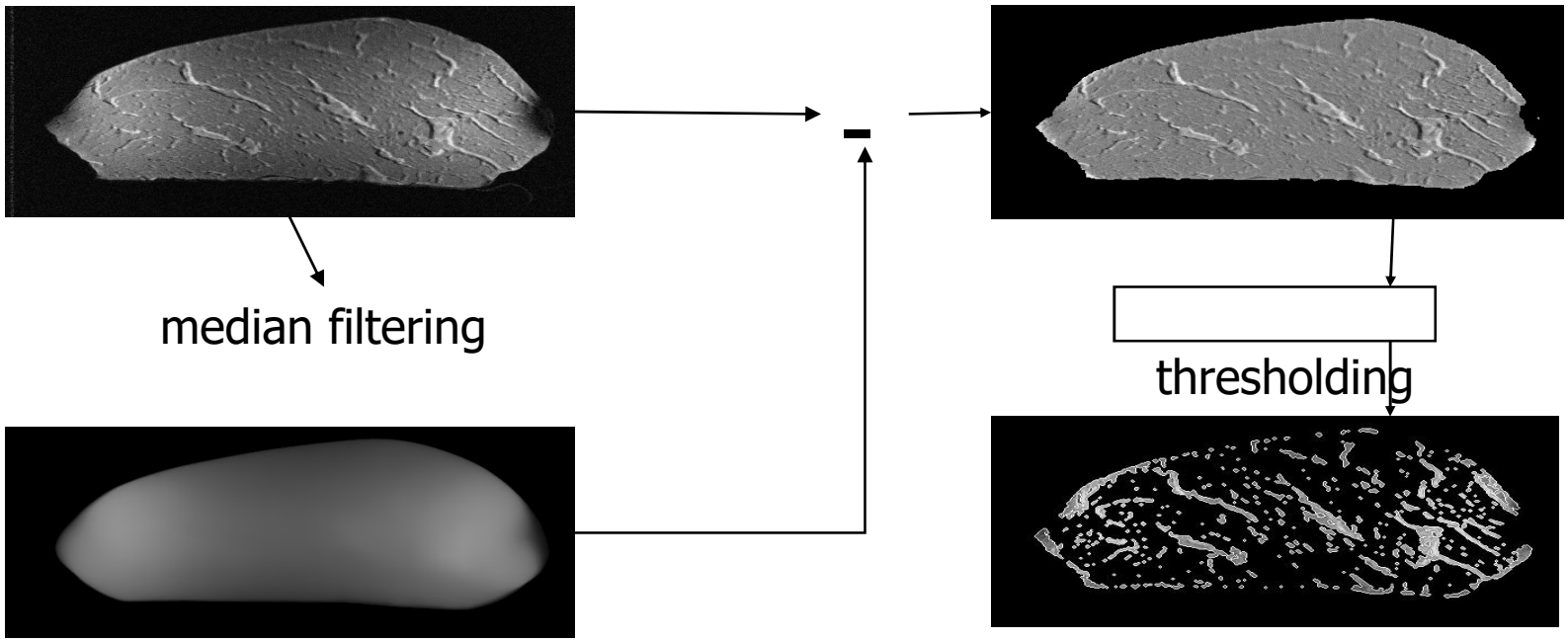


Good for thresholding



Troublesome

Effect of Smoothing



Optimal Thresholding

- Histogram shape can be useful in locating the threshold. However it is not reliable for threshold selection when peaks are not clearly resolved
- Optimal thresholding: a criterion function is devised that yields some measure of separation between regions
- A criterion function is calculated for each intensity and that which maximizes/minimizes this function is chosen as the threshold

Otsu's Method

- Otsu's thresholding method is based on selecting the lowest point *between* two *classes* (histo.peaks).
- Based on the threshold, the two classes have respective means and standard deviations

  – m1, m2, s1, s2 respectively, L total number of levels.

  – n_i: number of pixels in level i; N total number of pixels;

  ❖ ω_0, ω_1 fraction of pixels below and above threshold

$$\omega_0 = \frac{1}{N}\sum_{i=0}^{T} n_i \qquad \omega_1 = \frac{1}{N}\sum_{i=T+1}^{L-1} n_i$$

Otsu's Method

- Analysis of variance (variance=standard deviation^2)
  - Mean of pixels up to threshold T, above threshold T and overall mean

$$\mu_0 = \frac{1}{N_0}\sum_{i=0}^{T} n_i.i \qquad \mu_1 = \frac{1}{N_1}\sum_{i=T+1}^{L-1} n_i.i \qquad \mu = \frac{1}{N}\sum_{i=0}^{L-1} n_i.i$$

$$\sigma^2 = \frac{1}{N}\sum_{i=0}^{L-1} (i-\mu)^2 n_i$$

Between Class and Within Class Variance

- The total variance of the image is the sum of the between class variance and within class variance
- Both within class variance σ_ω^2 and between class variance σ_b^2 are dependent on the threshold selected
- We should minimize σ_ω^2 or maximize σ_b^2 since their sum is a constant, independent of threshold

Otsu's Method

Between-classes variance (δ_b^2): The variation of the mean values for each class from the overall intensity mean of all pixels:

$$\delta_b^2 = \omega_0 (\mu_0 - \mu)^2 + \omega_1(\mu_1 - \mu)^2,$$

Substituting $\mu = \omega_0 \mu_0 + \omega_1\mu_1$, we get:

$$\delta_b^2 = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

ω_0, ω_1, μ_0, μ_1 stands for the relative frequencies and mean values of two classes, respectively.

Otsu's Method

- The criterion function involves *between-classes* variance to the total variance is defined as:

$$\eta(T) = \delta_b^2 / \delta^2$$

- Since δ_b^2 is a function of threshold T, $\eta(T)$ is evaluated for all possible thresholds, and the one that maximizes η is chosen as the optimal threshold

Otsu's Method

- The within cluster variance need not be separately minimized, since maximizing between cluster variance automatically minimizes within cluster variance. This is because the sum of between cluster variance and within cluster variance equals the total variance of the image, which is independent of the threshold chosen

Finding the Threshold

- A sequential search can be carried out by choosing the threshold = 1 to 254, such that the value at which η is maximum is the optimal threshold.

Input Image

2 thresholds

3 display levels

0, 127, 254

6 thresholds

7 display levels

0, 42, 84, 126, 168, 210, 252

Entropy Method

- Entropy is served as a measure of information content
- A threshold level ***t*** separates the whole information into two classes, and the entropy associated with them is:

$$H_b = -\sum_{i=0}^{t} p_i \log(p_i) \qquad H_w = -\sum_{i=t+1}^{255} p_i \log(p_i)$$

- Optimal threshold is the one maximizes:

$$\boldsymbol{H = H_b + H_w}$$

Minimum Error Thresholding

- Proposed by Josef Kittler and John Illingworth, University of Surrey, UK
  - Let us consider an image whose pixels assume grey level values *g* in the interval [0,*n*]
  - Histogram given by *h*(*g*), normalized histogram by p(g) = h(g)/n
  - The histogram is an estimate of *p*(*g*), the PDF of a mixture population

Minimum Error Thresholding

- Let us consider that the two populations can be modeled as Gaussian distributions

$$p(g) = \sum_i P_i p(g \mid i), \text{ where}$$

$$p(g \mid i) = \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left[-\left(\frac{x - \mu_i}{\sqrt{2}\sigma_i} \right)^2 \right]$$

To avoid confusion, we write p(g|i) as p(g|i,T) since classes 1 and 2 are formed according to the threshold T chosen in our context.

Minimum Error Thresholding

- Suppose the grey level data is thresholded at some level T and each of the two resulting pixel populations are approximated by a normal density $h(g|i,T)$ with parameters $\mu_i(T)$, $\sigma_i(T)$, and *a priori* probability $P_i(T)$. For a single threshold, we can compute h(g|1,T) and h(g|2,T) with parameters $\mu_1(T)$, $\mu_2(T)$, $\sigma_1(T)$, $\sigma_2(T)$ respectively.

Minimum Error Thresholding

$$P_1(T) = \sum_{i=0}^{T} h(i)$$

$$P_2(T) = \sum_{i=T+1}^{n} h(i)$$

$$\mu_1(T) = \sum_{i=0}^{T} ih(i) \Big/ P_1(T)$$

$$\mu_2(T) = \sum_{i=T+1}^{n} ih(i) \Big/ P_2(T)$$

Minimum Error Thresholding

- The probability of a pixel being mapped correctly (below threshold or above threshold) is denoted by P(g,T) and given by

$$P(g,T) = \frac{p(g \mid i,T)P_i(T)}{p(g)}$$

where i=1 if g $\leq$ T, and i=2 if g > T

Substituting the expressions for the individual class conditional distributions, and taking logarithms, and multiplying by -2, we can rewrite P(g,T)

Minimum Error Thresholding

$$P(g,T) = \left[\frac{g - \mu_i(T)}{\sigma_i(T)} \right]^2 + 2.\log(\sigma_i(T)) - 2\log(\mu_i(T))$$

where i=1 if g $\leq$ T, and i=2 if g > T (Verify!)

The average performance for the whole image is given by

$$J(T) = \sum_g p(g) P(g,T)$$

For a given threshold T, this function indicates the extent of overlap between the two classes according to the threshold selected

Minimum Error Thresholding

- Expanding the previous equation,

$$J(T) = \sum_g \left[\frac{g - \mu_1(T)}{\sigma_1(T)} \right]^2 + 2.\log(\sigma_1(T)) - 2\log(\mu_1(T)) +$$

$$\sum_g \left[\frac{g - \mu_2(T)}{\sigma_2(T)} \right]^2 + 2.\log(\sigma_2(T)) - 2\log(\mu_2(T))$$

Substituting the previous equations in the above, we can write

$$J(T) = 1 + 2[\mu_1(T)\sigma_1(T) + \mu_2(T)\sigma_2(T)] -$$

$$2[\mu_1(T)\log(\mu_1(T)) + \mu_2(T)\log(\mu_2(T))$$

Minimum Error Thresholding

- J(T) is evaluated for each possible threshold and the threshold that minimizes J(T) is the optimum choice
- Difference between Otsu's method and this method is that here the histogram is assumed to be a mixture of two Gaussian distributions, one for object and other for background.
- In Otsu's method this assumption is not made, the inter-class separation is expressed in terms of difference of means of object and background

Segmentation by Edge Detection

What is an Edge ?

- An edge is a discontinuity in the perceptual property – brightness / color / texture / surface orientation
- An edge is a set of connected pixels that lie on the boundary between two regions
- The pixels on an edge are called edge pixels or edgels
- Gray level / color / texture discontinuity across an edge causes edge perception
- Position & orientation of edge are key properties

Different Edges

Different colors

Different brightness

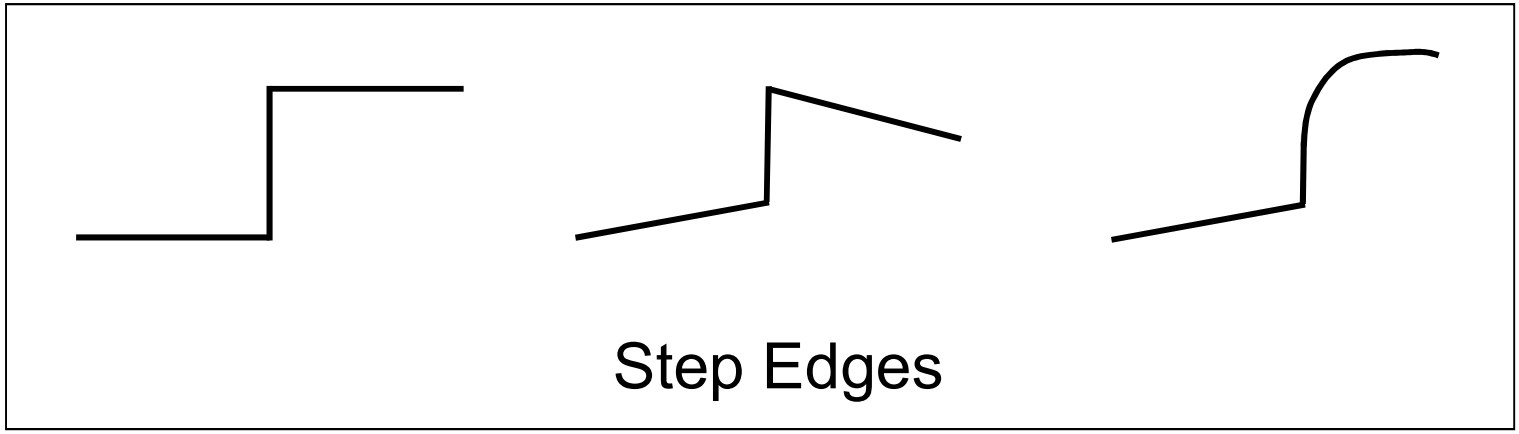
Different textures

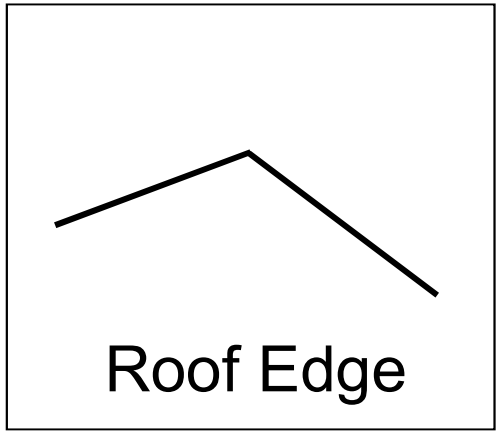
Different surfaces

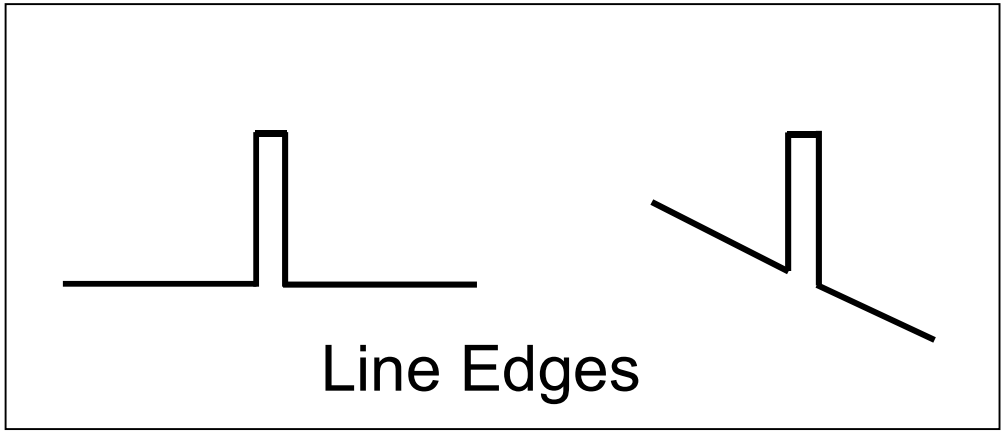
Cont…

- The boundaries of an object are often considered to be analogous to its edges.
- These boundaries are discovered by following a path of rapid change in image intensity.
- Most edge-detection functions look for places in the image where the intensity changes rapidly by locating places where the first derivative of the intensity is larger in magnitude than some threshold, or finding places where the second derivative of the intensity has a zero crossing

Types of Edge



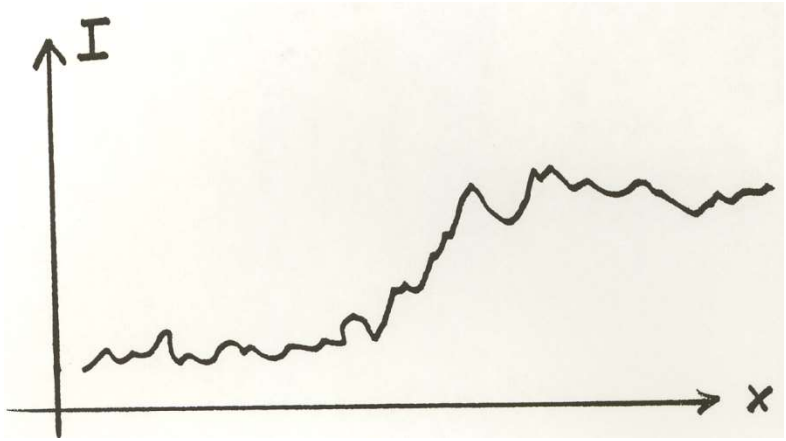




Different kinds of edges

Real Edges

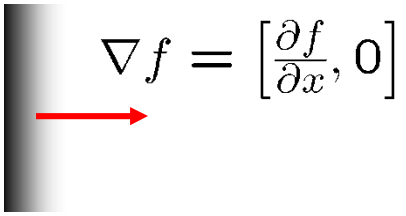
Noisy and Discrete!

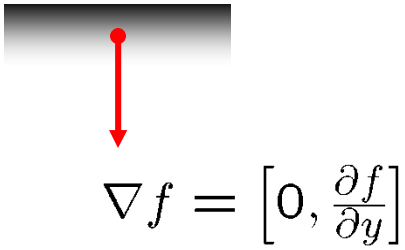


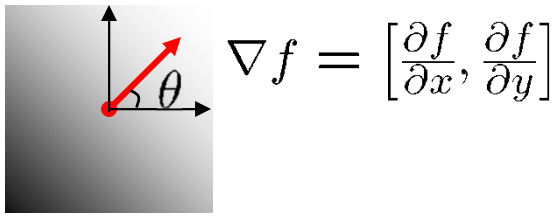
We want an **Edge Operator** that produces:

- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

- The gradient points in the direction of most rapid change in intensity







- The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

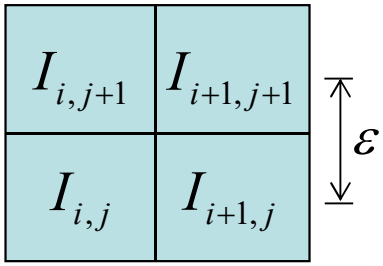
Discrete Edge Operators

How can we differentiate a ***discrete*** image?

Finite difference approximations:

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i,j+1}\right) + \left(I_{i+1,j} - I_{i,j}\right)\right)$$

$$\frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\left(\left(I_{i+1,j+1} - I_{i+1,j}\right) + \left(I_{i,j+1} - I_{i,j}\right)\right)$$



Convolution masks :

$$\frac{\partial I}{\partial x} \approx \frac{1}{2\varepsilon}\begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad \frac{\partial I}{\partial y} \approx \frac{1}{2\varepsilon}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

- Second order partial derivatives:

$$\frac{\partial^2 I}{\partial x^2} \approx \frac{1}{\varepsilon^2}\left(I_{i-1,j} - 2I_{i,j} + I_{i+1,j}\right)$$

$$\frac{\partial^2 I}{\partial y^2} \approx \frac{1}{\varepsilon^2}\left(I_{i,j-1} - 2I_{i,j} + I_{i,j+1}\right)$$

$I_{i-1,j+1}$	$I_{i,j+1}$	$I_{i+1,j+1}$
$I_{i-1,j}$	$I_{i,j}$	$I_{i+1,j}$
$I_{i-1,j-1}$	$I_{i,j-1}$	$I_{i+1,j-1}$

- **Laplacian** :

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

Convolution masks :

$$\nabla^2 I \approx$$

0	1	0
1	−4	1
0	1	0

or $\frac{1}{20}$

1	4	1
4	−20	4
1	4	1

(more accurate)

Sobel operators

Better approximations of the gradients exist

The *Sobel* operators below are commonly used

$\frac{1}{8}$

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

s_x

$\frac{1}{8}$

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

s_y

Comparing Edge Operators

Gradient: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

Roberts (2 x 2):

0	1
-1	0

1	0
0	-1

Sobel (3 x 3):

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

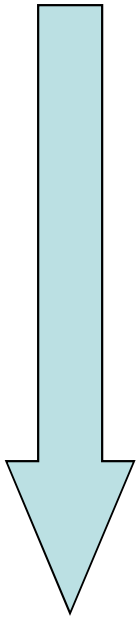
1	1	1
0	0	0
-1	-1	1

Sobel (5 x 5):

-1	-2	0	2	1
-2	-3	0	3	2
-3	-5	0	5	3
-2	-3	0	3	2
-1	-2	0	2	1

1	2	3	2	1
2	3	5	3	2
0	0	0	0	0
-2	-3	-5	-3	-2
-1	-2	-3	-2	-1

Good Localization
Noise Sensitive
Poor Detection

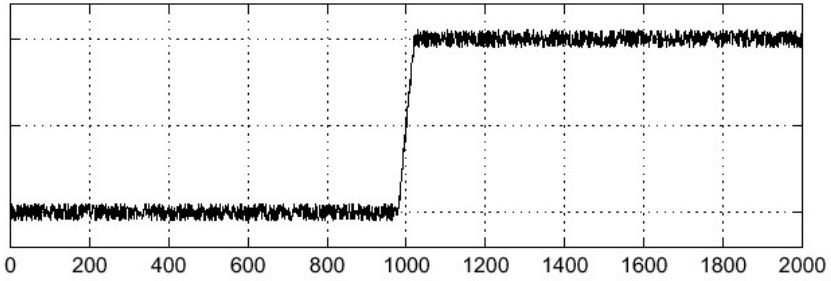


Poor Localization
Less Noise Sensitive
Good Detection

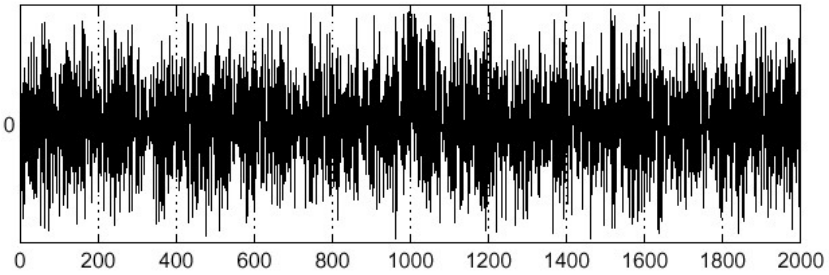
Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

Solution: Smooth first

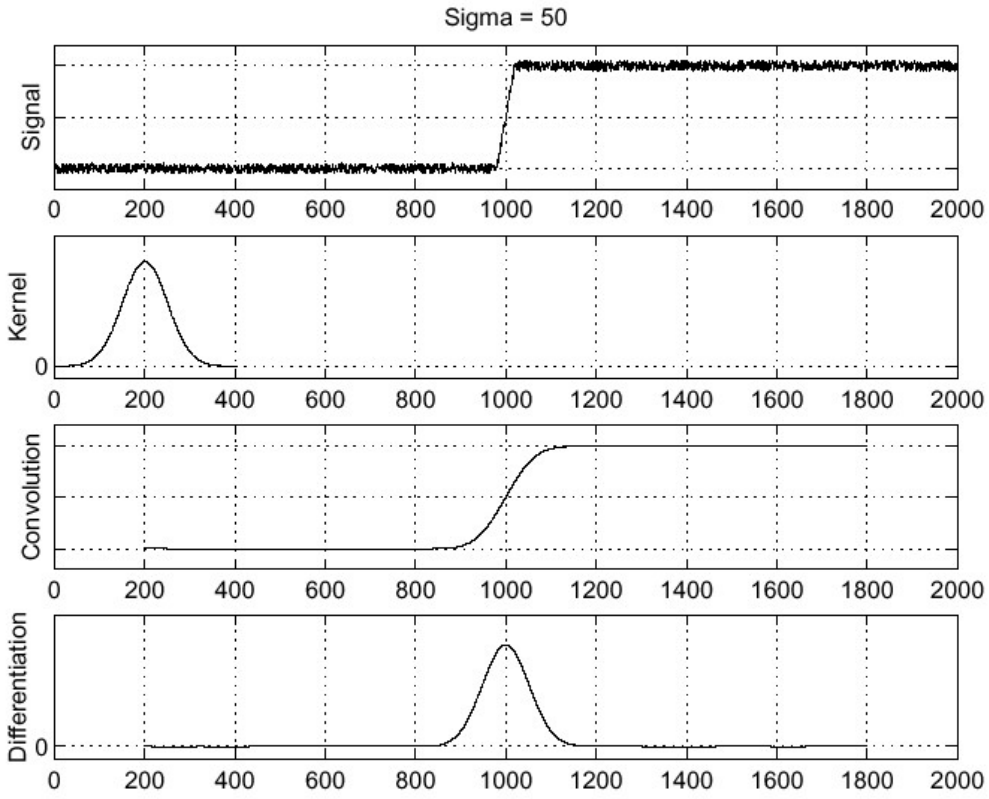
Where is the edge?

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$

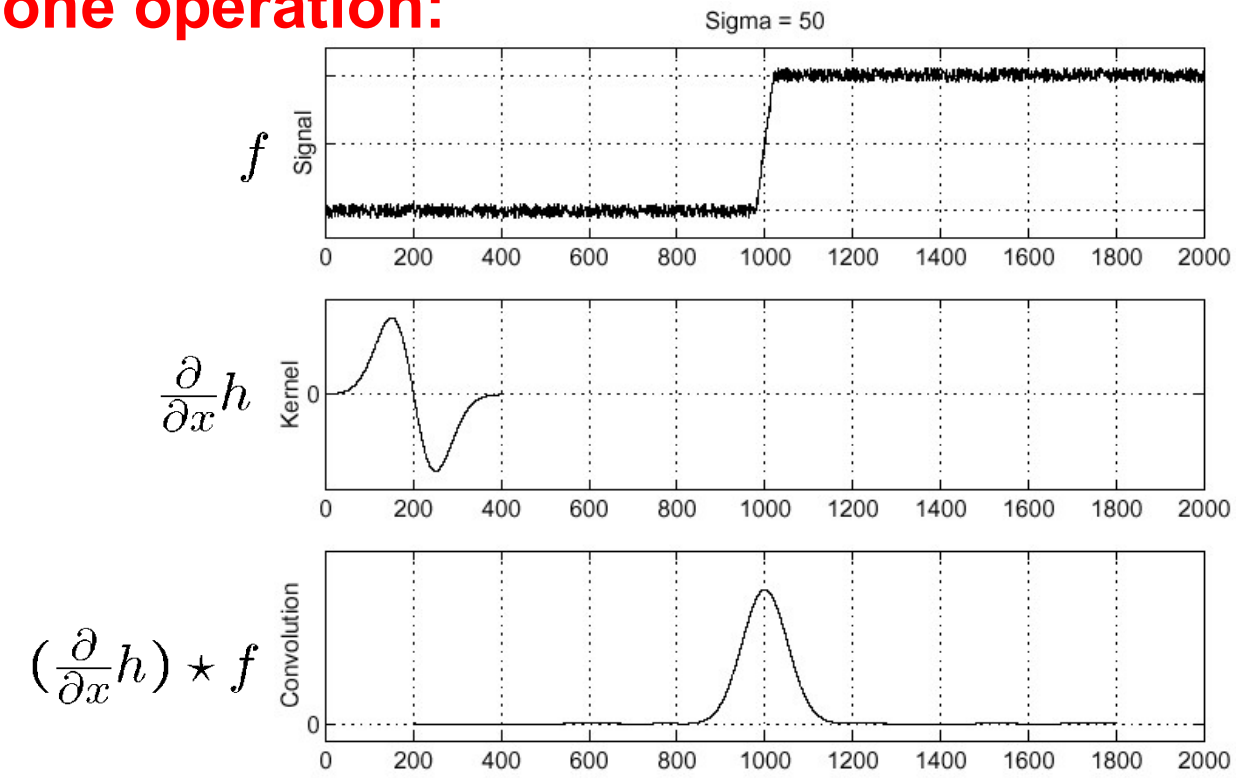


Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

This saves us one operation:



The Laplacian Operator

Laplacian operator is a second derivative operator that is a discrete approximation of the Laplace heat equation.

The Laplacian operator combines the second order derivatives as follows:

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

Cont…

Common Laplacian kernels are:

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

−1	2	−1
2	−4	2
−1	2	−1

Higher derivative operators amplify noise. Thus, Laplace operator output is much more noisy compared to the first derivative operators such as Sobel, Prewitt etc.

Laplacian of Gaussian

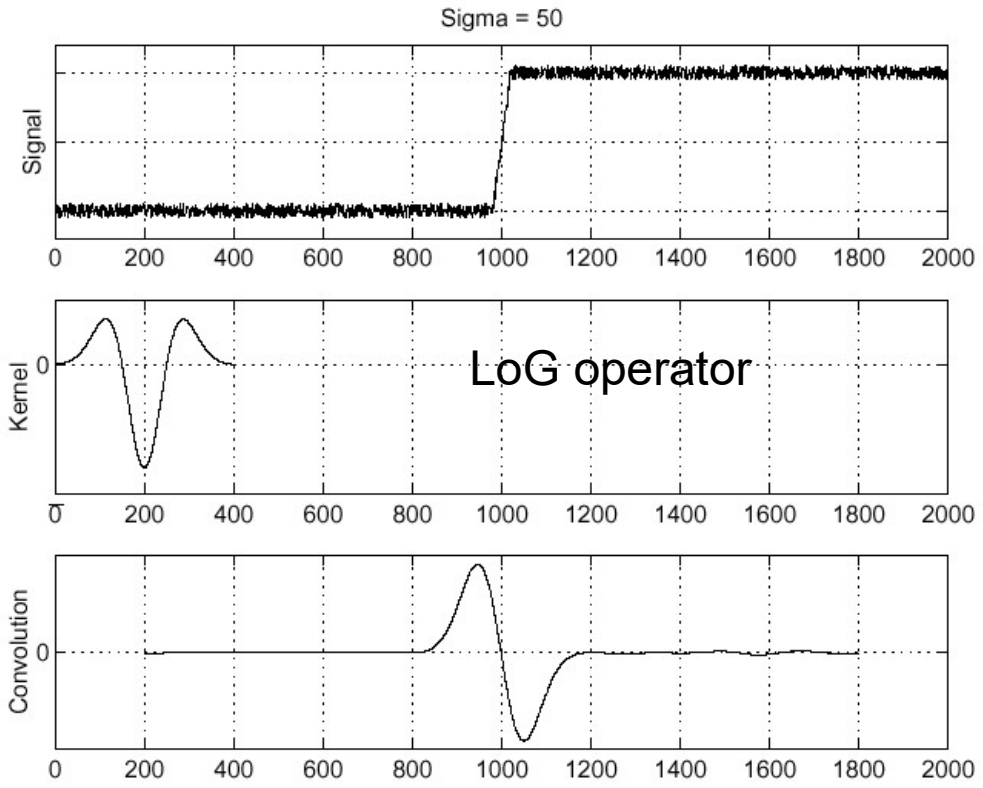
- The noise in the input image is reduced by smoothing.
- Among the various smoothing operators, Gaussian filter has desirable properties in terms of space-frequency localization.
- Input image is therefore smoothed using the Gaussian shaped smoothing operator, whose width s is user controllable.
- In this approach, an image should first be convolved with Gaussian filter

$$g(x,y) = \nabla^2 \left[G(x,y,\sigma) * f(x,y) \right]$$

Laplacian of Gaussian

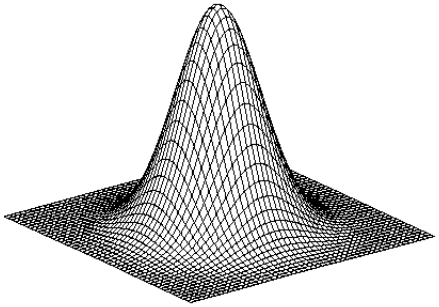
Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

Where is the edge?



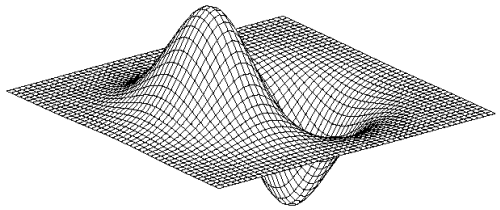
Zero-crossings

2D edge detection filters



Gaussian

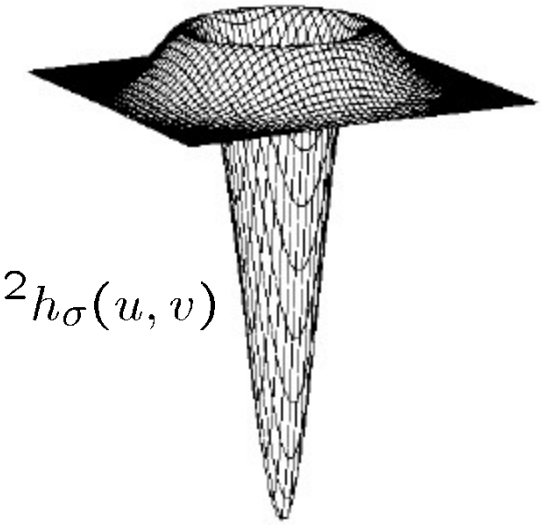
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u,v)$$

Laplacian of Gaussian



$$\nabla^2 h_\sigma(u,v)$$

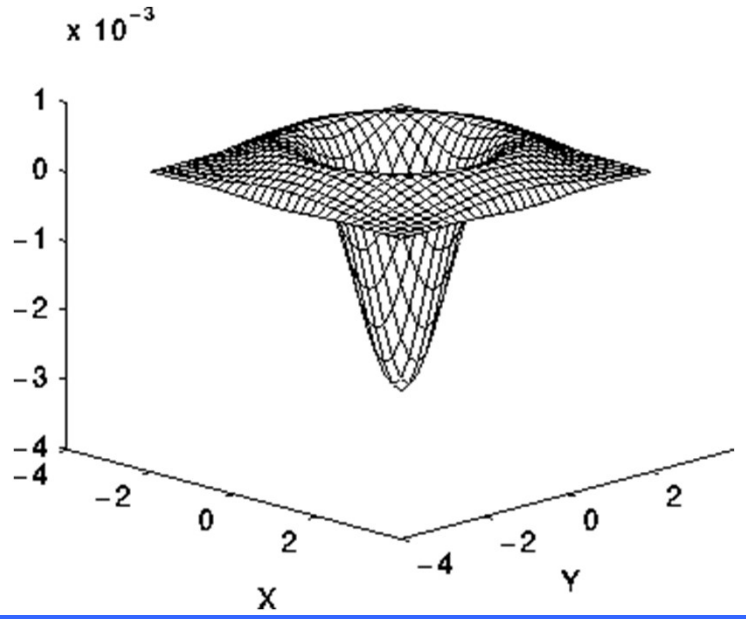
∇^2 is the **Laplacian** operator: $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$

Cont…

- The order of performing differentiation and convolution can be interchanged because of the linearity of the operators involved:

$$g(x,y) = \left[\nabla^2 G(x,y)\right] * f(x,y)$$

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$



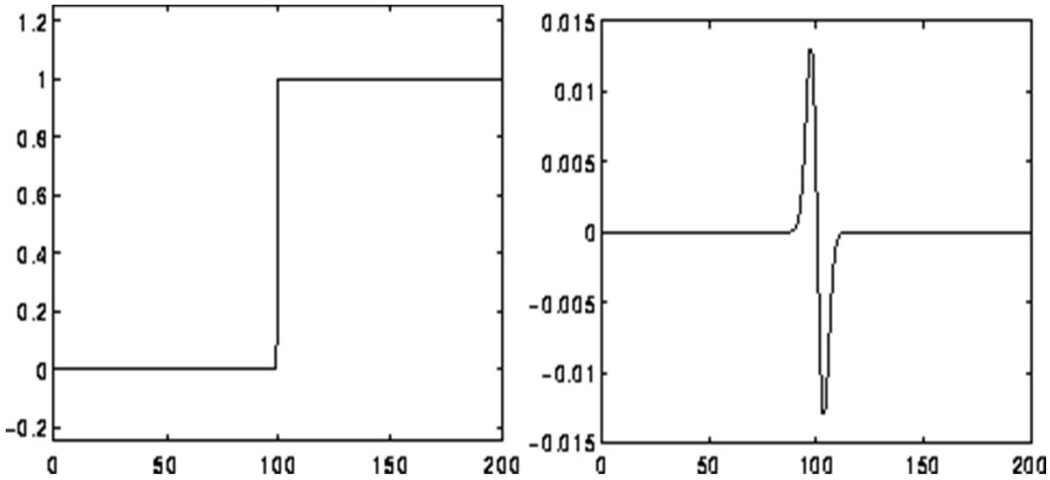
LoG - Guidelines for Use

- The LoG operator calculates the second spatial derivative of an image. This means that in areas where the image has a constant intensity (*i.e.* where the intensity gradient is zero), the LoG response will be zero.
- In the vicinity of a change in intensity, however, the LoG response will be positive on the darker side, and negative on the lighter side. This means that at a reasonably sharp edge between two regions of uniform but different intensities, the LoG response will be:

Cont…

- zero at a long distance from the edge,
- positive just to one side of the edge,
- negative just to the other side of the edge,
- zero at some point in between, on the edge itself.

Cont…



- Response of 1-D LoG filter to a step edge. The left hand graph shows a 1-D image, 200 pixels long, containing a step edge. The right hand graph shows the response of a 1-D LoG filter with Gaussian = 3 pixels.

Applications of LoG

- Spurious edges detected away from any obvious edges hence to increase the smoothing of the Gaussian to preserve only strong edges.
- The gradient of the LoG at the zero crossing (*i.e.* the third derivative of the original image) and only keep zero crossings where this is above a certain threshold. This will tend to retain only the stronger edges, but it is sensitive to noise, since the third derivative will greatly amplify any high frequency noise in the image.

Coarse to Fine Tracking

- Useful to track edges at a variety of scales
- Edge detection at fine scale picks up a lot of noisy / minor texture edges
- Edge detection at coarse scale only picks up broad edges, missing out fine detail
- Option to get edges of varying detail, without noise is to track edges from coarse to fine scale
- This method is also referred to as edge focusing

Edge Focusing

- Start with zero crossings at a coarse scale, say at sigma = 5
- Record the positions of the edge pixels at the coarse scale
- Reduce sigma by a step size *s (e.g. 0.5)*
- Repeat the zero crossing detection
- Retain only those zero crossings at the smaller sigma level that are in the neighborhood of the zero crossings at the coarse scale

Cont…

- Reduce current sigma by *s* again
- Repeat the process till the desired scale of sigma is reached

Advantage :

- The degree of detail can be controlled by the choice of starting sigma and step size.
- Computational load can be reduced by limiting the zero crossing detection at smaller sigma values to those locations that are in the neighborhood of those detected at the previous level

Refer

- F. Bergholm's paper "Edge Focusing" in IEEE Trans. Pattern Analysis and Machine Intelligence, November 1987.

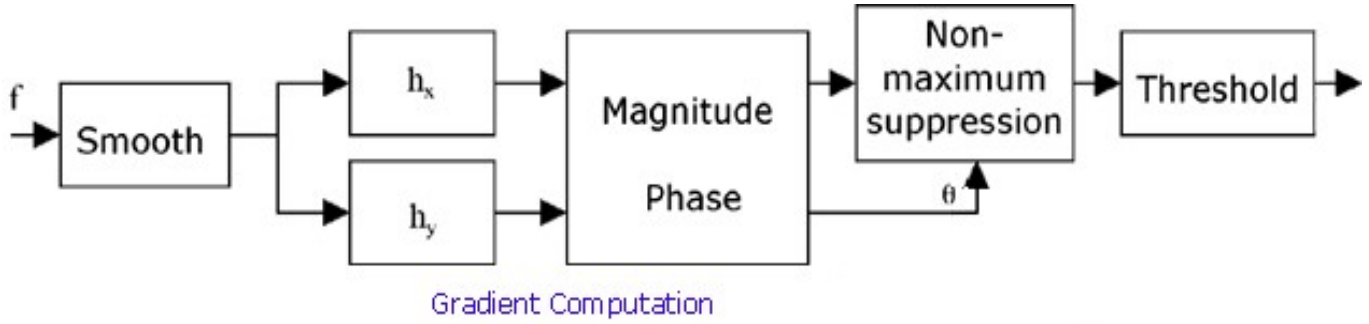
The Canny Edge Detector

The mask we want to use for edge detection should have certain desirable characteristics, called Canny's criteria:

1. Low probability of missing a genuine edge
2. Good locality, i.e the edge should be detected where it actually is.
3. Single response for an edge
4. Low probability of false alarm

Cont…

- There are four steps following the diagram:



1. Smoothing: using a gaussian smoothing operator
2. Gradient
3. Non-Maximum Suppression
4. Hysteresis Threshold

Smoothing

- For the smoothing step ➔ Gaussian LPF.
- The standard deviation, determines the width of the filter and hence the amount of smoothing.

$$S(x,y) = G(x,y,\sigma) * f(x,y)$$

- Let f(x,y) denote the input image. The result from convolving the image with Gaussian smoothing filter using separable filtering is an array of smoothed data.
- *S(x,y)* = The spread of the Gaussian and controls the degree of smoothing.

Cont…

The edge enhancement step simply involves calculation of the gradient vector at each pixel of the smoothed image.

$$g_x(x,y) \approx S(x+1,y) - S(x-1,y) \quad g_y(x,y) \approx S(x,y+1) - S(x,y-1)$$

$$Magnitude(x,y) \quad = |g| = \sqrt{g_x^2 + g_y^2}$$

$$\theta = \tan^{-1}\left(\frac{g_y}{g_x}\right)$$

Gradient

- At each point convolve with

$$G_x = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}$$

- Magnitude and Orientation of the Gradient are computed as

$$M[i,j] = \sqrt{P[i,j]^2 + Q[i,j]^2}$$

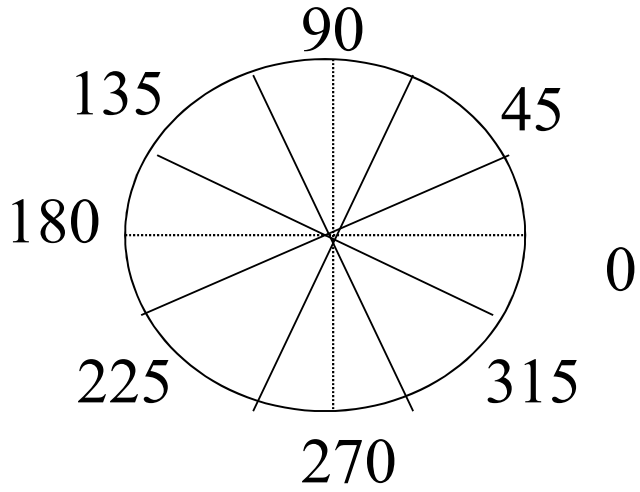
$$\theta[i,j] = \tan^{-1}(Q[i,j], P[i,j])$$

- **Avoid floating point arithmetic for fast computation**

NMS

- The localization step has two stages: non-maximal suppression and hysteresis thresholding

Non-Maximal Suppression (NMS) thins the ridges of gradient magnitude in magnitude image by suppressing all values along the line of the gradient that are not peak values of a ridge .



$$\theta_s = Sector\ [\theta(x, y)]$$

NMS

- Search in a 3x3 neighborhood at every point in magnitude image
- Consider the gradient direction at the centre pixel
- Compare the edge magnitude at centre pixel with its two neighbors along the gradient direction.
- If the magnitude value at the center point is not greater than both of the neighbor magnitudes along the gradient direction, then g(x,y) is set zero. The values for the height of the ridge are retained in the NMS magnitude.

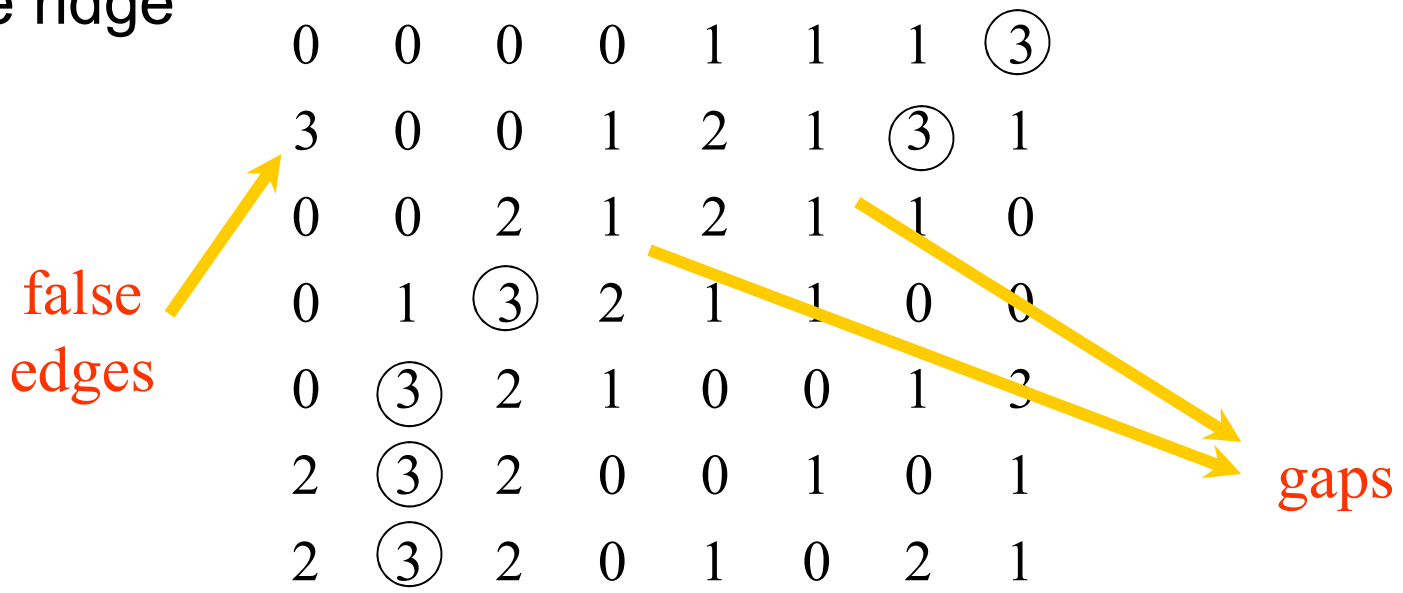
$$g_{\text{suppressed image}} = \text{nms}\,[g(x,y),\ \theta_s]$$

Non-Maxima Suppression

- After nonmaxima suppression one ends up with an image which is zero everywhere except the local maxima points.
- Thin edges by keeping large values of Gradient

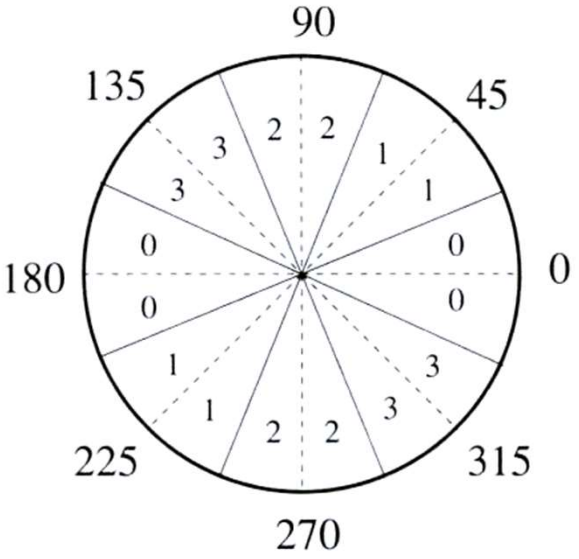
Principle of NMS

- Thin the broad ridges in M[i,j] into ridges that are only one pixel wide
- Find local maxima in M[i,j] by suppressing all values along the line of the Gradient that are not peak values of the ridge

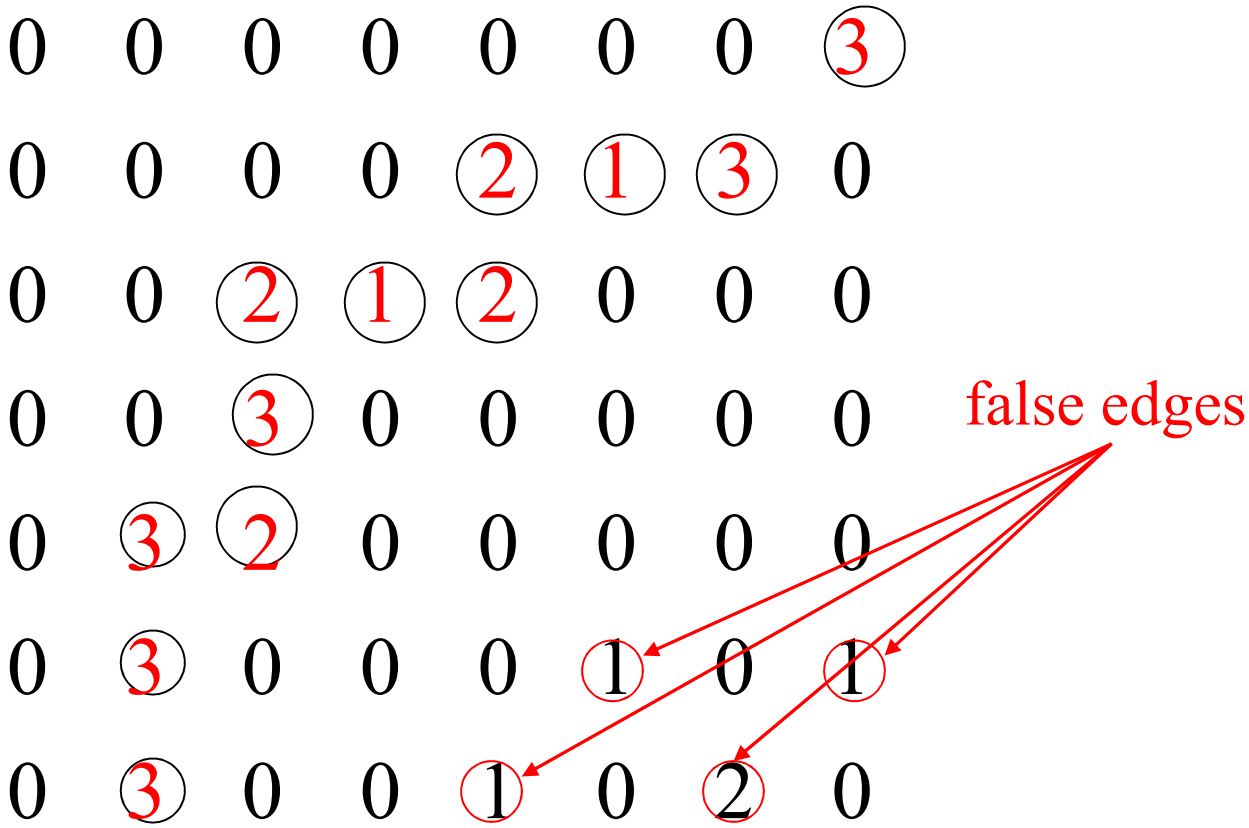


Gradient Orientation

- Reduce angle of Gradient θ[i,j] to one of the 4 sectors
- Check the 3x3 region of each M[i,j]
- If the value at the center is not greater than the 2 values along the gradient, then M[i,j] is set to 0



The suppressed magnitude image will contain many false edges caused by noise or fine texture



NMS

Definite Edges

Definite non-edges

To be considered

Thresholding

- Reduce number of false edges by applying a threshold T
  - all values below T are changed to 0
  - selecting a good value for T is difficult
  - some false edges will remain if T is too low
  - some edges will disappear if T is too high
  - some edges will disappear due to softening of the edge contrast by shadows

Double Thresholding

- Apply two thresholds in the suppressed image
  - $T_2 > T_1$
  - two images in the output
  - the image from T_2 contains fewer edges but has gaps in the contours
  - the image from T_1 has many false edges
  - combine the results from T_1 and T_2
  - link the edges of T_2 into contours until we reach a gap
  - link the edge from T_2 with edge pixels from a T_1 contour until a T_2 edge is found again

Input Image

Canny Output s=1, T1=0.2, T2=0.5

Canny Output s=1, T1=0.4, T2=0.7

Input Image

Canny Output s=2, T1=0.2, T2=0.5

Canny Output s=2, T1=0.4, T2=0.7

Input Image

Canny Output s=5, T1=0.2, T2=0.5

Canny Output s=5, T1=0.4, T2=0.7

Color Edge Detection

- Color images present a challenge in detecting edges due to higher dimensionality
- A simple approach to detecting edges based on color differences:
  - Apply RGB-HSI transform
  - Apply conventional edge detectors on Hue component

Color Edge Detection

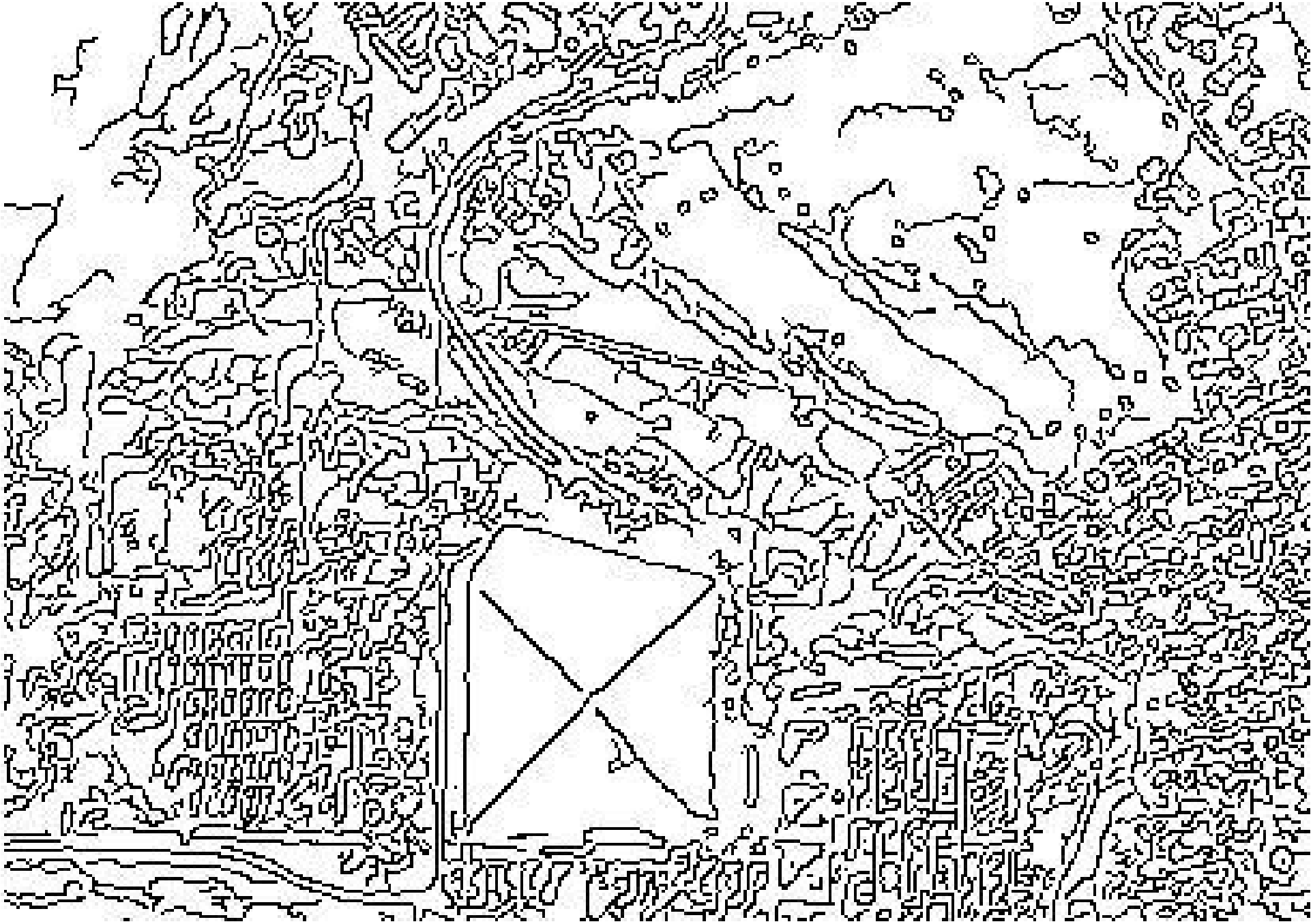
- Approach 2:
- Detect edges independently in Hue, Saturation and Intensity components
- Compute union of all the edges detected
- This will ensure that whether the edge is caused due to color, intensity or saturation it will still be detected

Color Edge Detection

- Approach 3:
- Compute the gradient in hue, saturation and intensity components
- Compute the edge gradient magnitude as the maximum of the three components

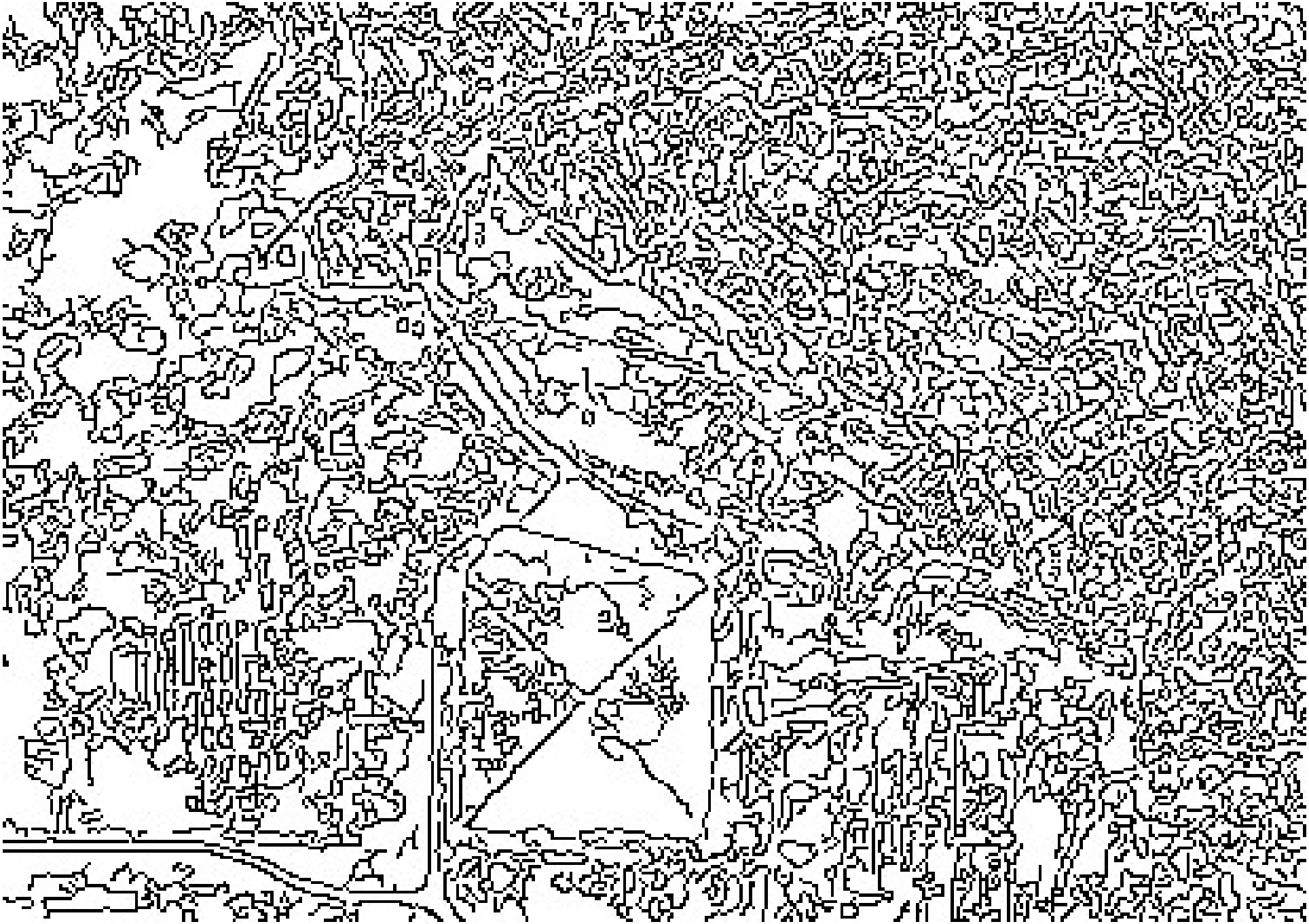
Input Image

PC1 of Multiband Image



PC1 Canny Output s=1, T1=0.2, T2=0.5

Hue Image



Canny Output s=1, T1=0.2, T2=0.5

Mathematical Morphology Approach

- Color gradient
- This can be computed in terms of morphological dilation and erosion on vectors.
- e($\underline{X}$) = dil($\underline{X}$,S) – ero($\underline{X}$,S)
- dil($\underline{X}$,S) is dilation of $\underline{X}$ by a structuring element S, usually a 3x3 box
- ero($\underline{X}$,S) is erosion of $\underline{X}$ by a structuring element S, usually a 3x3 box

Mathematical Morphology Approach

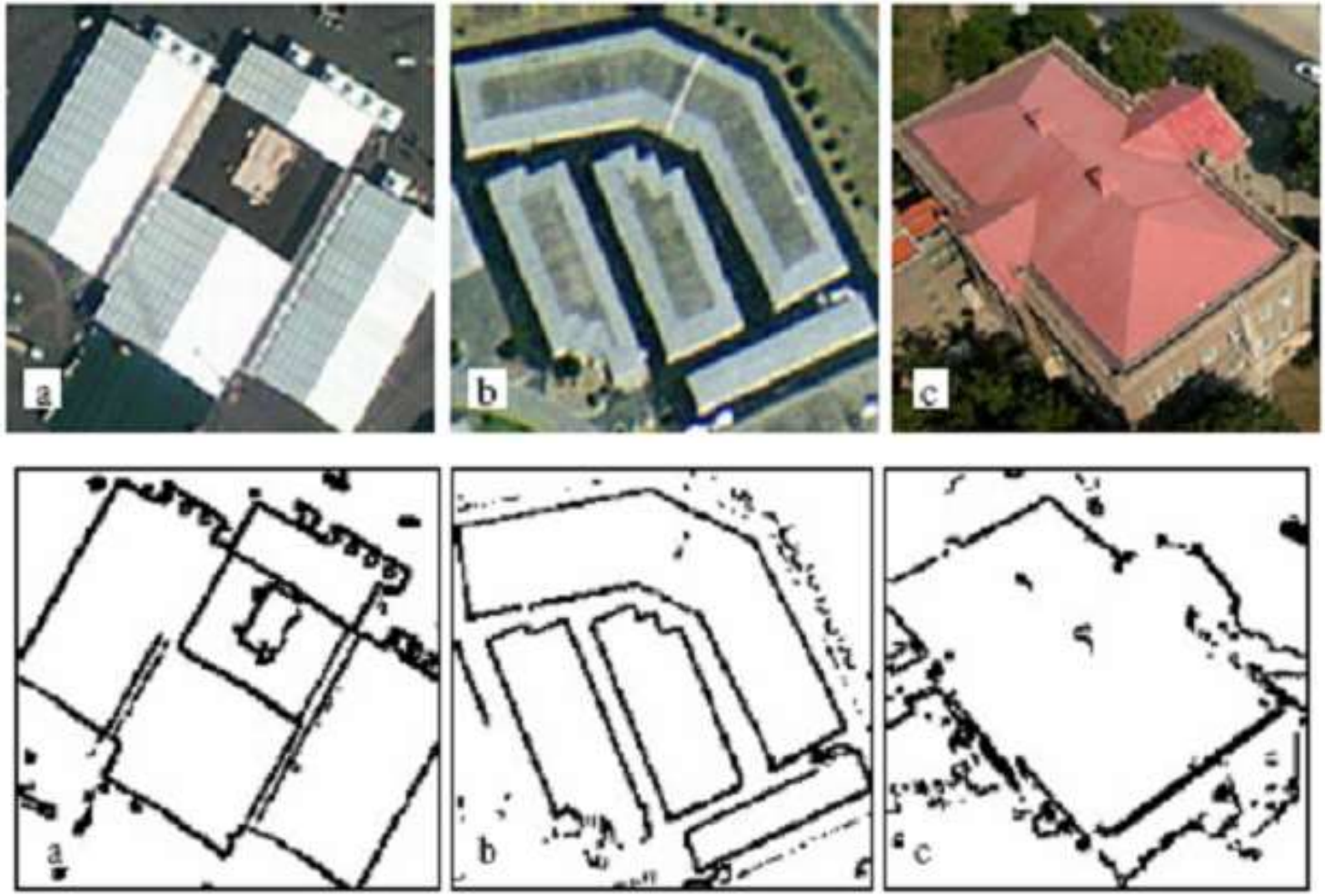
- Gray scale dilation depends on local maximum and erosion depends on local minimum. In case of color (i.e., 3-element vectors), what is maximum and what is minimum? In other words, how are the vectors ordered in increasing or decreasing order?

Computing Vector Dilation

- Consider n vectors in a neighborhood, written as $X = [X_1, X_2, \ldots, X_n]^T$, that is
- X is a nx3 matrix, each row a color vector or size 3, having elements

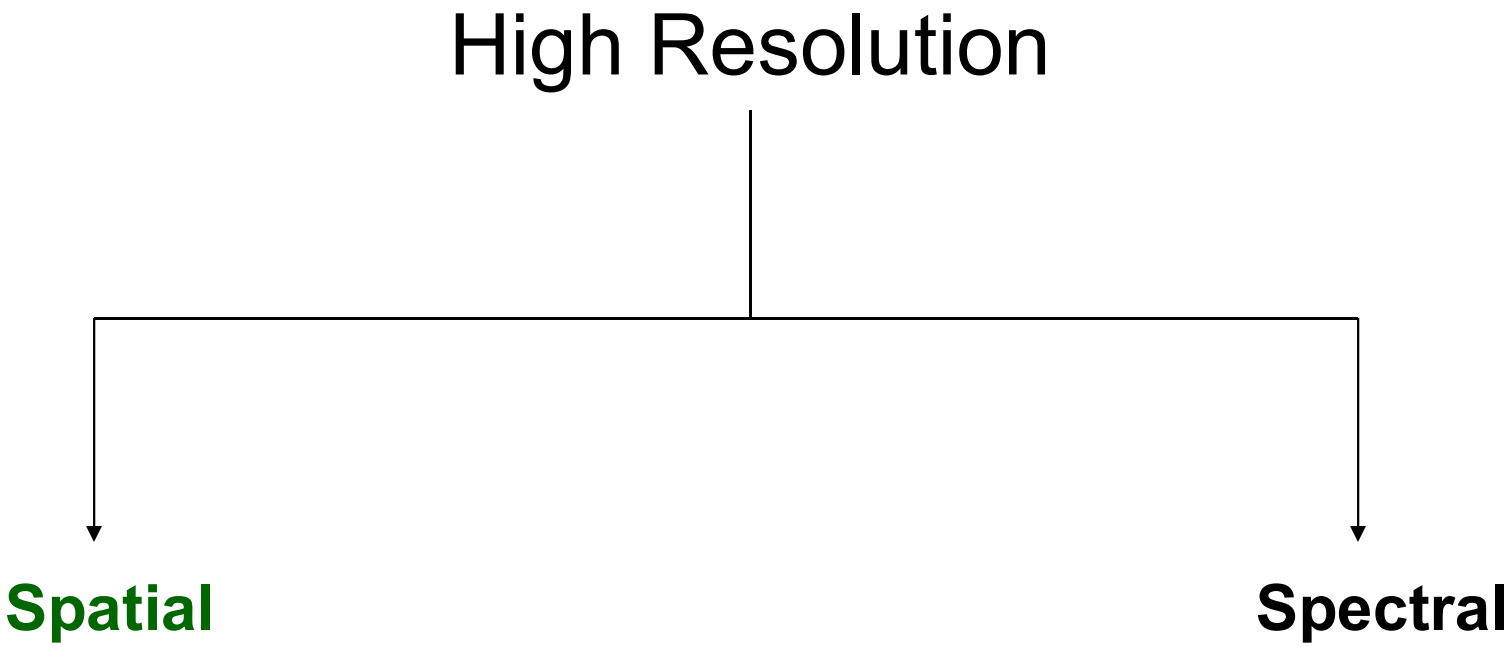
  (r_i, g_i, b_i).
- Then Max(X) ≅
  - $[\max_i(r_i), \max_i(g_i), \max_i(b_i)]^T$
  - Similarly Min(X) is also defined

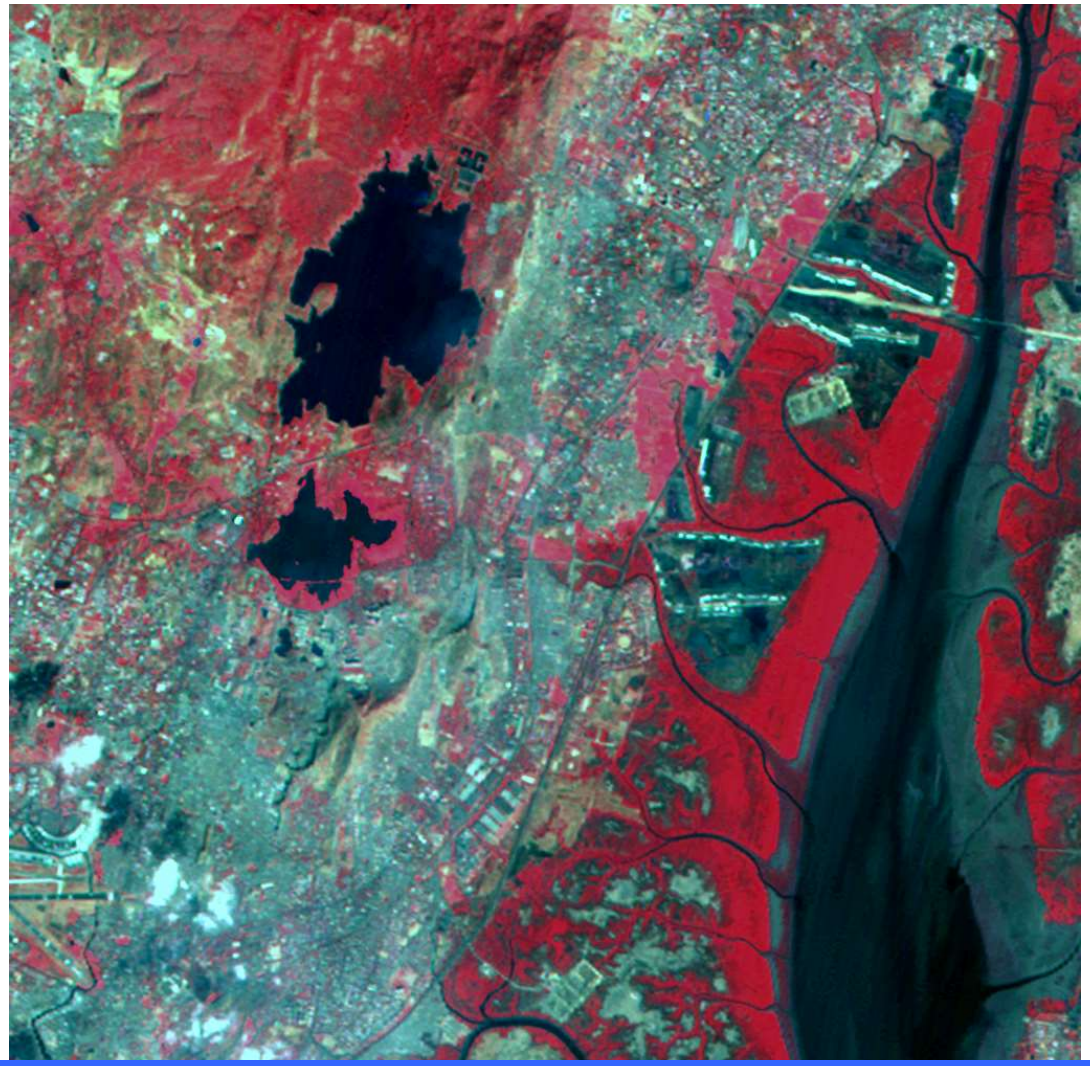
Example



Region Segmentation

High Resolution





IRS-1C LISS3
23.5m

IRSP6 LISS4 Image

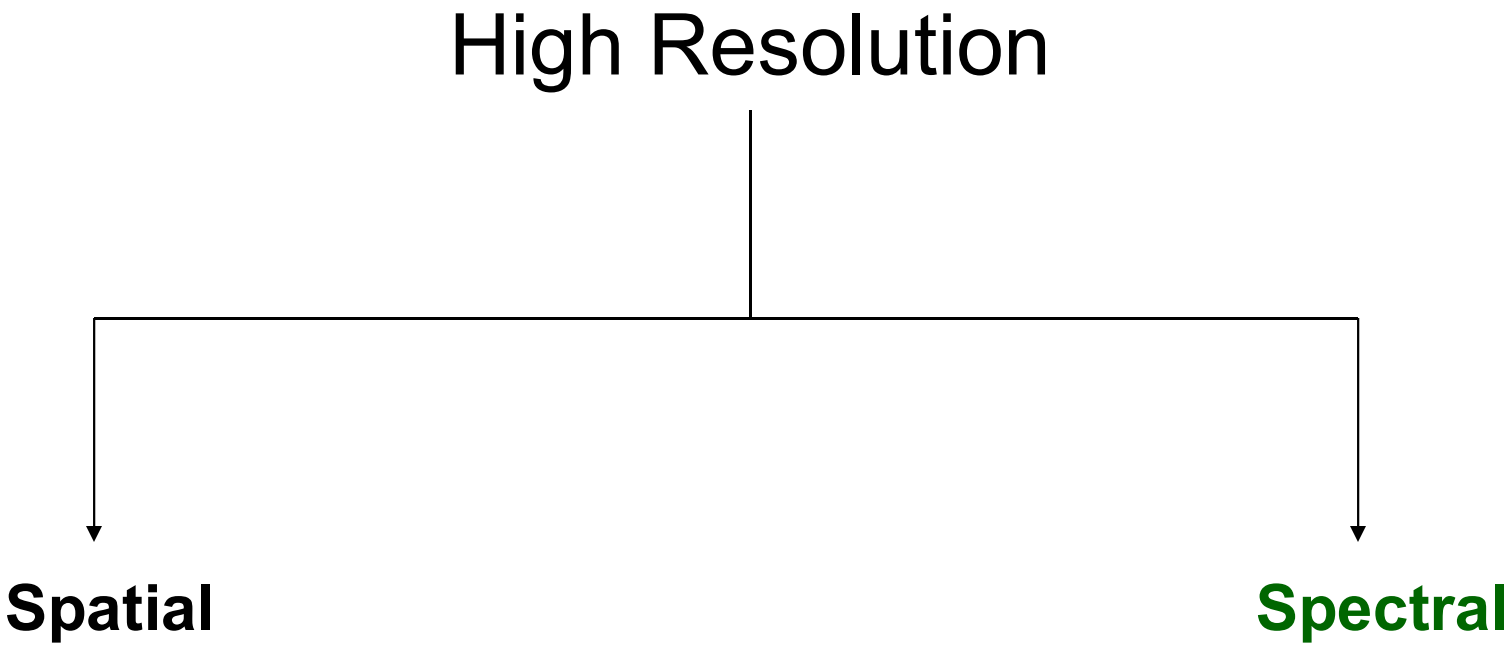
Quickbird Window

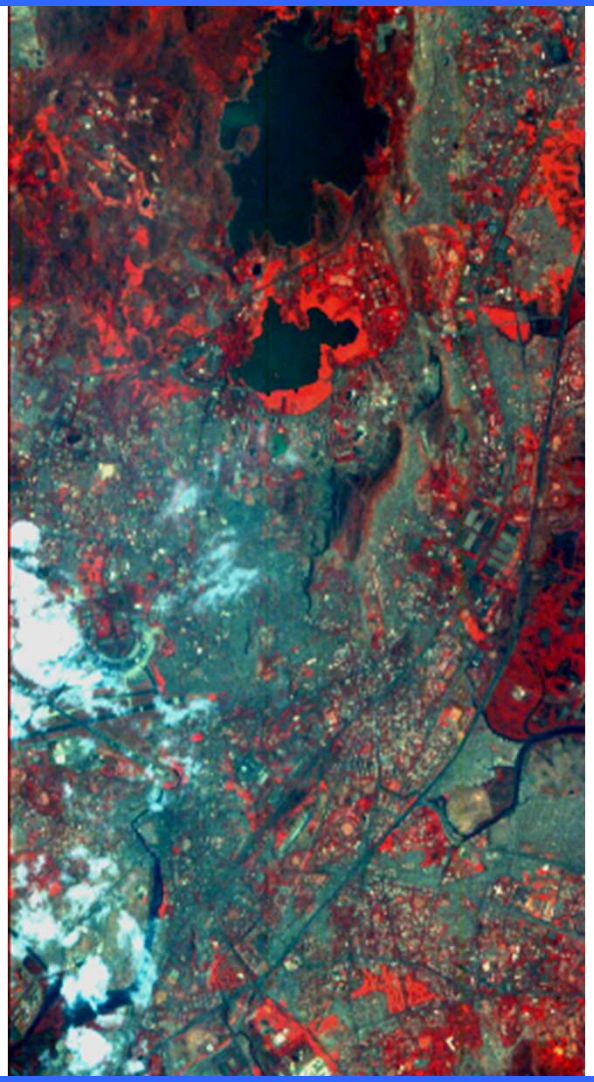
AIRBORNE

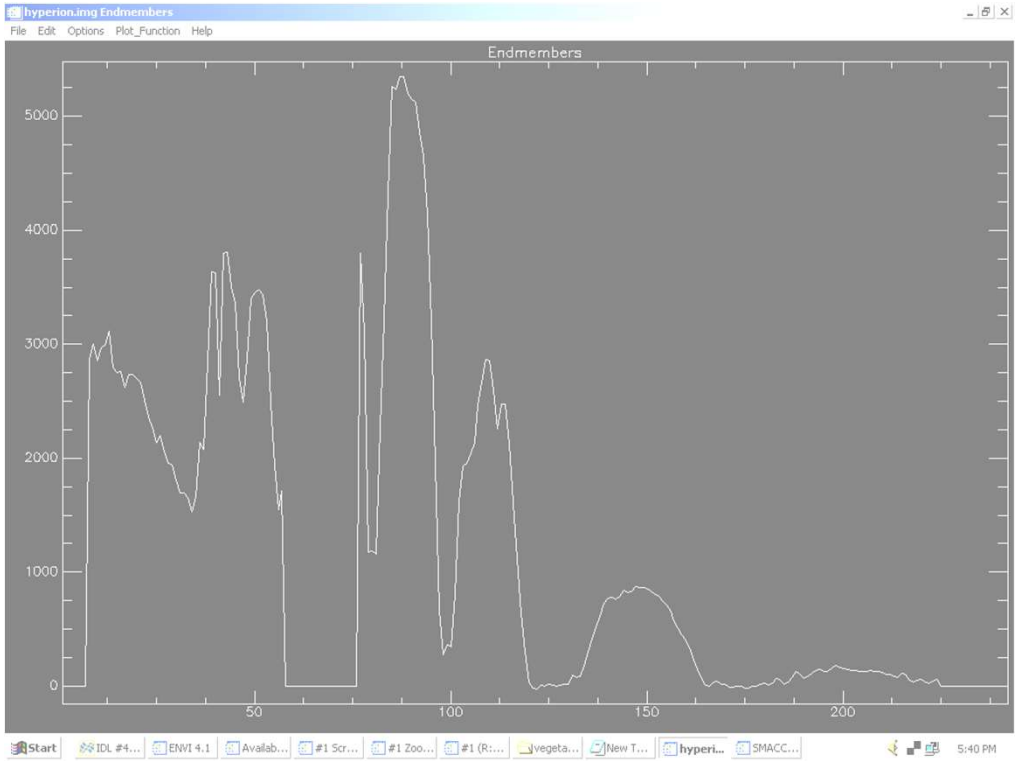
NATURAL COLOR

COMPOSITE

High Resolution







Hyperspectral Image Window and Spectrum of Vegetation near Powai Lake

Evolution of Segmentation Techniques

- Pixel based classification using spectral features (Landsat, IRS 1C/1D, SPOT 1,2,3)
- Pixel based classification using spectral and textural features (IRS P6, SPOT 5, …)
- Object based classification using spatial features, spectral and textural features

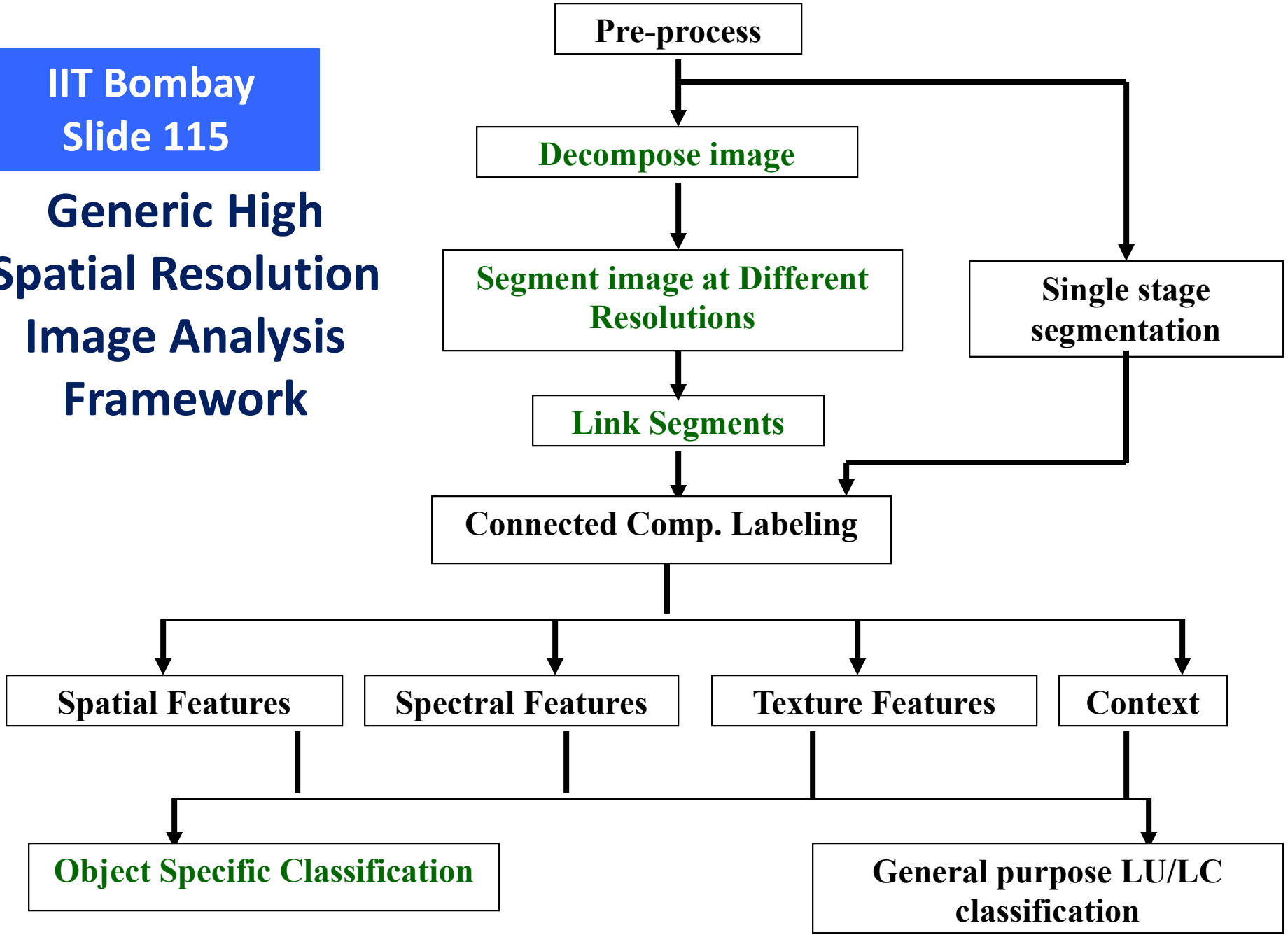
High Resolution Images

- High resolution images are information rich
  - Spatial information
  - Multispectral information
  - Textural information
- Image can be viewed as a collection of objects with spatial relationships

General Methodology

- **Smooth image (if needed)**
- **Segment the image into regions**
  - Find seed points to start region growing
    - Find gradient
    - Find gradient minima
  - Grow regions from seed points
    - Use suitable algorithm to start looking for similar neighbors such that we can grow from seed points to regions
- **Compute features for each region**
  - Shape
  - Textural
  - Contextual
  - Spectral
- **Classify regions**

Generic High Spatial Resolution Image Analysis Framework



Step 1

Preprocessor

- Image smoothing
- Suppress noise / eliminate minute detail that is not of interest
- Adaptive Gaussian / Median
- Very useful to produce proper segmentation
- Optional step

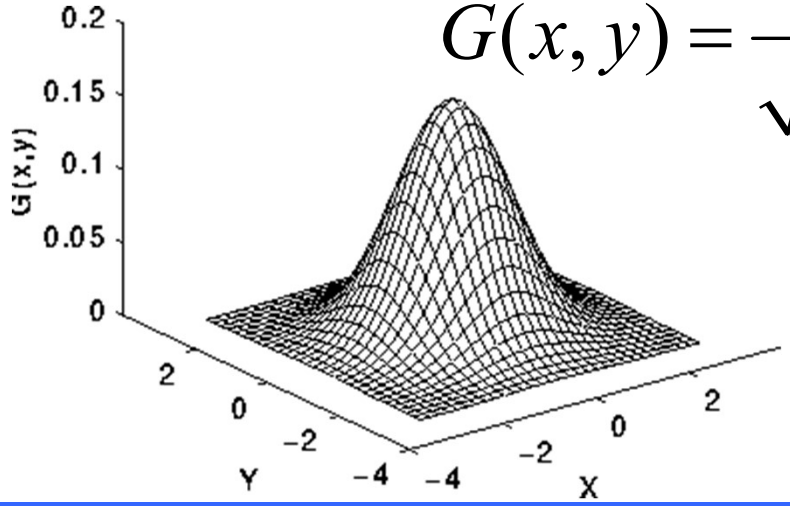
Preprocessing

Mean Filter

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Gaussian Filter

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(x^2+y^2\right)}{2\sigma^2}}$$



123	125	126	130	140
122	124	126	127	135
118	120	150	125	134
119	115	119	123	133
111	116	110	120	130

- Median Filter

Neighbourhood values:

115, 119, 120, 123, 124,
125, 126, 127, 150

Median value: 124

Preprocessing

(close-open) alternating sequential morphological filter

Adaptive Smoothing Algorithm

- Compute the image gradient at a pixel (x,y) as

$$\left(\frac{\partial I^t(x,y)}{\partial x}, \frac{\partial I^t(x,y)}{\partial y} \right)^T$$

- $=(G_x, G_y)^T$
- where superscript *t* in I^t stands for iteration index.

Algorithm

- Since the weights assigned to neighboring pixels decrease with magnitude of the gradient, we consider
- $w(x,y) = e^{-\left|\frac{d^t(x,y)}{2k^2}\right|^2}$

where $d^t(x,y)$ is the gradient, and

$|d^t(x,y)|^2 = \text{sqrt}[G_x^2 + G_y^2]$

Algorithm

- The parameter *k* is user-specified, and the rate of decrease in neighbor weights with distance from the centre pixel (x,y) depends on the value of k. If k is small, then the centre pixel is given large weightage, and hence little smoothing. If k is large, then neighbors get more weightage, and hence more smoothing.

Algorithm

- The smoothed signal is given by

$$I^{t+1}(x,y) = \frac{1}{N^t} \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} I(x+i, y+j) w^t(x+i, y+j)$$

- This is obviously space-variant filtering

- $N^t = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} w^t(x+i, y+j)$

Performance of the algorithm

- In practice, within a few iterations, the edges become sharp, with interior details of regions smoothed
- $I^{t+1}(x,y) = I^{t}(x,y)$, happens after many iterations
- The process is terminated after a few iterations when the smoothing is perceived to be adequate

Performance of the Algorithm

- The weights assigned to the neighbors w decrease as the gradient magnitude d increases.
- When there are discontinuities, the neighboring pixels are assigned small weights, resulting in edge preserving smoothing
- Micro-textures are filtered out

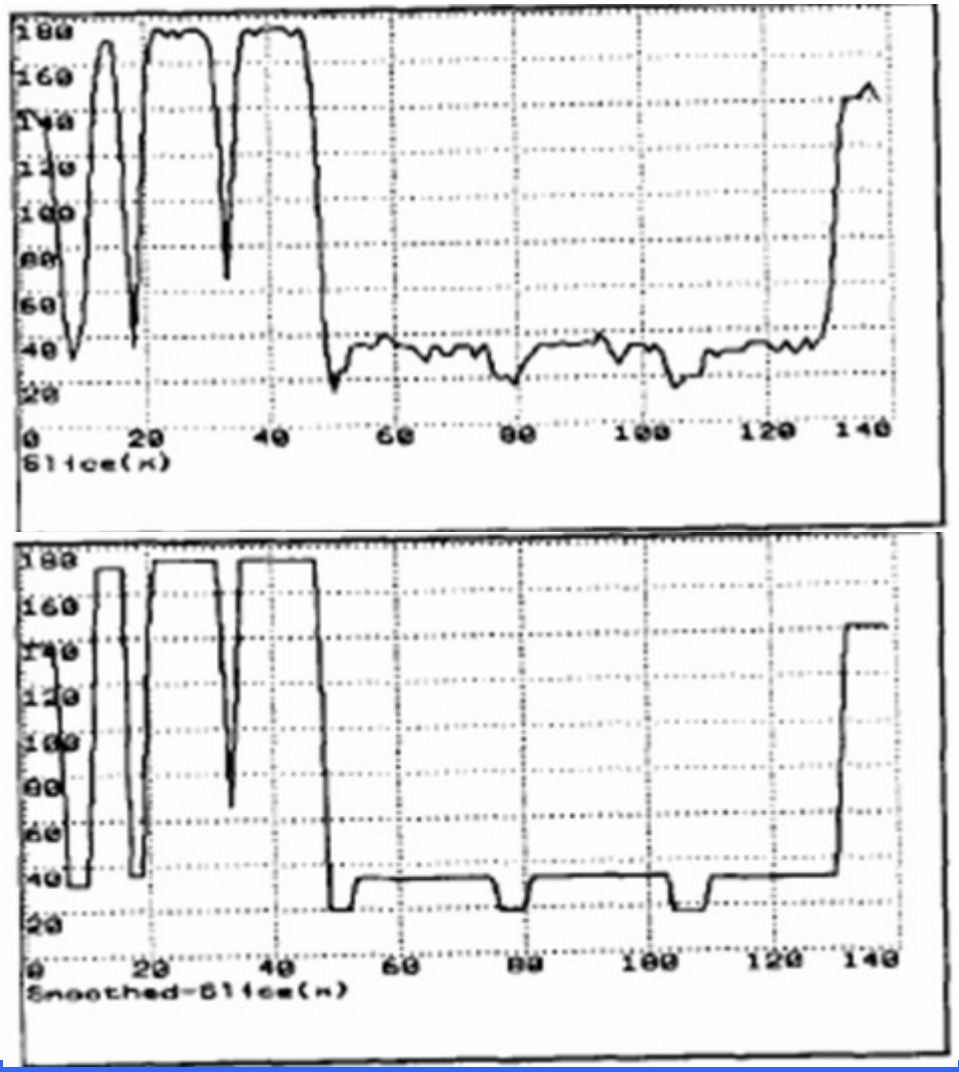


Illustration of the working of edge-preserving smoothing

Region Based Image Analysis

- Images contain spatially contiguous blocks of similar pixels or *regions*
- A region corresponds to an object or part of a real world object
- Region has spatial, spectral and shape properties
- Regions have connectivity to adjacent regions

How is region based image analysis performed?

- We assume that the images we consider are of high spatial resolution to employ region based methods
- e.g.
  - Quickbird, Ikonos, GeoEye, Cartosat2
  - Aerial imagery

Issues with high resolution imagery

- **Advantages**
  - High spatial resolution
  - Individual objects can be extracted and counted
- **Limitations**
  - Large data volume
  - Some of the objects may be irrelevant! e.g., small repair patches on roads, potholes

Region Based Approches

- **Region Growing**
  - Start with individual pixels
  - Add neighbors with similar attributes to form regions
  - If neighbor is very different, do not include

This is a bottom-up scheme, starting with pixels, going up to form regions

Region Based Approches

- **Region Splitting Approaches**
  - Consider entire image as a single region
  - Split region if homogeneity criterion fails
  - Now consider each region separately and repeat the process recursively

This is a top-down process, starting from full image, towards small regions

Region Based Approches

- Hybrid Approaches
  - Initially split using top-down methods
  - Small regions that are spatially adjacent and having similar properties are merged to form larger regions

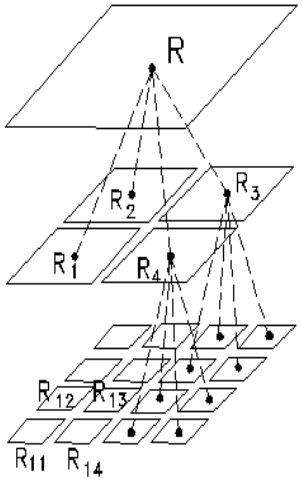
This is the well known strategy of split-and-merge

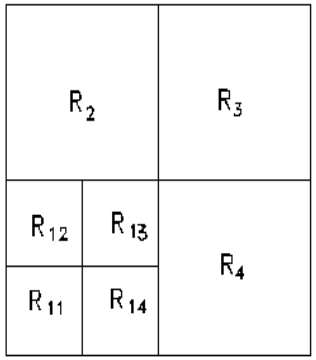
Example

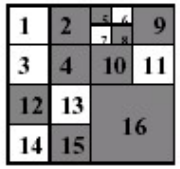
- Split image using a quadtree hierarchy
- Continue till the leaf nodes are small
- Consider the leaf nodes that correspond to spatially adjacent parts of the image
- Compare means and variances
- If similar, merge

Final result is no longer a QuadTree!

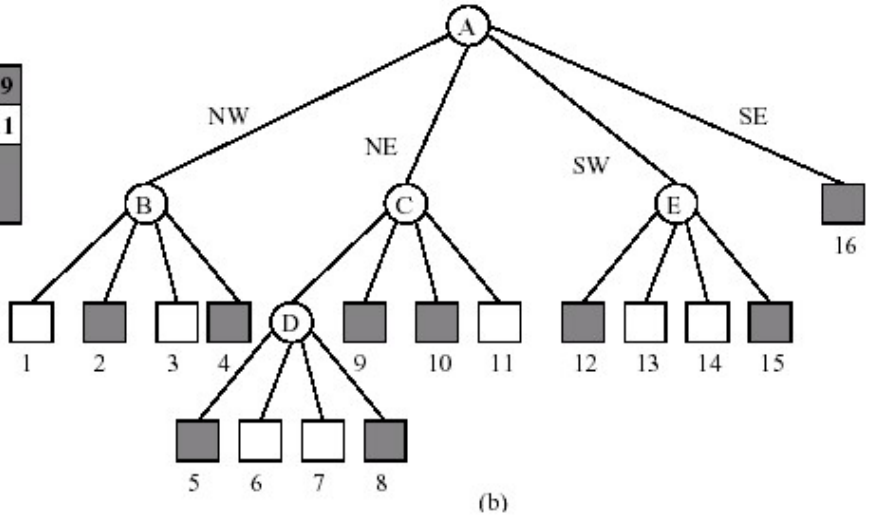
Quadtree Based Split and Merge







(a)



(b)

Tree structure

Parent Node

NW node

NE node

SW node

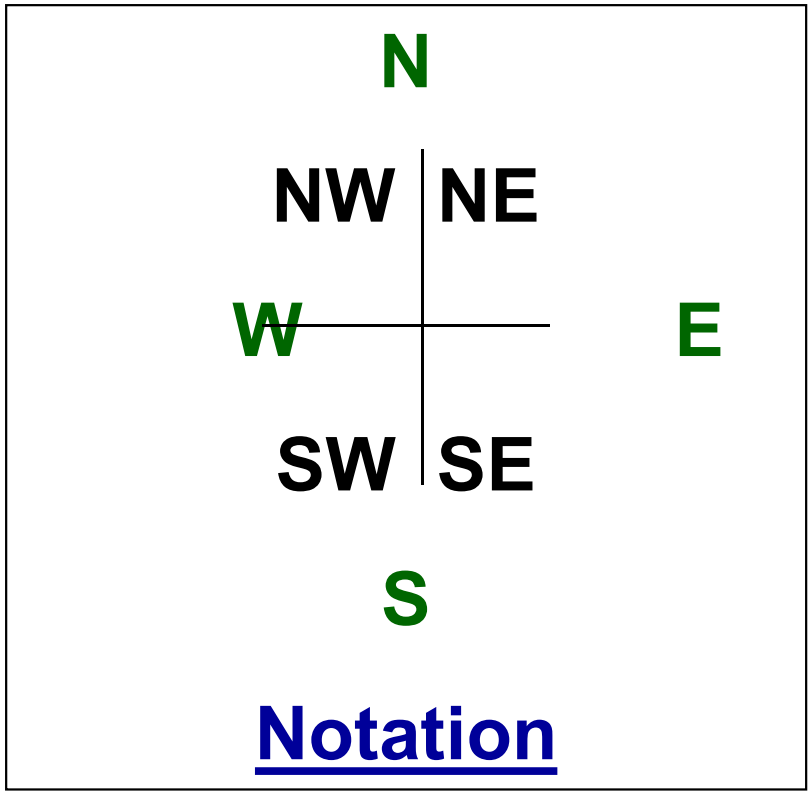
SE node

Nodetype *(root/leaf/intermediate)*

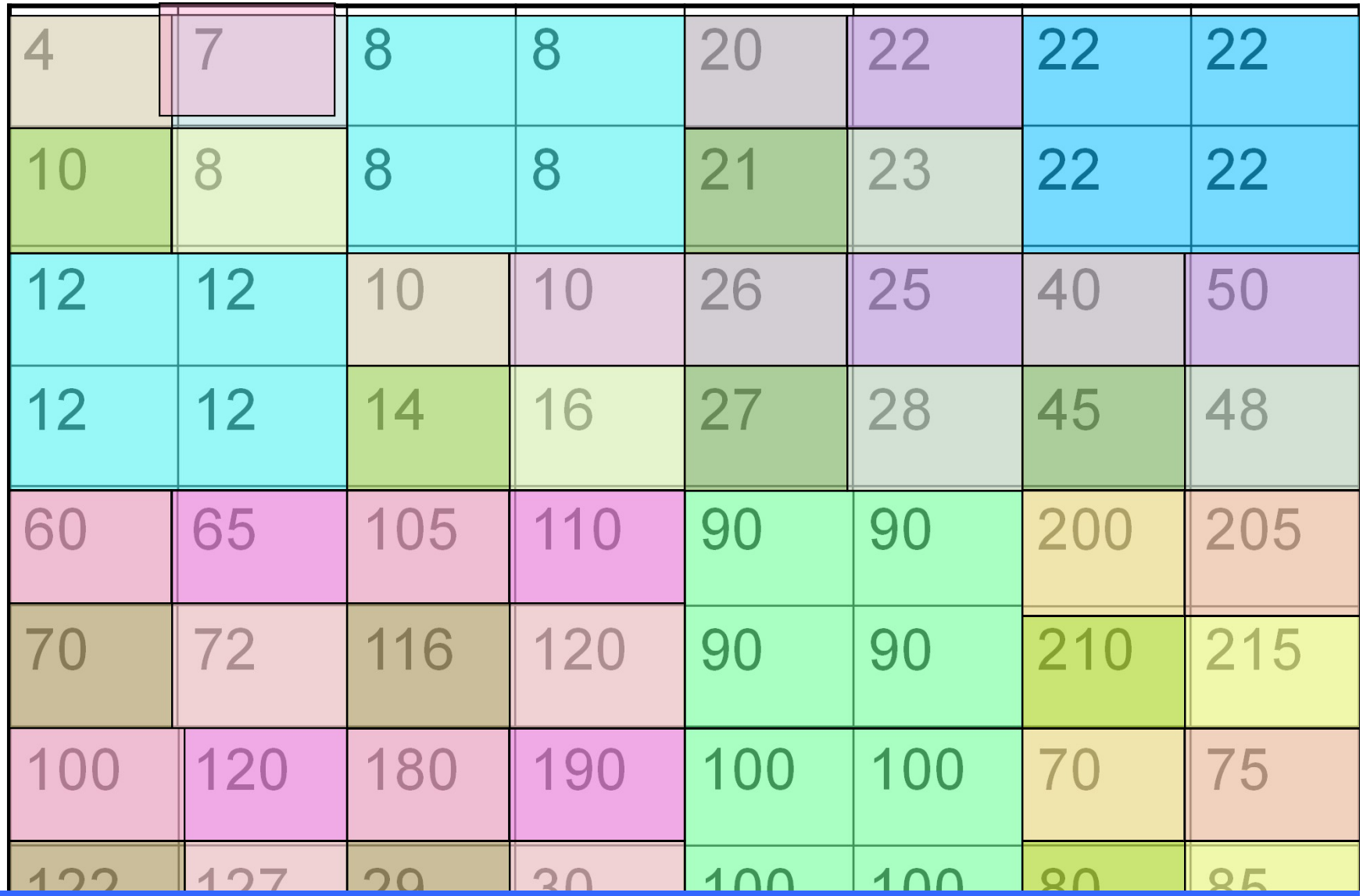
Size

Mean

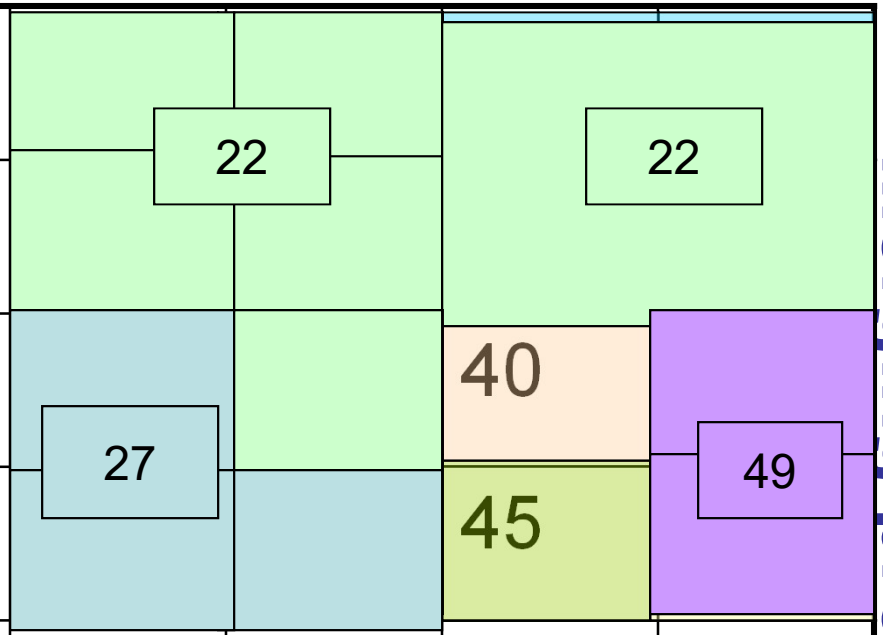
Texture



Splitting of an image

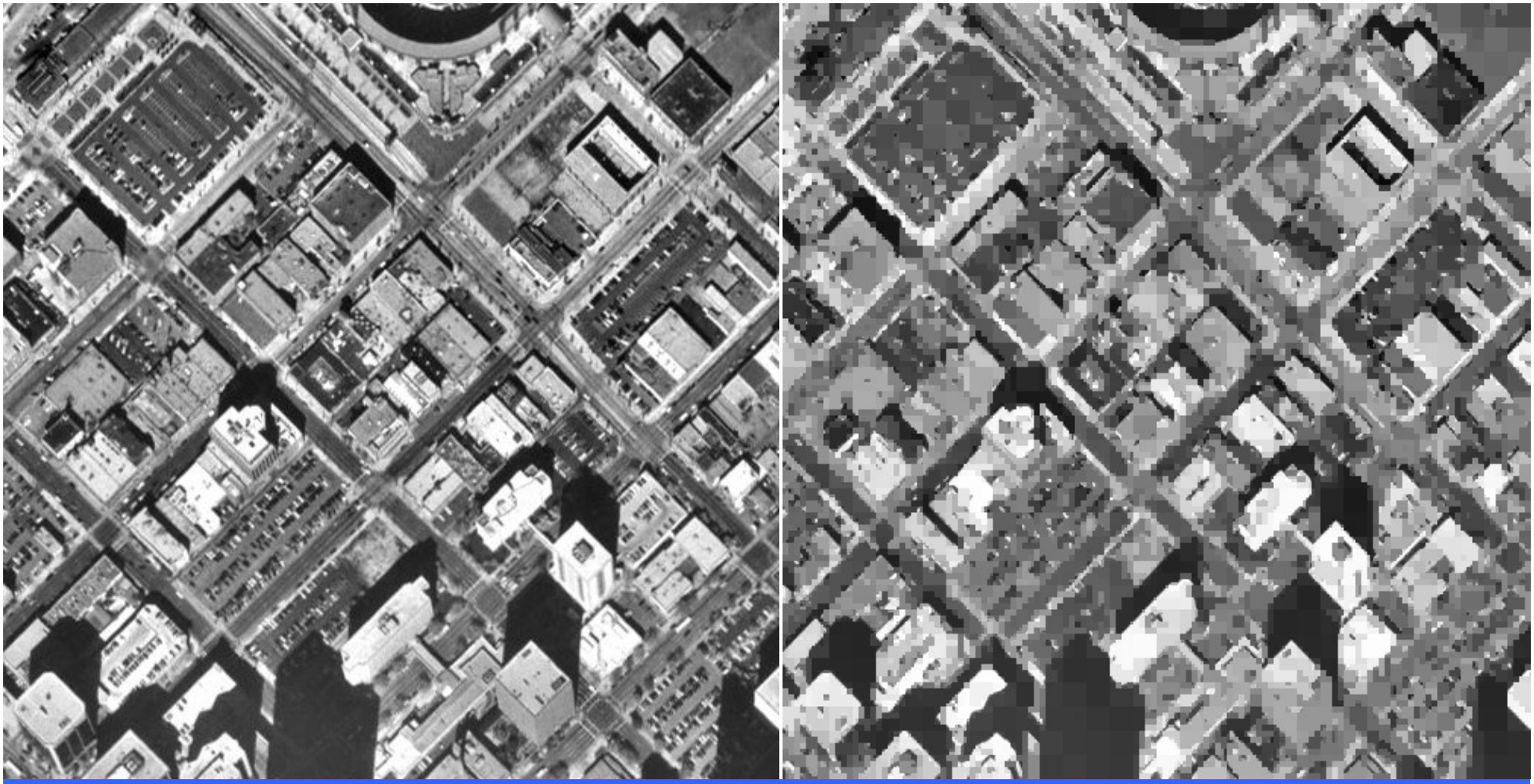


4	7	8	8				
10	8	8	8				
12	12	10	10				
12	12	14	16				
60	65	105	110	90	90	200	205
70	72	116	120	90	90	210	215
100	120	180	190	100	100	70	75
122	127	29	30	100	100	80	85



Merging process

Quadtree Segmentation with Merging



Comments

- Region shapes can be jagged, due to the shape of the sub-images spanned by each of the quadrants and sub-quadrants
- Splitting can be fast, but merging will be slow

Region Growing Methods

- Issues
  - Identification of seed pixels from where to start region growing
  - Different seed pixels can result in different final results – sequential procedure!
  - Alternatively, generate multiple seed pixels or micro-regions and grow regions in parallel

Segmentation Algorithms

- Region Based
  - Seeded Region Growing (SRG)
  - Quadtree based split and merge
- Edge Based
  - Sobel
  - Laplacian
  - Canny's
- Hybrid
  - Morphological Watershed transform

Calculation of Seed Pixels

- For simple intensity images, one can choose the very bright and very dark pixels to start region growing
- In other words, one can compute the white top-hat and black top-hat transformations and use the resulting pixels as seeds
- One can optionally apply a threshold to limit the number of seed pixels

Calculation of Seed Pixels

- Optionally, one can compute gradient image, and pick pixels that are gradient minima.
- This is applicable since region interiors are where the intensity gradient is minimum. Gradient maxima occur near region boundaries.

Calculation of Seed Pixels

- In case of color images or multidimensional data?
- Option 1: Compute PC1 of the higher dimensional data, and use gradient minima or locally bright and dark pixels as seeds.

Calculation of Seed Pixels

- Option 2: Color gradient
- Compute color gradient and locate local gradient minima pixels which will be used as seed pixels.

Calculation of Seed Pixels

- Apply clustering and pick pixels whose feature vectors are very close to the cluster mean vectors. Usually these will be well inside the regions and can be used as seed pixels

Seed Pixel Selection

- Given the cluster centres $\underline{C}_i$, the seed pixels are those whose feature vectors are within a user specified distance (in the feature space) from the cluster centres.
- $\underline{X}$ is a seed pixel if
- $||\underline{X} - \underline{C}_i||_p <= \epsilon$
- The cluster centres normally coincide with region interiors.

Seeded Region Growing

- Consider the image being decomposed into a collection of regions A_i.
- Initially the regions A_i would start with one or a small number of pixels based on the initialization strategy
- Starting with the seed pixels above, the remaining pixels **x** are assigned to different regions based on the criteria chosen

Adams and Bischof, IEEE T-PAMI, vol. 16, no.6, pp.641-647

Seeded Region Growing

- Label seed points according to their initial grouping.

$$T = \{x \notin \bigcup_{i=1}^{n} A_i \mid N(x) \cap \bigcup_{i=1}^{n} A_i \neq \phi\}$$

Compute

$$\delta(x) = | g(x) - mean[g(y)] |$$

to determine whether x can be added to region y.

Small value of δ(x) means high affinity of x to region y.

The algorithm for implementing SRG (boundary flagging case) is as follows:

Label seed points according their initial grouping.
Put neighbors of seed points (the initial T) in the SSL.
While the SSL is not empty:
  Remove first point y from SSL.
  Test the neighbors of this point:
    If all neighbors of y which are already labeled
    (other than with the boundary label) have
    the same label—
      Set y to this label.
      Update running mean of corresponding region.
      Add neighbors of y which are neither already
      set nor already in the SSL to the SSL according
      to their value of δ. (See note below).
    Otherwise
      Flag y with the boundary label.

Pseudo-code

Algorithm

- If the neighbors of an unassigned pixel meet more than one region, then we consider

$$\delta(x) = \min_i \left| g(x) - mean[g(y_i)] \right|$$

- where y_i is a pixel belonging to region i.

Algorithm

- Alternatively, x can be labeled as a boundary pixel, and we append a pixel z to region A_i such that

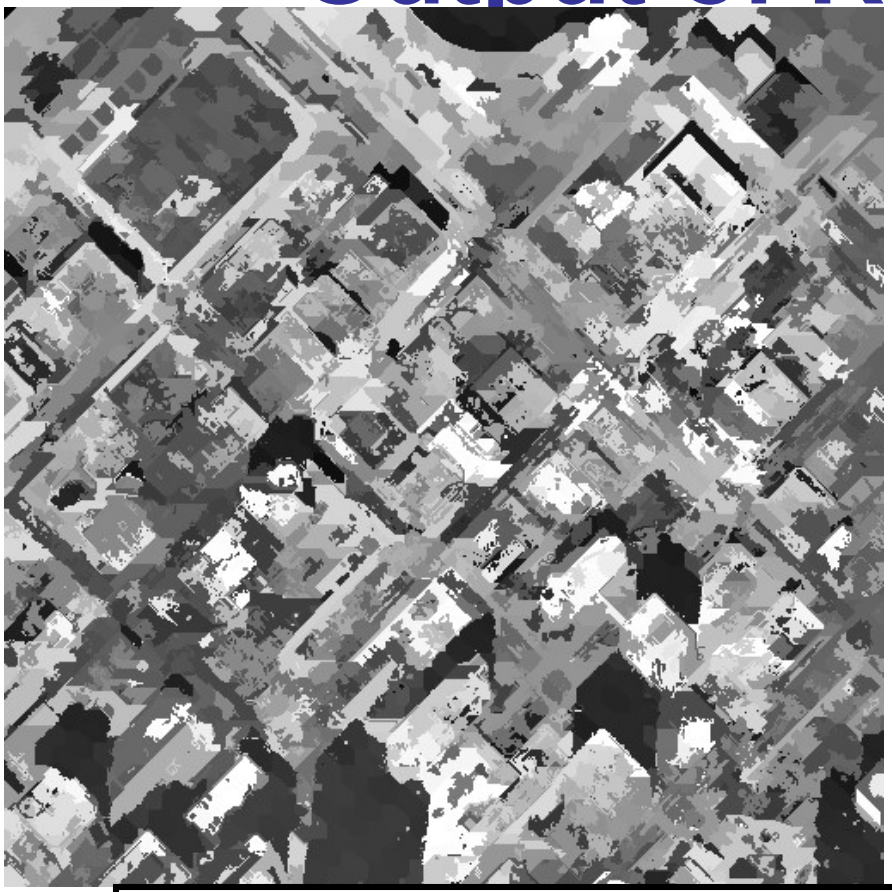
$$\delta(z) = \min_x \left[\delta(x)\right]$$

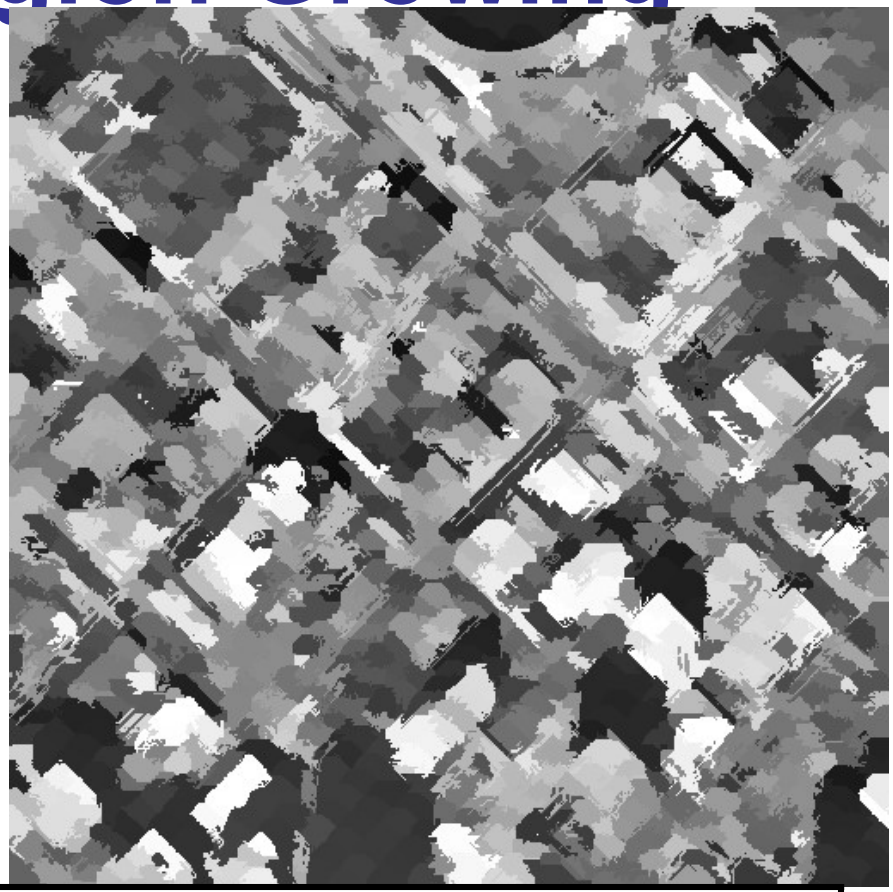
- This process refers to one iteration, that is repeated till all pixels in the image are allocated to one region or the other.

5	6	7	22	25
7	8	9	26	27
11	10	12	29	28
20	104	20	26	27
100	102	103	63	62
100	101	103	63	64
102	105	101	62	63

Algorithm for Seeded Region Growing

Output of Region Growing





From Randomly selected seeds	From Manually selected seeds

Comments

- If seed points are too many, region fragmentation may occur
- If seed points are too few, different objects can be merged within a single region
- The second case is easier to handle. One may consider each large region and grow sub-regions within it by repeating the same process performed on the full image.

Contd...