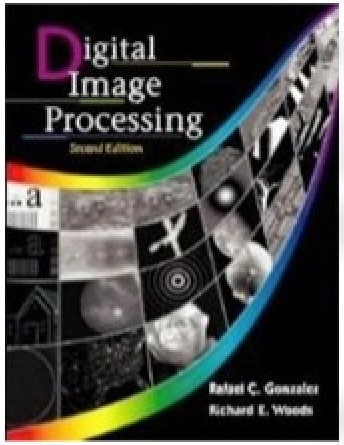
Chapter 6

Image Segmentation

REFERENCES



"Digital Image Processing", Rafael C. Gonzalez & Richard E. Woods, Addison-Wesley, 2002

Much of the material that follows is taken from this book

Slides by Brian Mac Namee
Brian.MacNamee@comp.dit.ie

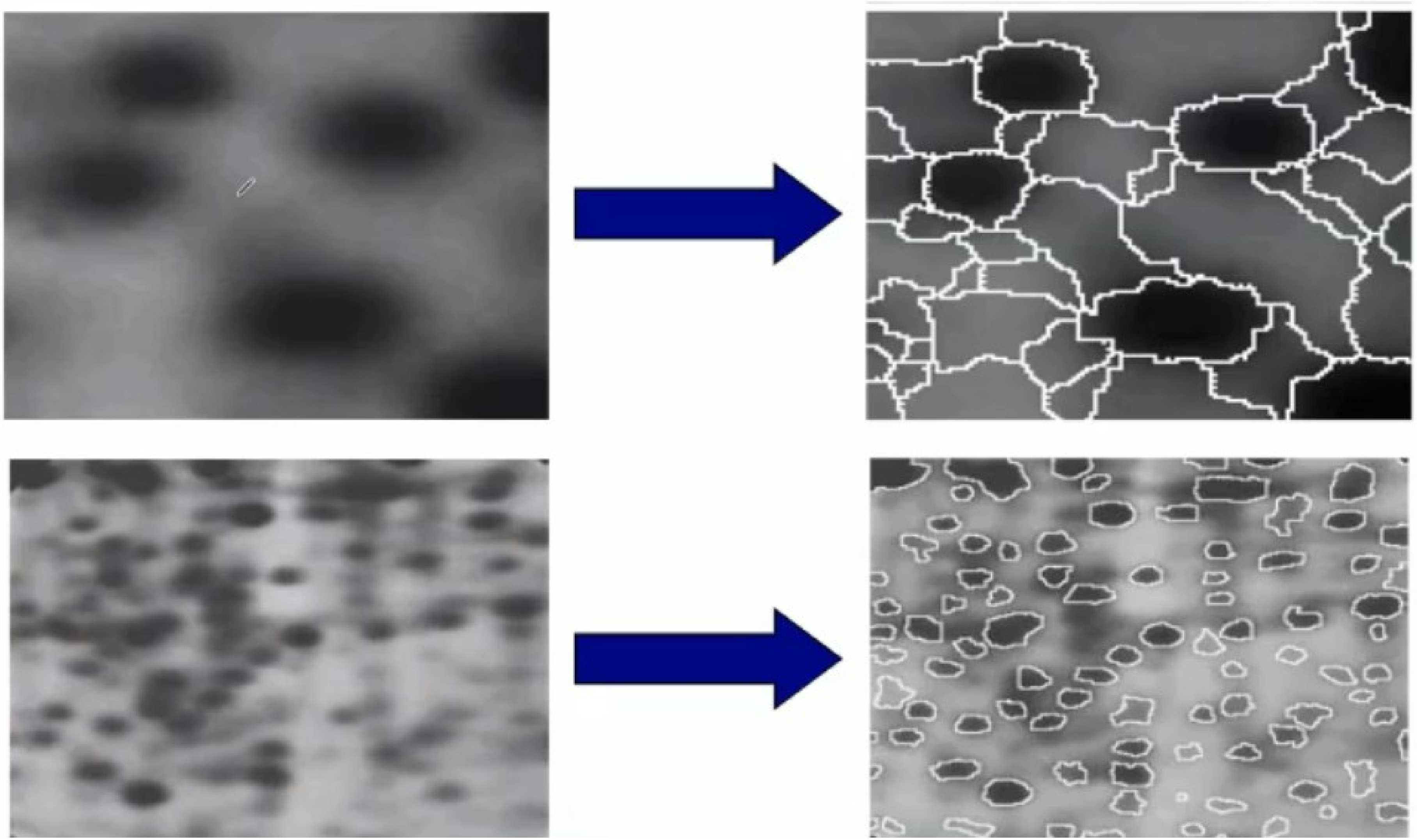
Segmentation

- Image Segmentation is the process by which a digital image is partitioned into various subgroups (of pixels) called Image Objects, which can reduce the complexity of the image, and thus analysing the image becomes simpler.
- The level to which the subdivision is carried depends on the problem being solved. That is, segmentation should stop when the objects of interest in an application have been isolated.

Segmentation

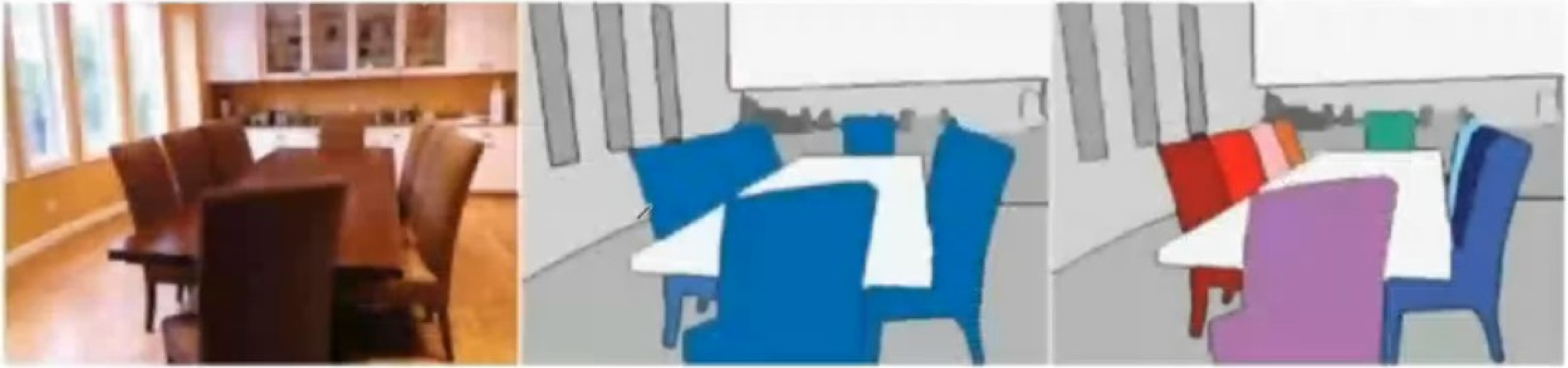
- We use various image segmentation algorithms to split and group a certain set of pixels together from the image.
- By doing so, we are actually assigning labels to pixels and the pixels with the same label fall under a category where they have some or the other thing common in them

Segmentation Examples



Segmentation

From a main image on the left, we try to get the major components, e.g. chair, table etc. and hence all the chairs are colored uniformly. In the next tab, we have detected instances, which talk about individual objects, and hence the all the chairs have different colors.



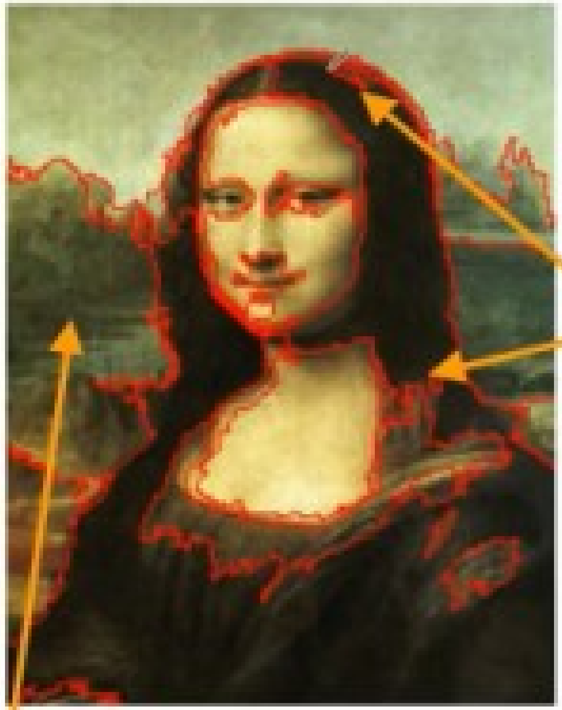
Segmentation

boundaries labels pseudocolors mean colors

Segmentation

The concept of partitioning, dividing, fetching, and then labeling and later using that information to train various ML models have indeed addressed numerous business problems

Segmentation Problems



oversegmentation
(hair should
be one group)

undersegmentation
(water should be
separated from trees)

Types of Image Segmentation

Whenever we try to take a consideration of the Image Segmentation tasks, we need to observe a crucial process that happens -- object identification.

Any simple to complex application areas, everything is based out of object detection.

Detection is made possible because the image segmentation algorithms try to collect similar pixels together and separate out dissimilar pixels. This is done by following two approaches based on the image properties:

- Similarity Detection
- Discontinuity Detection

Similarity Detection (Region Approach)

- This fundamental approach relies on detecting similar pixels in an image – based on a threshold, region growing, region spreading, and region merging.

- Machine learning algorithms like clustering relies on this approach of similarity detection on an unknown set of features, so does classification, which detects similarity based on a pre-defined (known) set of features.

Discontinuity Detection (Boundary Approach)

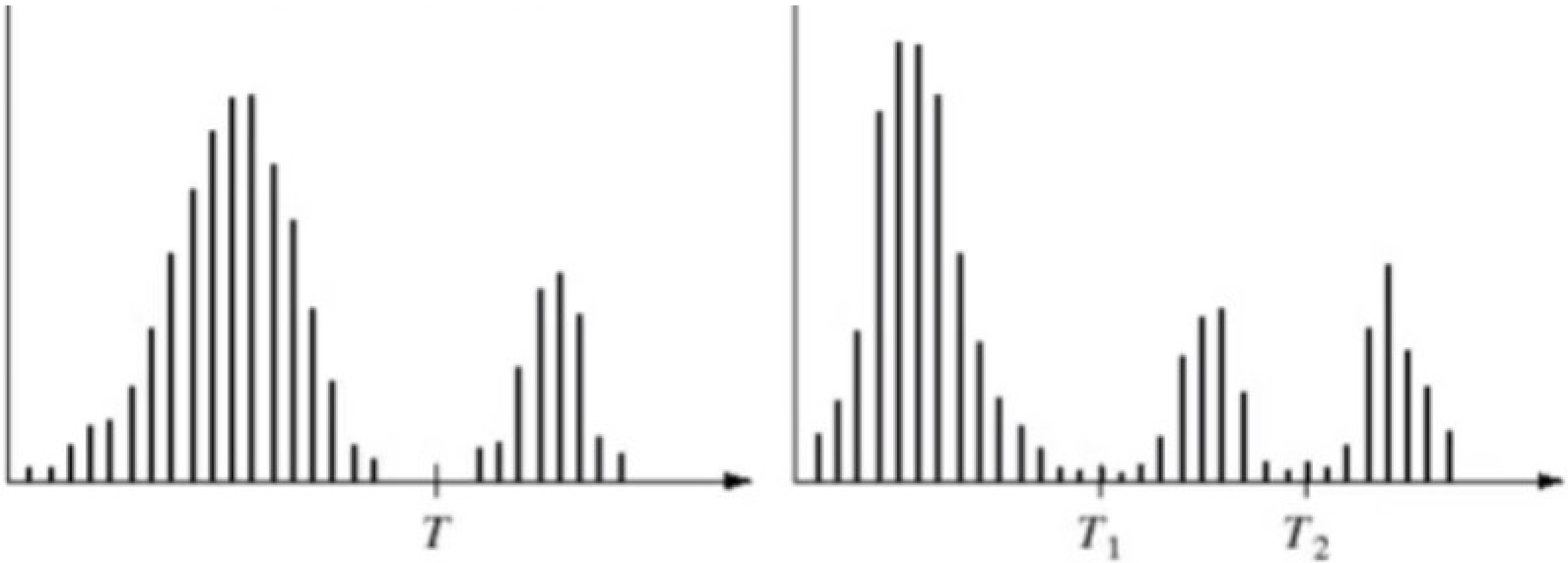
- Discontuity Detection is opposite of similarity detection approach where the algorithm rather searches for discontinuity.
- Image Segmentation Algorithms like Edge Detection, Point Detection, Line Detection follows this approach – where edges get detected based on various metrics of discontinuity like intensity etc.

Thresholding

- Thresholding is usually the first step in any segmentation approach
- We have talked about simple single value thresholding already
- Single value thresholding can be given mathematically as follows:

$$g(x,y)=\begin{cases}1 \; if \; f(x,y)>T \\ 0 \; if \; f(x,y)\leq T\end{cases}$$

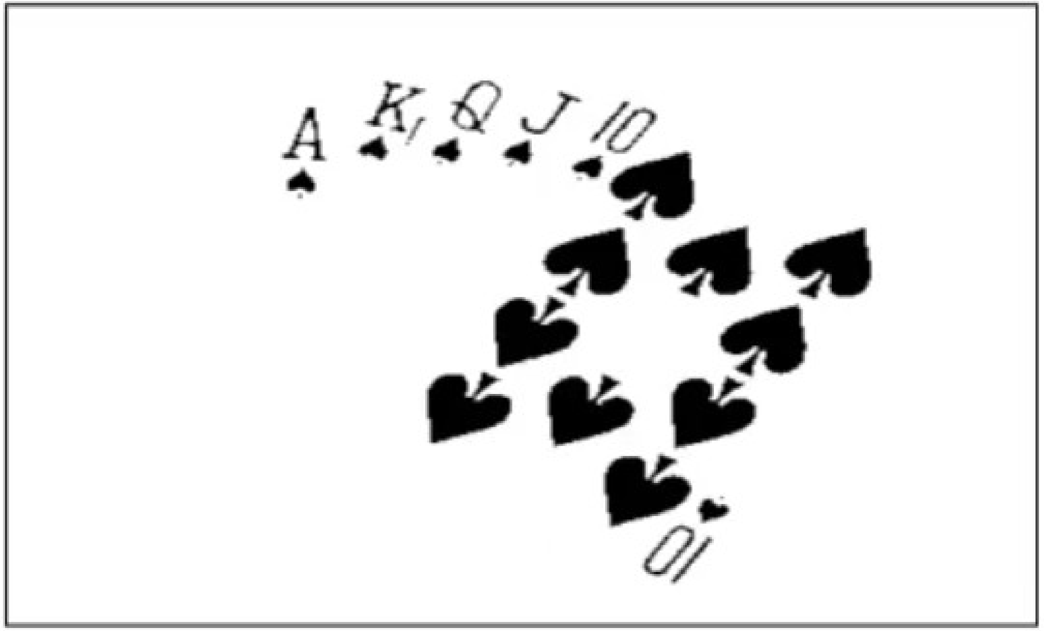
Thresholding



Thresholding Example

Imagine a poker playing robot that needs to visually interpret the cards in its hand

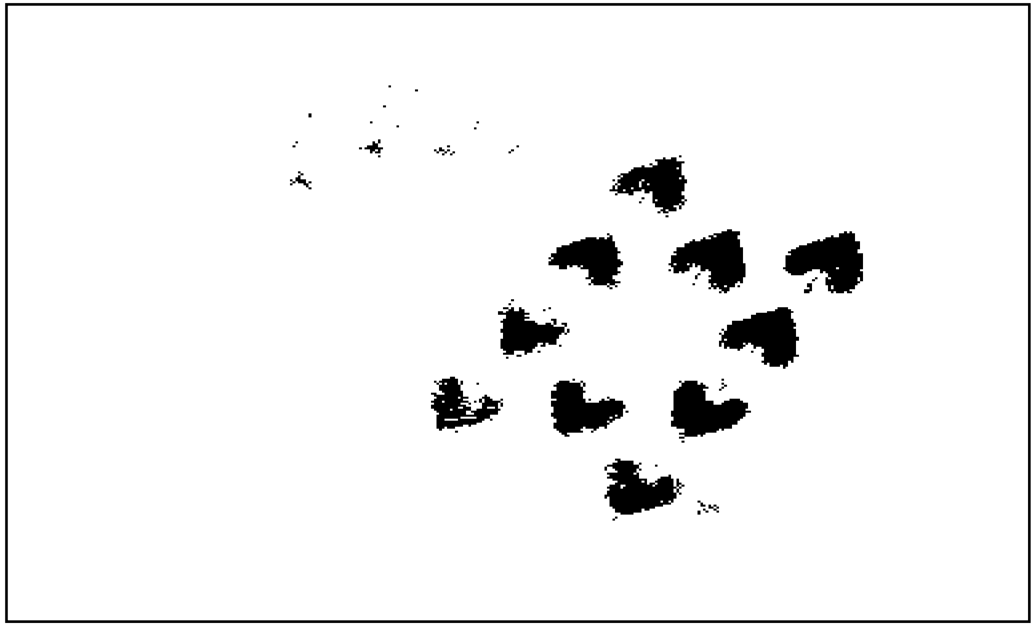
Original Image



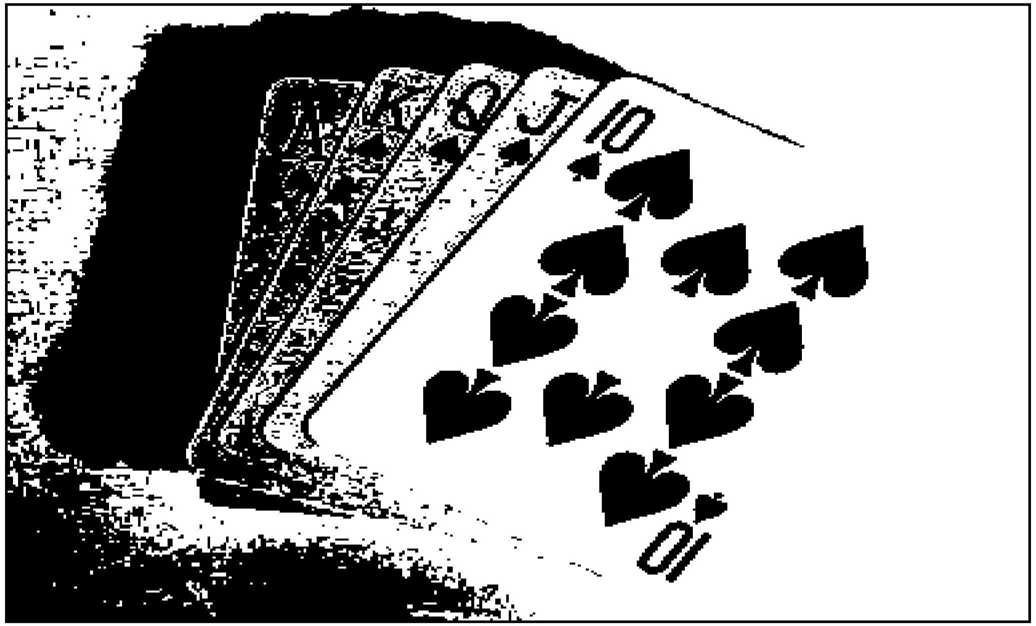
Thresholded Image

But Be Careful

If you get the threshold wrong the results can be disastrous

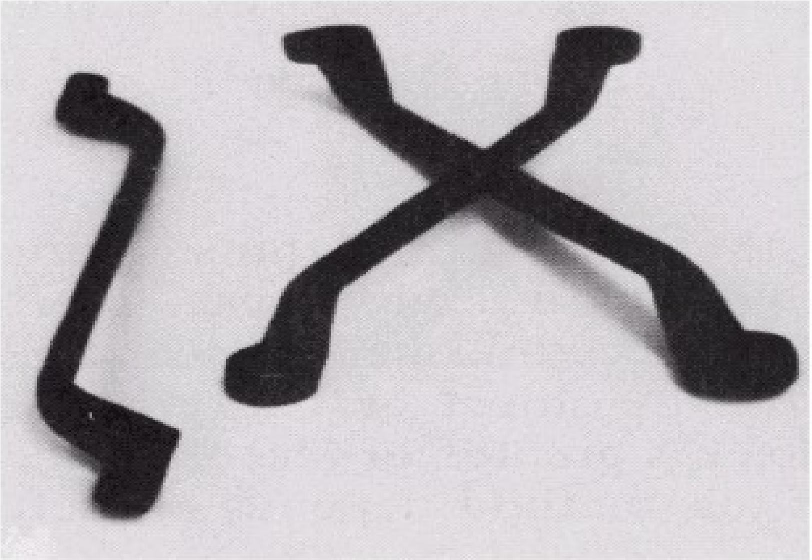


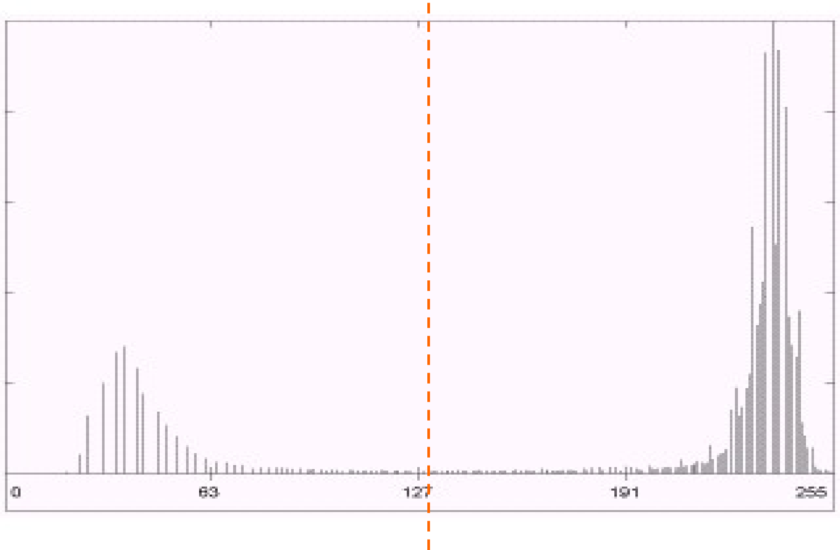
Threshold Too Low

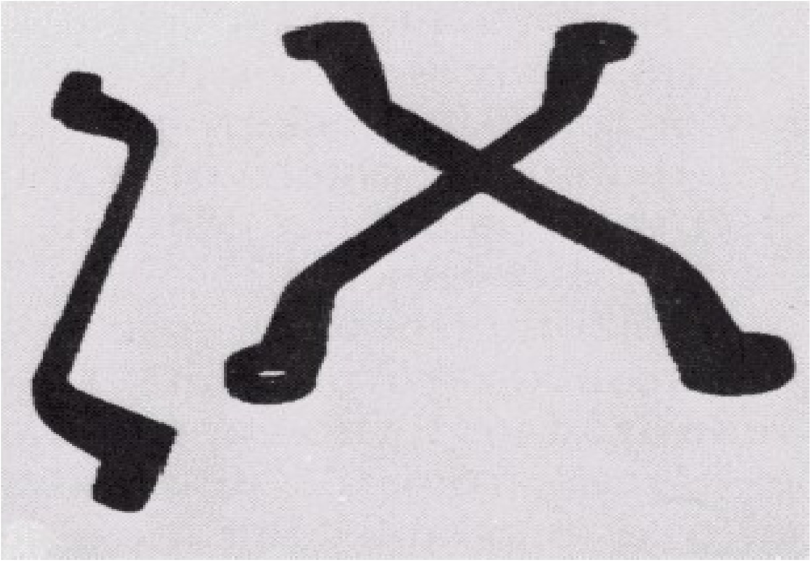


Threshold Too High

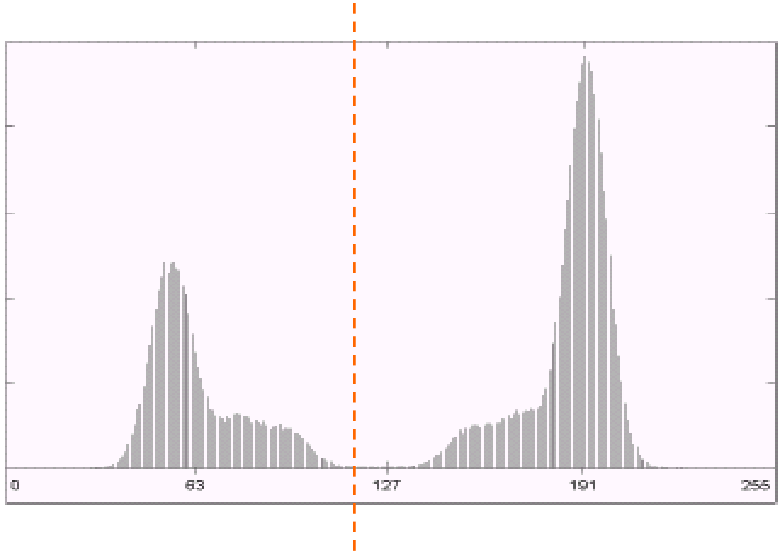
Thresholding Example 1







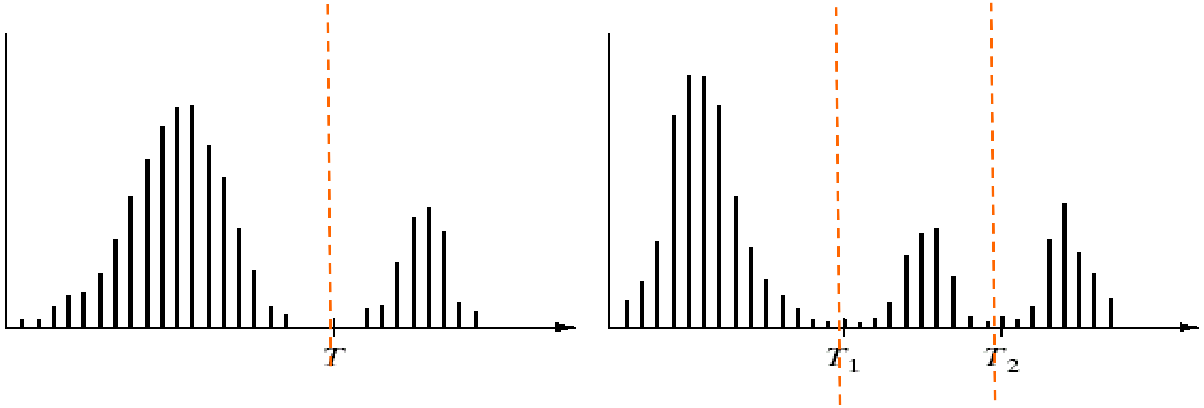
Thresholding Example 2



Problems With Single Value Thresholding

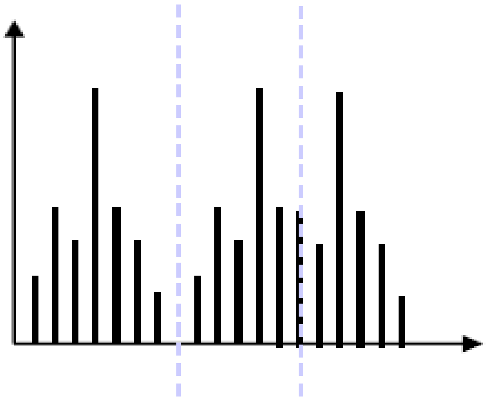
Single value thresholding only works for bimodal histograms

Images with other kinds of histograms need more than a single threshold



Thresholding

- Suppose several objects with differing gray levels (with a dark background) comprise the image
- An object may be classified as belonging to one object class if $T_1 < f(x,y) \le T_2$, to a second class if $f(x,y) > T_2$ or to the background if $f(x,y) \le T_1$
- This, however, is generally less reliable than single level thresholding



T1
T2

Thresholding

- Thresholding may be viewed as an operation that tests against a given function of the form

$$T = T[x, y, p(x, y), f(x, y)]$$

- where *f*(*x*,*y*) is the gray level of point (*x*,*y*) and *p*(*x*,*y*) is some local property of the point -- the average gray level of a neighborhood around (*x*,*y*)
- The thresholded image is given by

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

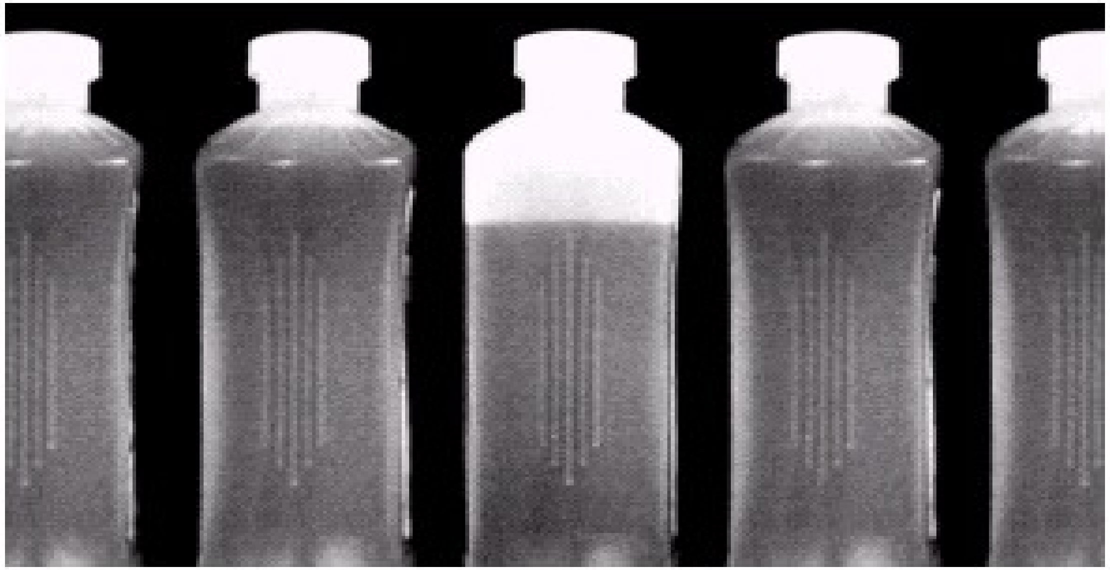
- Pixels labeled 1 (or any other convenient gray level value) correspond to objects

Thresholding

- When T depends only on $f(x,y)$ the threshold is called *global*
- If T depends on $f(x,y)$ and $p(x,y)$ the threshold is *local*
- If, in addition, T depends on the spatial coordinates (x,y), the threshold is called *dynamic*
- For example, a local threshold may be used if certain information about the nature of the objects in the image is known a priori
- A dynamic threshold may be used in the case where object illumination is non-uniform

Problems With Single Value Thresholding

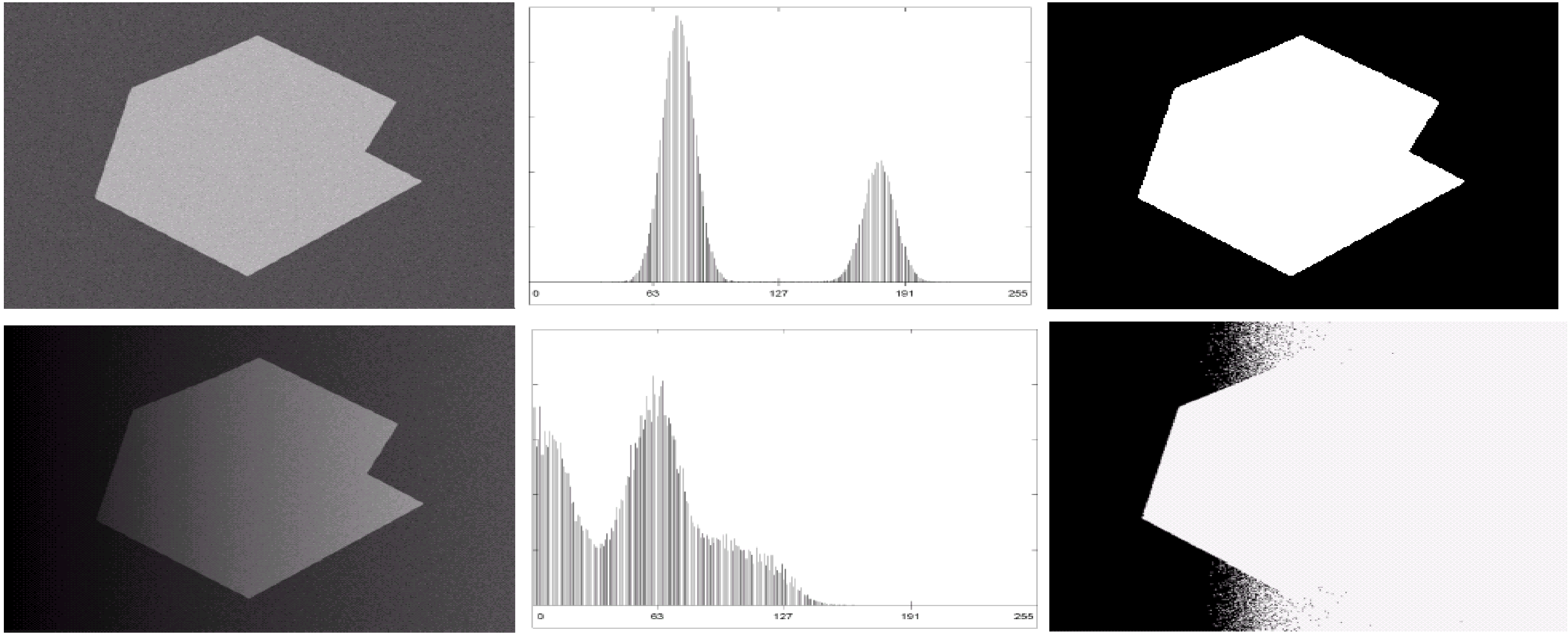
Let’s say we want to isolate the contents of the bottles



Think about what the histogram for this image would look like

What would happen if we used a single threshold value?

Single Value Thresholding and Illumination



Uneven illumination can really upset a single valued thresholding scheme

Basic Global Thresholding

- Based on the histogram of an image
- Partition the image histogram using a single global threshold
- The success of this technique very strongly depends on how well the histogram can be partitioned

Basic Global Thresholding Algorithm

The basic global threshold, T, is calculated as follows:

1. Select an initial estimate for T (typically the average grey level in the image)
2. Segment the image using T to produce two groups of pixels: G_1 consisting of pixels with grey levels >T and G_2 consisting pixels with grey levels $\leq$ T
3. Compute the average grey levels of pixels in G_1 to give μ_1 and G_2 to give μ_2

Basic Global Thresholding Algorithm

4. Compute a new threshold value:

$$T = \frac{\mu_1 + \mu_2}{2}$$

5. Repeat steps 2 – 4 until the difference in T in successive iterations is less than a predefined limit T

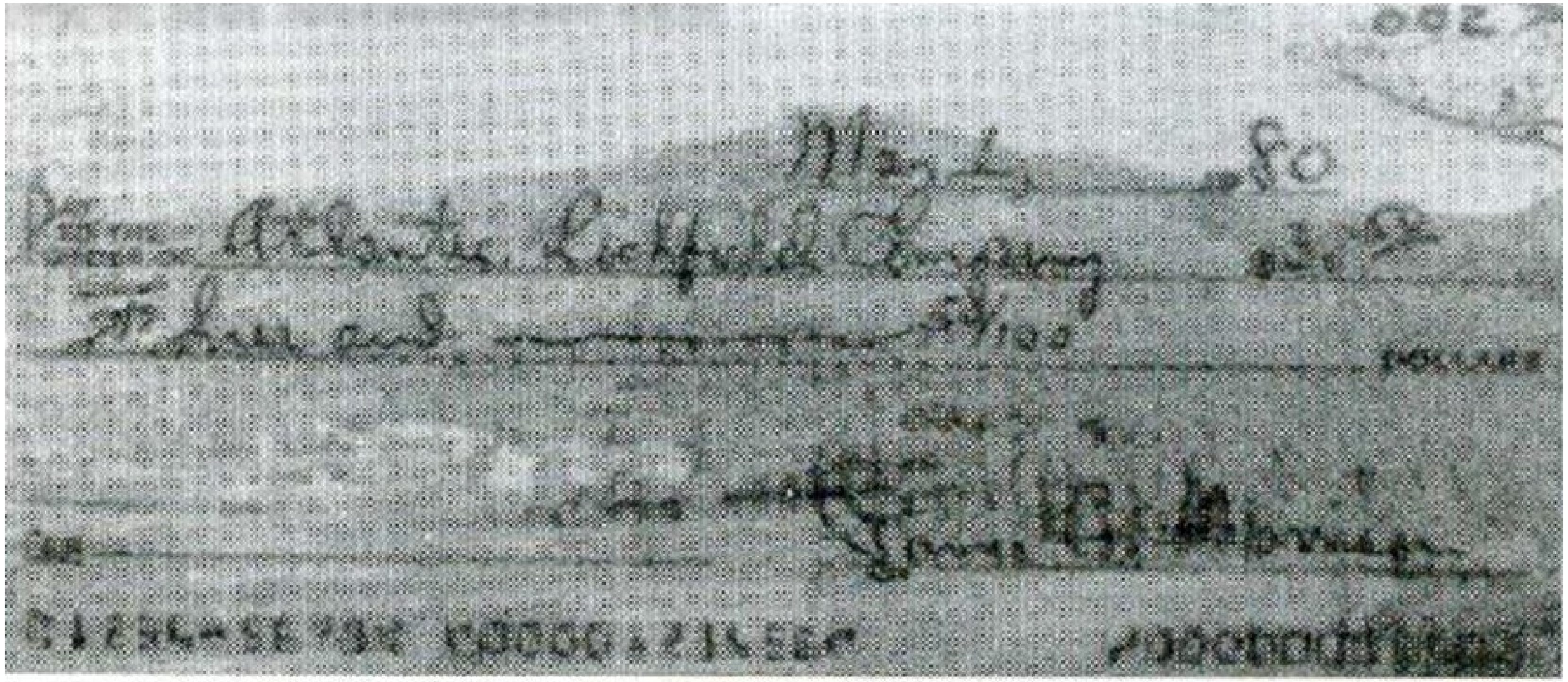
This algorithm works very well for finding thresholds when the histogram is suitable

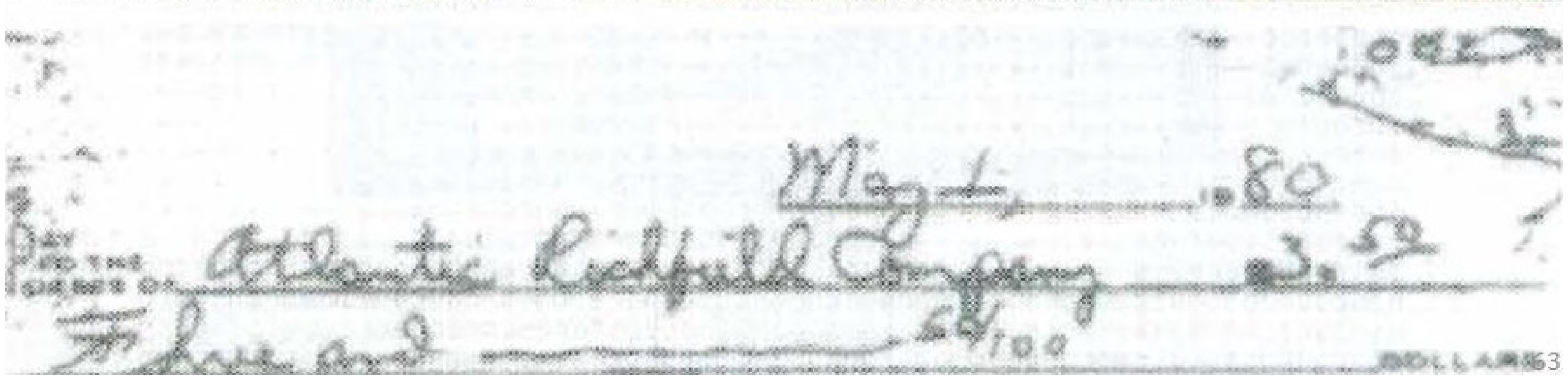
Basic Adaptive Thresholding

An approach to handling situations in which single value thresholding will not work is to divide an image into sub images and threshold these individually

Since the threshold for each pixel depends on its location within an image this technique is said to *adaptive*

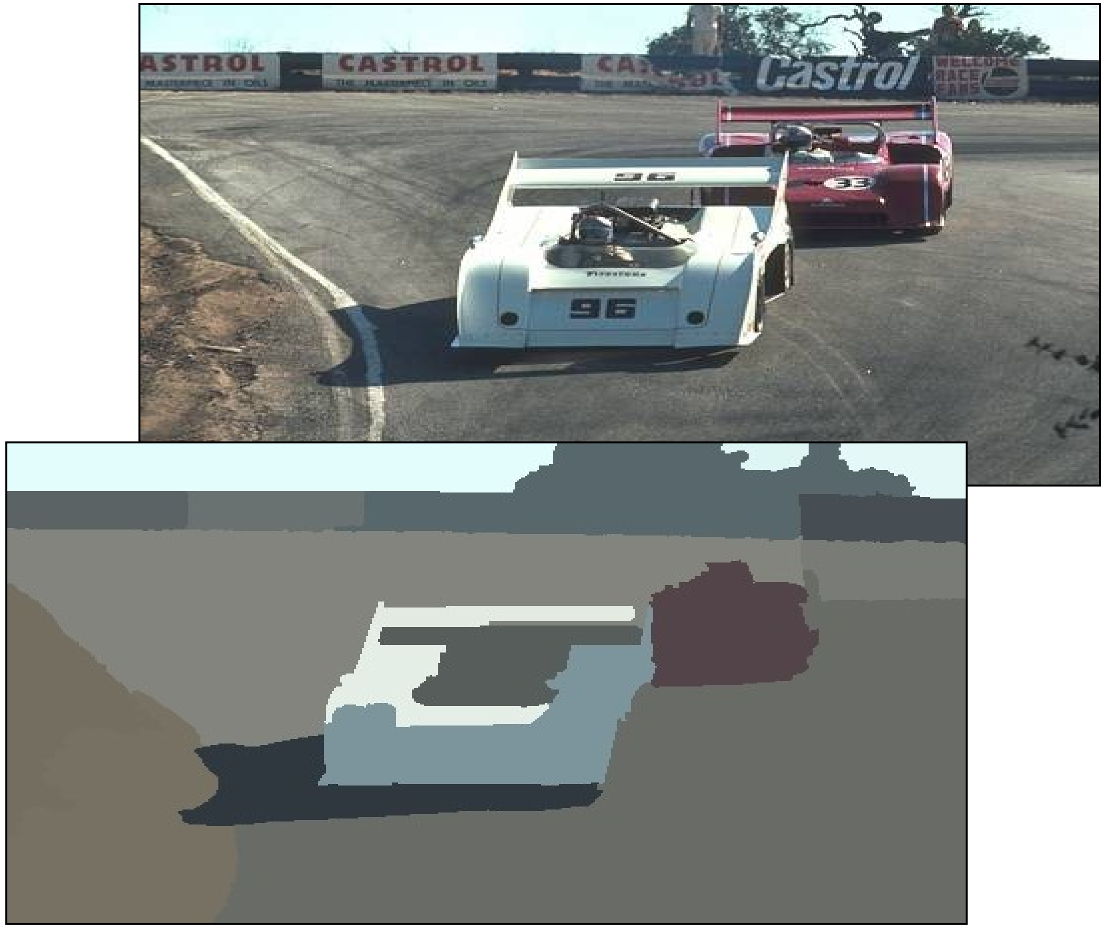
Thresholding Based on Boundary





Why Region-Based Segmentation?

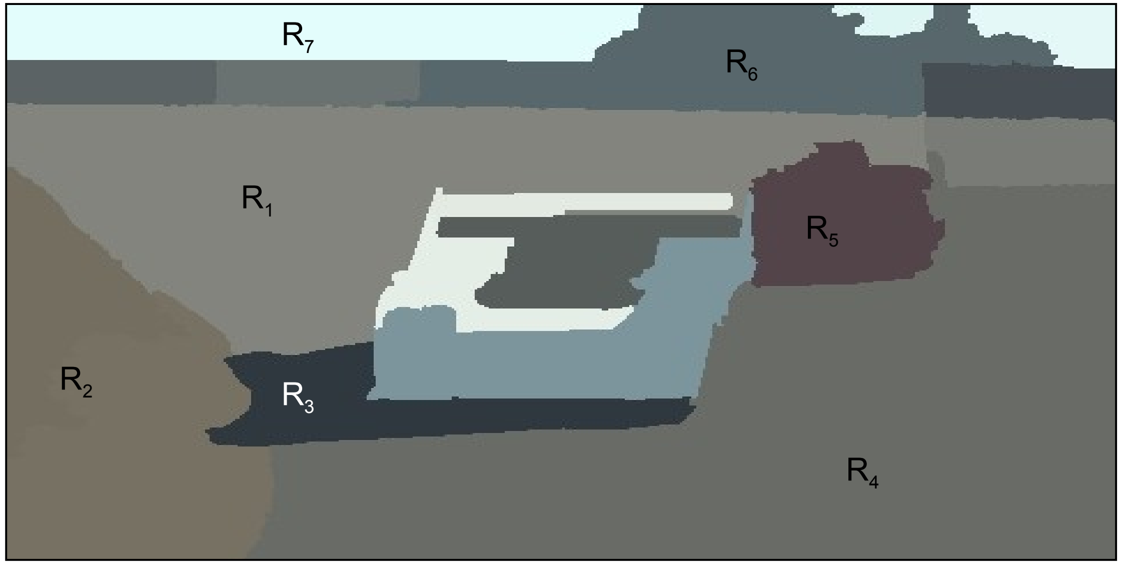
- Segmentation
  - Edge detection and Thresholding not always effective.
- Homogenous regions
  - *Region-based segmentation.*
  - Effective in noisy images.



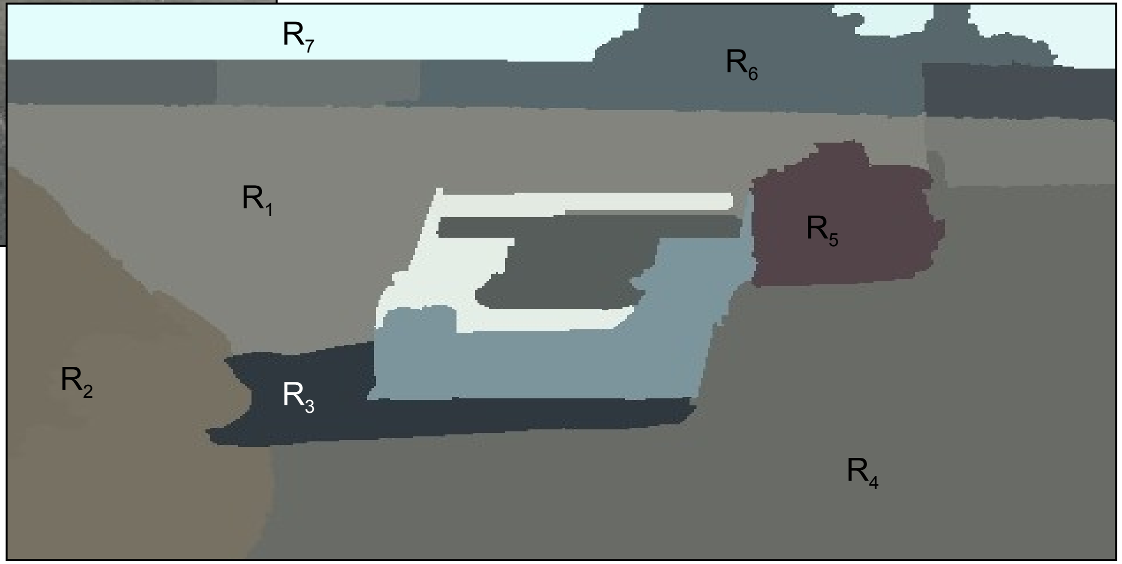
Definations

- Based on *sets*.
- Each image R is a set of regions R_i.
  - Every pixel belongs to one region.
  - One pixel can only belong to a single region.

$$R = \bigcup_{i=1}^{S} R_i \qquad R_i \text{ intersect } R_j = \emptyset$$



Definations



Basic Formulation

Let R represent the entire image region.

Segmentation partitions R into n subregions, R_1, R_2, ..., R_n, such that:

a) $\mathop{}_{i=1}^{n} R_i = R$

b) R_i is a connected region, $i = 1, 2, \ldots, n$.

c) $R_i \cap R_j = \varphi$ for all i and j, $i \neq j$

d) $P(R_i) = \text{TRUE}$ for $i = 1, 2, \ldots, n$.

e) $P(R_i \cup R_j) = \text{FALSE}$ for $i \neq j$.

a) Every pixel must be in a region

b) Points in a region must be connected.

c) Regions must be disjoint.

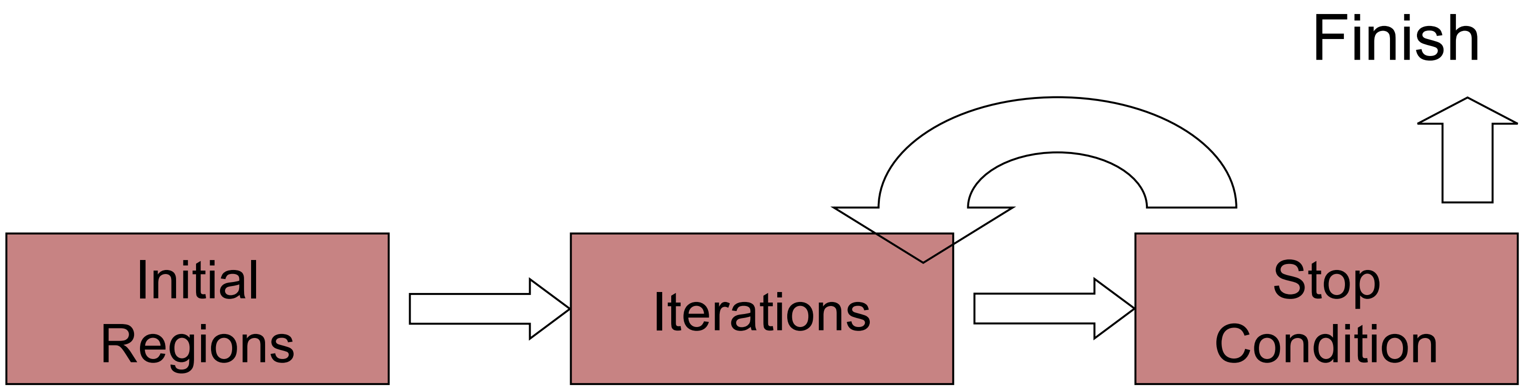
d) All pixels in a region satisfy specific properties.

e) Different regions have different properties.

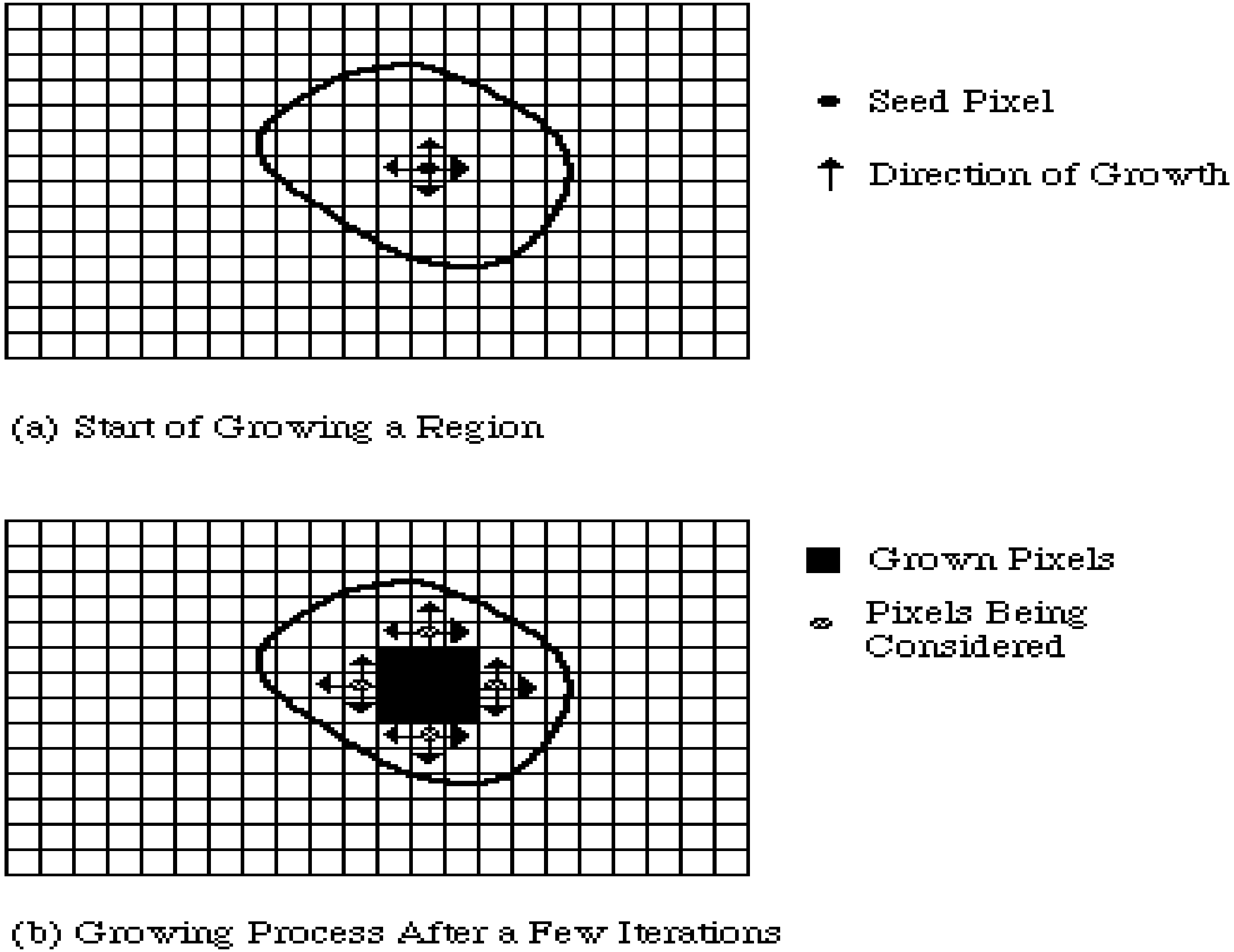
Basic Formulation

- Groups pixels into larger regions.
- Starts with a **seed** region.
- **Grows** region by **merging** neighboring pixels.

- Iterative process
  - How to start?
  - How to iterate?
  - When to stop?



Region Growing



Region Merging

- Algorithm
  - Divide image into an initial set of regions.
    - One region per pixel.
  - Define a **similarity criteria** for merging regions.
  - **Merge** similar regions.
  - Repeat previous step until no more merge operations are possible.

Similarity Criteria

- Homogeneity of regions is used as the main segmentation criterion in region growing.
  - gray level
  - color, texture
  - shape
  - model
  - etc.

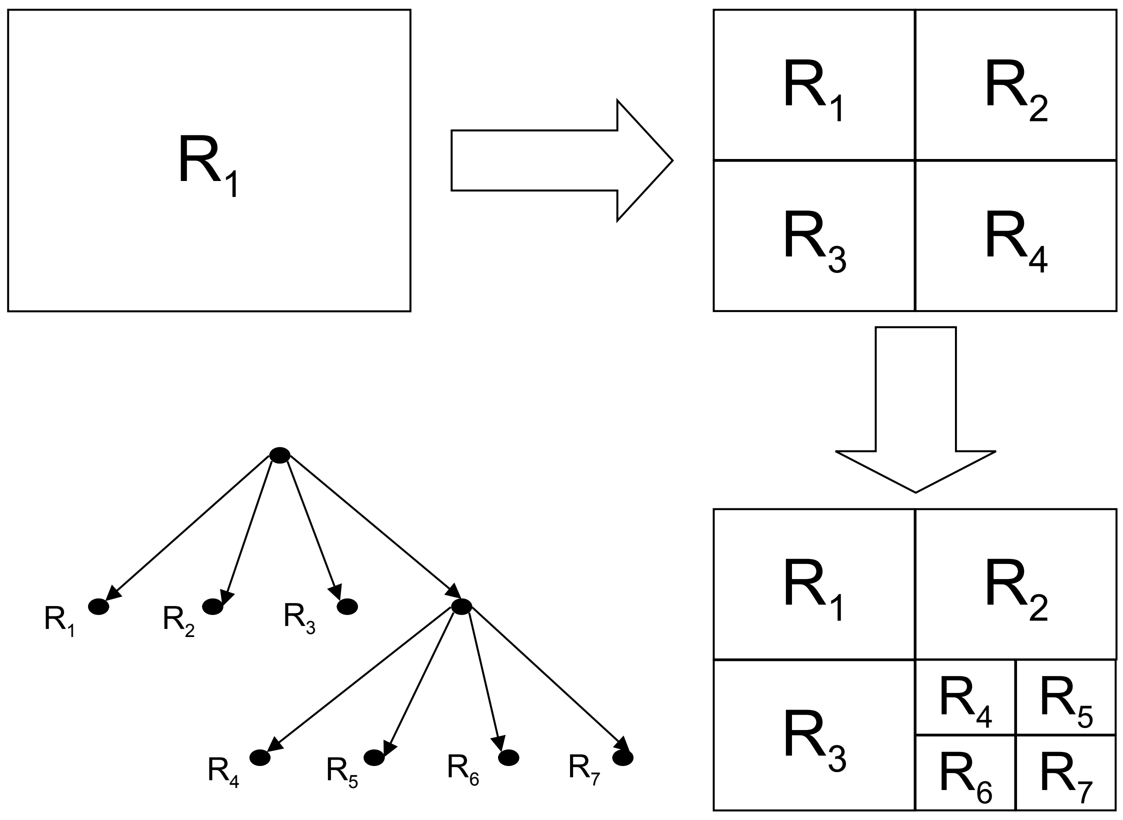
Choice of criteria affects segmentation results dramatically!

Gray-Level Criteria

- Comparing to Original Seed Pixel
  - Very sensitive to choice of **seed point**.
- Comparing to Neighbor in Region
  - Allows gradual changes in the region.
  - Can cause significant drift.
- Comparing to Region Statistics
  - Acts as a **drift dampener**.
- Other possibilities!

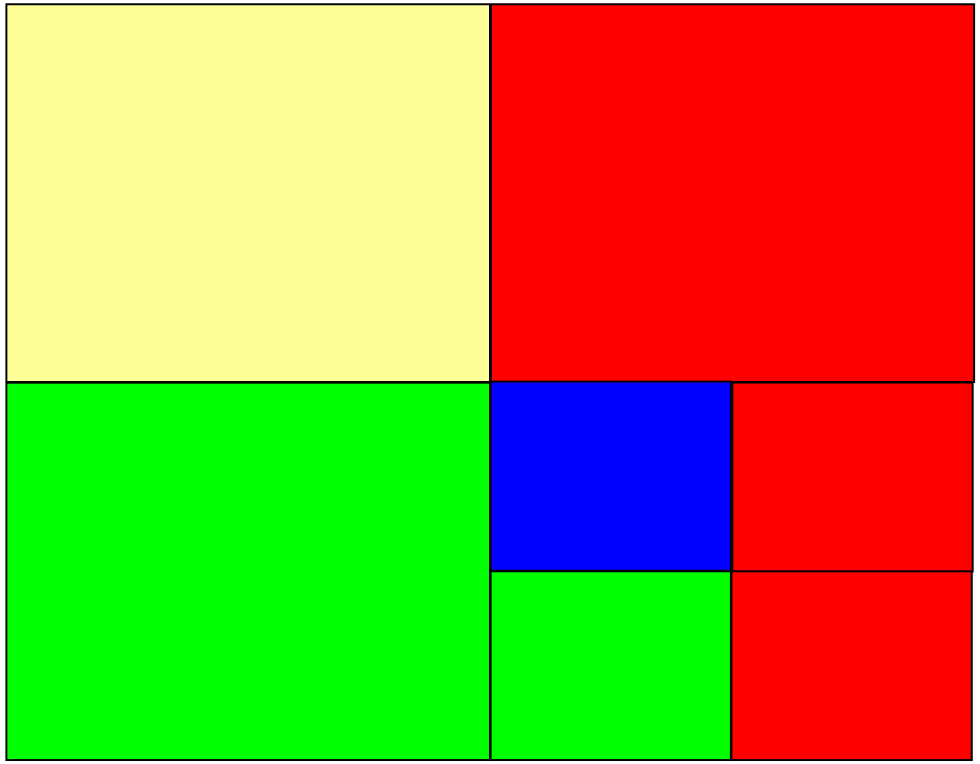
Region Splitting

- Algorithm
  - One initial set that includes the **whole image**.
  - **Similarity criteria**.
  - Iteratively **split** regions into sub-regions.
  - Stop when no more splittings are possible.



Splitting and Merge

- Combination of both algorithms.
- Can handle a larger variety of shapes.
  - Simply apply previous algorithms consecutively.



Detection Of Discontinuities

There are three basic types of grey level discontinuities that we tend to look for in digital images:

- Points
- Lines
- Edges

We typically find discontinuities using masks and correlation

Detection Of Discontinuities

- Detecting discontinuities (points, lines and edges) is generally accomplished by mask processing (much as in the spatial domain filter examples)
- Use the response equation

$$R = w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9$$
$$= \sum_{i=1}^{9} w_i z_i$$

- A mask used for detecting isolated points (different from a constant background would be

-1	-1	-1
-1	8	-1
-1	-1	-1

Detection Of Discontinuities

- Detection of isolated points is accomplished by using the previous mask
- An isolated point is detected if the response of the mask is greater that a predetermined threshold

$$|R| > T$$

- This measures the weighted difference between a center point and its neighbors
- The mask is the same as the high frequency filtering mask
- The emphasis here is on the detection of points
  - Only differences that are large enough to be considered isolated points in an image are of interest

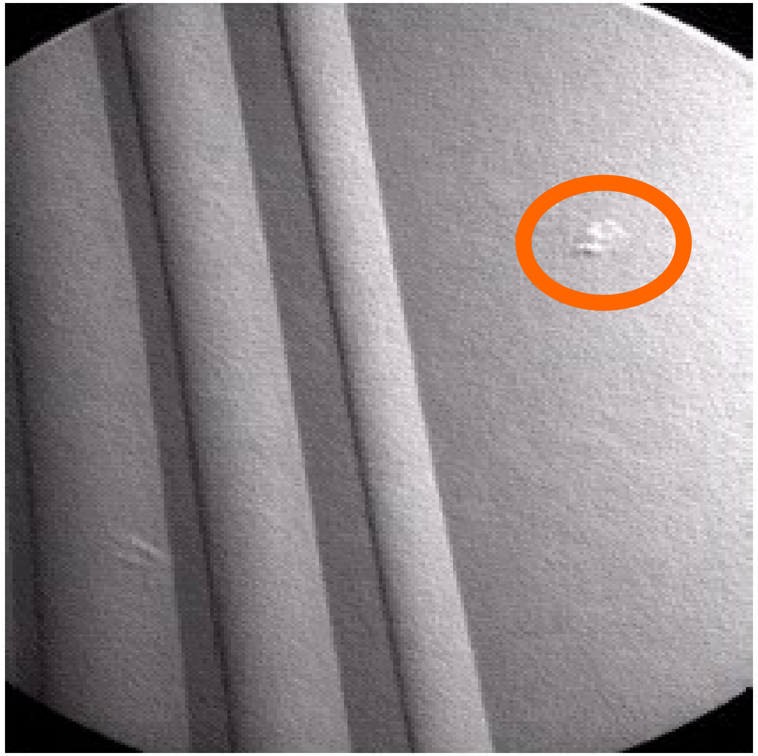
Point Detection

Point detection can be achieved simply using the mask below:

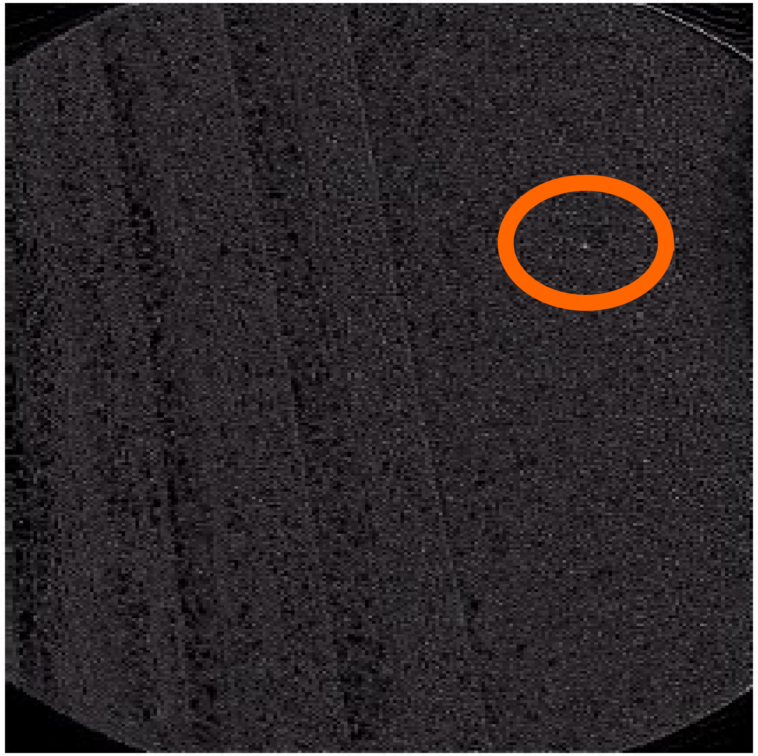
-1	-1	-1
-1	8	-1
-1	-1	-1

Points are detected at those pixels in the subsequent filtered image that are above a set threshold

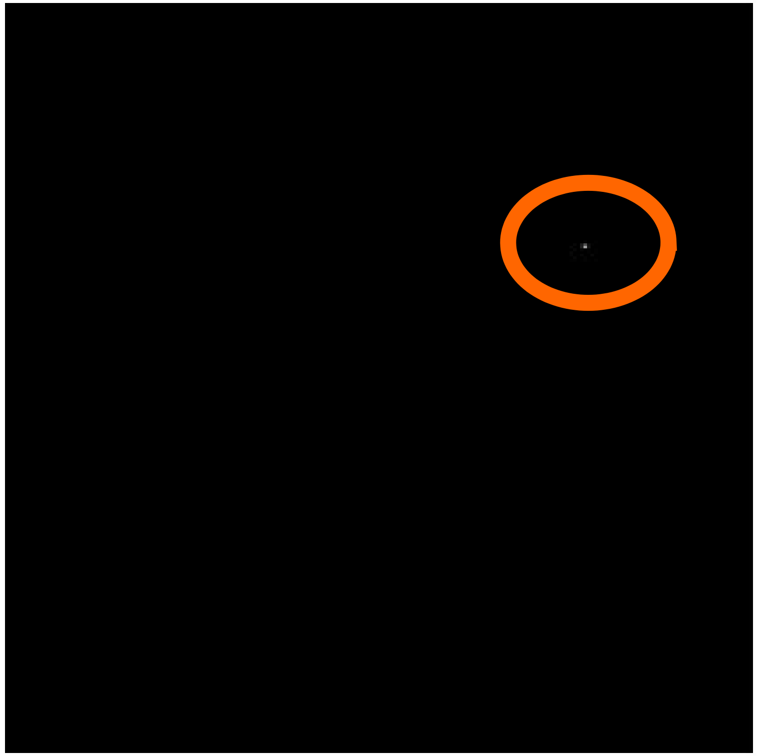
Point Detection



X-ray image of a turbine blade



Result of point detection



Result of thresholding

Line Detection

The next level of complexity is to try to detect lines

The masks below will extract lines that are one pixel thick and running in a particular direction

−1	−1	−1
2	2	2
−1	−1	−1

Horizontal

−1	−1	2
−1	2	−1
2	−1	−1

+45°

−1	2	−1
−1	2	−1
−1	2	−1

Vertical

2	−1	−1
−1	2	−1
−1	−1	2

−45°

Line Detection

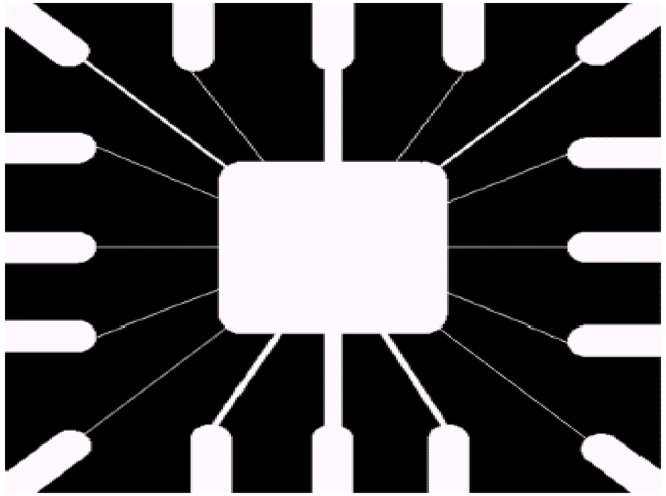
- With a constant background, the maximum response occurs when the line is “lined up” with the center of the mask
- Note that the preferred direction of each mask is weighted with a larger coefficient than other possible directions
- Let R_1, R_2, R_3 and R_4 denote the responses of the masks
- If, at a certain point in the image,

$$|R_i| > |R_j| \text{ for all } j \neq i$$

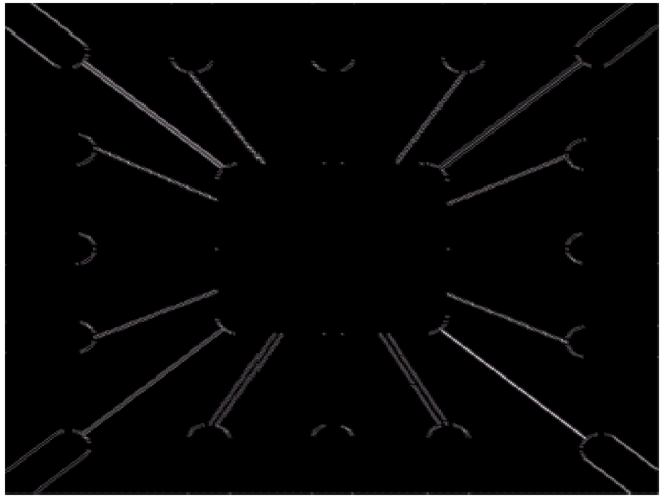
- that point is said to be more likely associated with a line in the direction of mask *i*

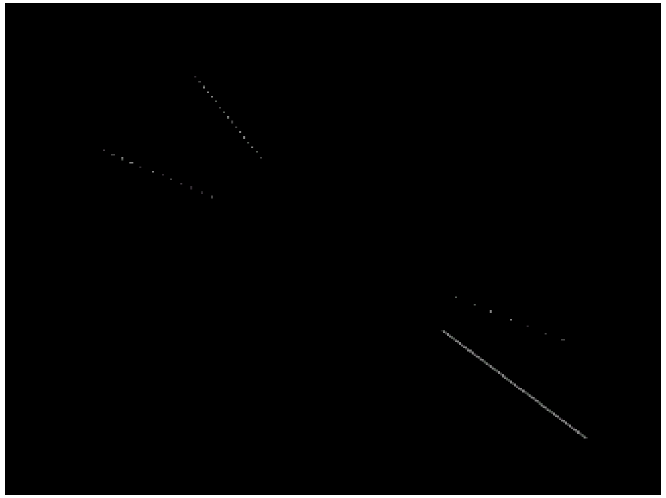
Line Detection

Binary image of a wire bond mask



After processing with -45° line detector





Result of thresholding filtering result

Edge

Edges are significant local changes of intensity in a digital image.

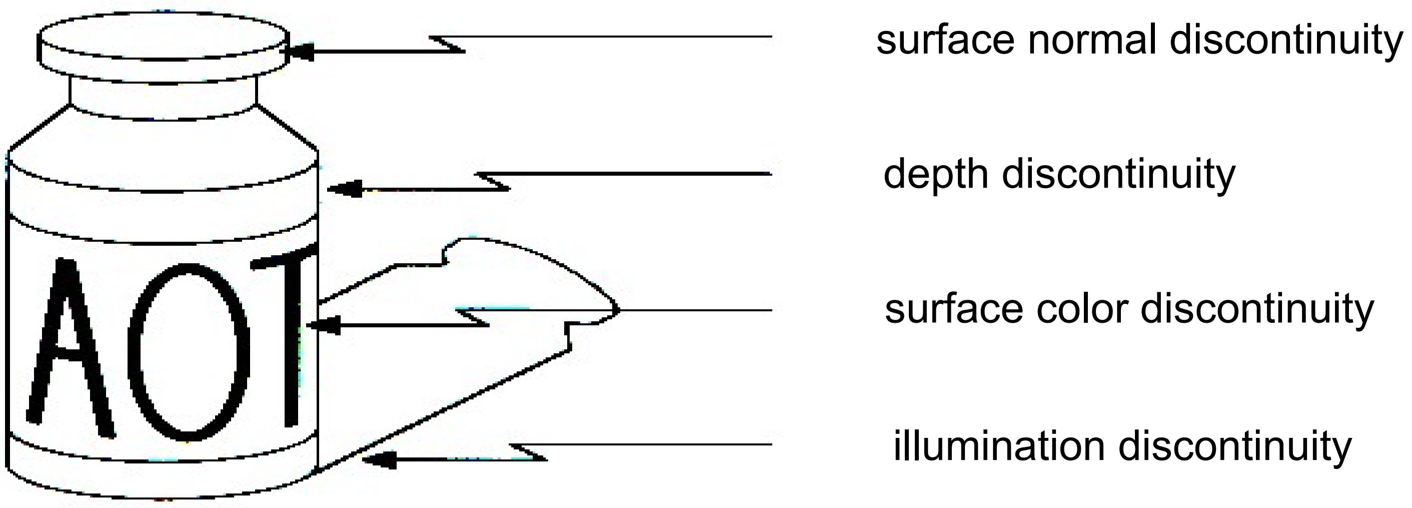
An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions. There are three types of edges:

- Horizontal edges
- Vertical edges
- Diagonal edges

Edge Detection

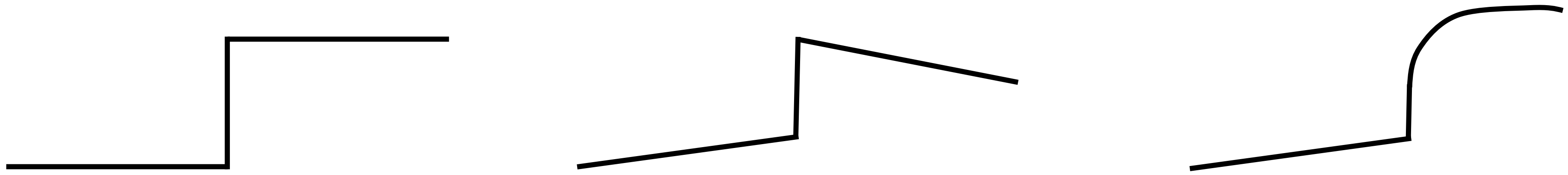
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Origin of Edges

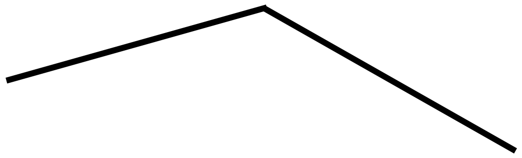


- Edges are caused by a variety of factors

Edge Types



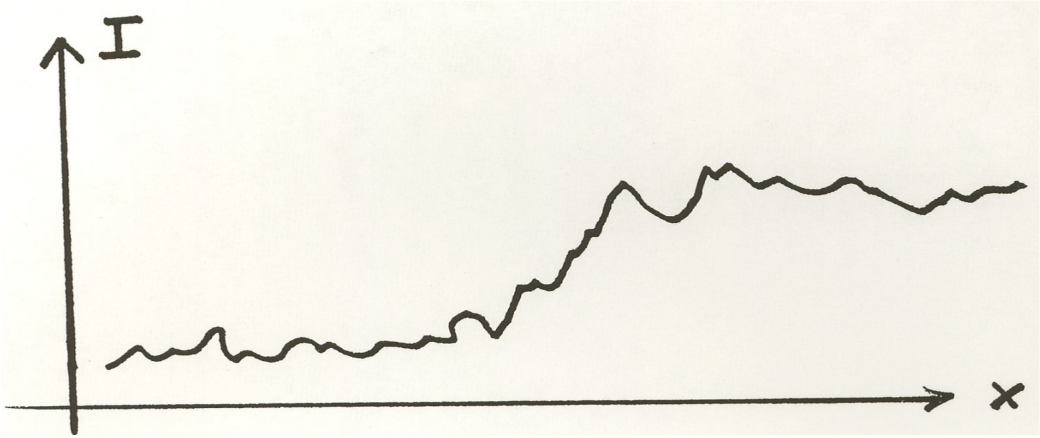
Step Edges



Roof Edge

Line Edges

Real Edges



Noisy and Discrete!

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

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

Edge Detection

Edge Detection is a method of segmenting an image into regions of discontinuity.

It is a widely used technique in digital image processing like

- pattern recognition
- image morphology
- feature extraction

Edge Detection

Edge detection is a fundamental tool in image processing, machine vision and computer vision, particularly in the areas of feature detection and feature extraction.

The purpose of detecting sharp changes in image brightness is to capture important changes in the image.

Image filters are commonly used for edge detection.

Edge Detection

The process of edge detection consists of three main steps:

- Noise reduction
- Detection of edge points
- Edge localization

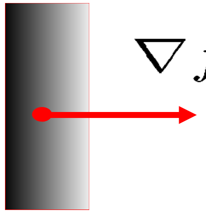
Edge Detection

- First-order derivatives generally produce thicker edges in image
- Second-order derivatives have a stronger response to fine detail, such as thin lines, isolated points, and noise

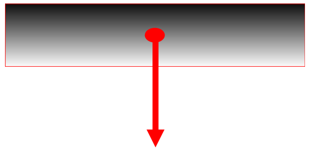
Gradient

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

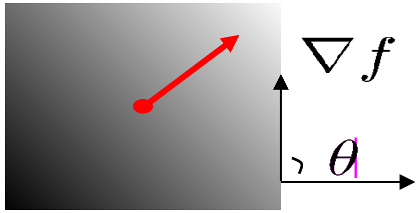
- Represents direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

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

- 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 Operator

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

  - Finite difference approximations:

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

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

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

  - Convolution masks :

$\frac{\partial I}{\partial x}$

-1	1
-1	1

$\frac{\partial I}{\partial y}$

1	1
-1	-1

Discrete Edge Operator

- Second order partial derivatives:

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

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

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

- **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

1	4	1
4	−20	4
1	4	1

Image Edge Detection Operators in Digital Image Processing

Edge Detection Operators are of two types:

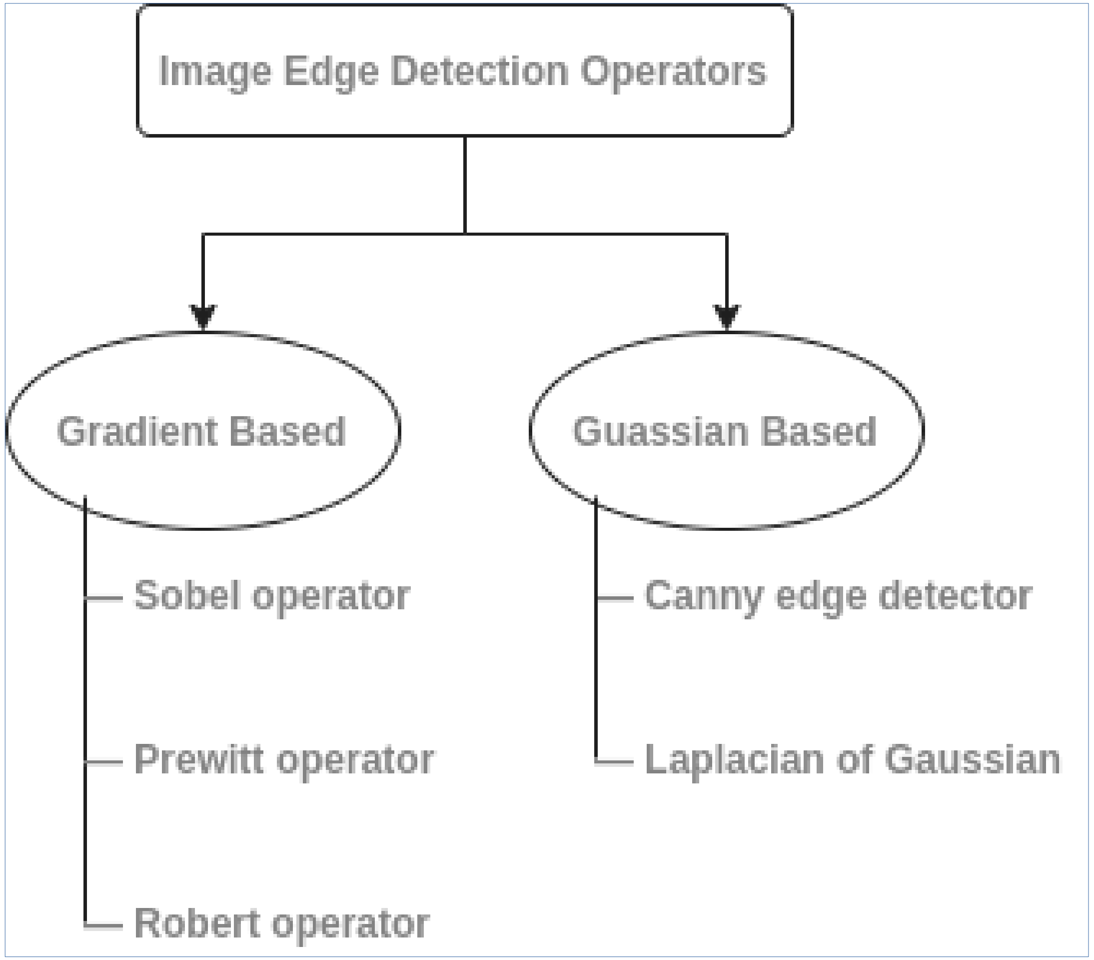
1. Gradient based operator

   - computes first-order derivations in a digital image like Sobel operator, Prewitt operator, Robert operator

2. Gaussian based operator

   - computes second-order derivations in a digital image like, Canny edge detector, Laplacian of Gaussian

Image Edge Detection Operators in Digital Image Processing



Sobel Operator

It is a discrete differentiation operator.

It computes the gradient approximation of image intensity function for image edge detection.

At the pixels of an image, the Sobel operator produces either the normal to a vector or the corresponding gradient vector.

Sobel Operator

It uses two 3 x 3 kernels or masks which are convolved with the input image to calculate the vertical and horizontal derivative approximations respectively

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Sobel Operator

Advantages:

- Simple and time efficient computation
- Very easy at searching for smooth edges

Limitations:

- Diagonal direction points are not preserved always
- Highly sensitive to noise
- Not very accurate in edge detection
- Detect with thick and rough edges does not give appropriate results

Prewitt Operator

- This operator is almost similar to the sobel operator.
- It also detects vertical and horizontal edges of an image.
- It is one of the best ways to detect the orientation and magnitude of an image. It uses the kernels or masks

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Prewitt Operator

Advantages:

- Good performance on detecting vertical and horizontal edges
- Best operator to detect the orientation of an image

Limitations:

- The magnitude of coefficient is fixed and cannot be changed
- Diagonal direction points are not preserved always

Robert Operator

This gradient-based operator computes the sum of squares of the differences between diagonally adjacent pixels in an image through discrete differentiation.

Then the gradient approximation is made. It uses the following 2 x 2 kernels or masks

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Robert Operator

Advantages

- Detection of edges and orientation are very easy
- Diagonal direction points are preserved

Limitations

- Very sensitive to noise
- Not very accurate in edge detection

Comparision of 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

Edge Detection Example

Laplace Operator

The Laplacian

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

$$= f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1)$$

$$-4f(x,y)$$

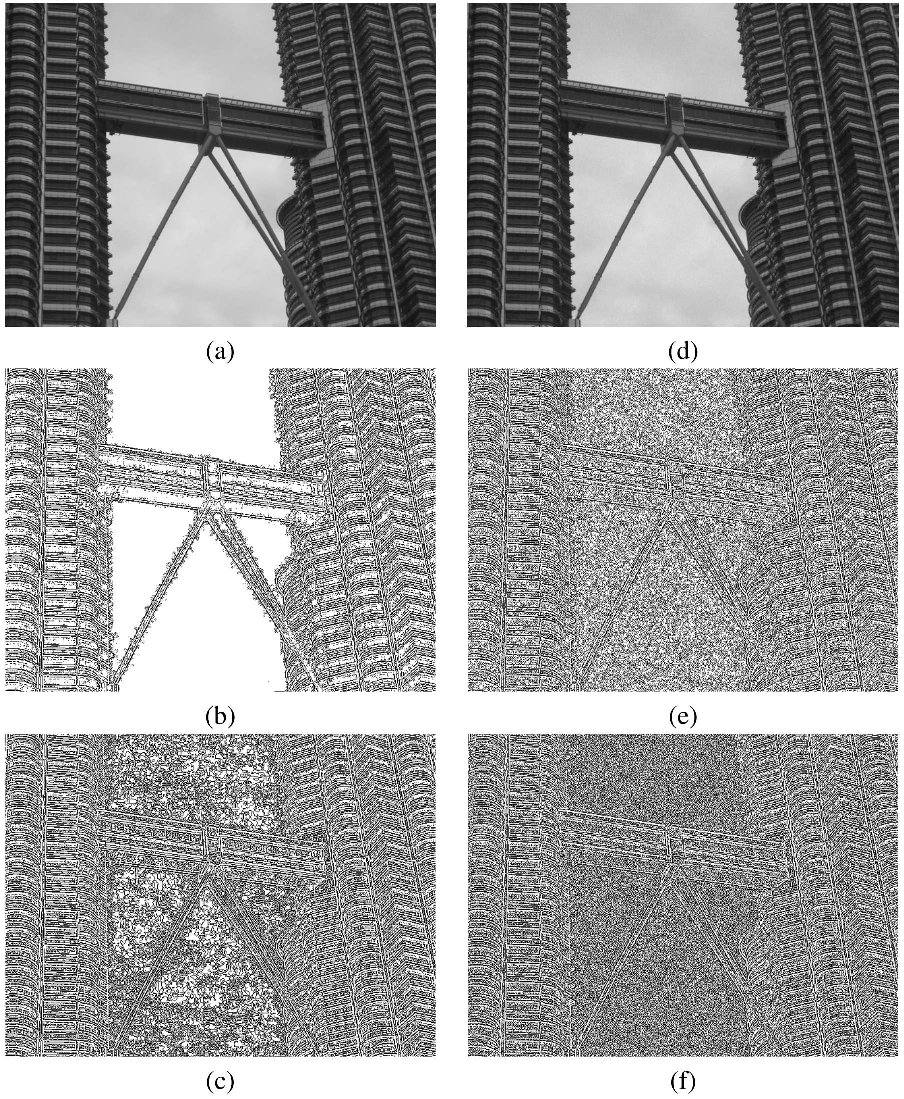
Problems with Laplacian:

- It generates “double edges”, i.e., positive and negative values for each edge.
- It is extremely sensitive to noise.

Laplacian and Zero Cross

Laplacian operator seeks out points in the signal stream where the digital signal of an image passes through a pre-set '0' value, and marks this out as a potential edge point.

Because the signal has crossed through the point of zero, it is called a zero-crossing



Laplacian of Gaussian

- Works by smoothing the image with a Gaussian low-pass filter, and then applying a Laplacian edge detector to the result.
- The LoG filter can sometimes be approximated by taking the differences of two Gaussians of different widths, in a method known as *Difference of Gaussians* (DoG).

Laplacian of Gaussian

It is a Gaussian-based operator which uses the Laplacian to take the second derivative of an image.

This really works well when the transition of the gray level seems to be abrupt.

It works on the zero-crossing method i.e when the second-order derivative crosses zero, then that particular location corresponds to a maximum level. It is called an edge location. The Gaussian operator reduces the noise and the Laplacian operator detects the sharp edges.

Laplacian of Gaussian

The Laplacian of Gaussian of the image is defined by the formula:

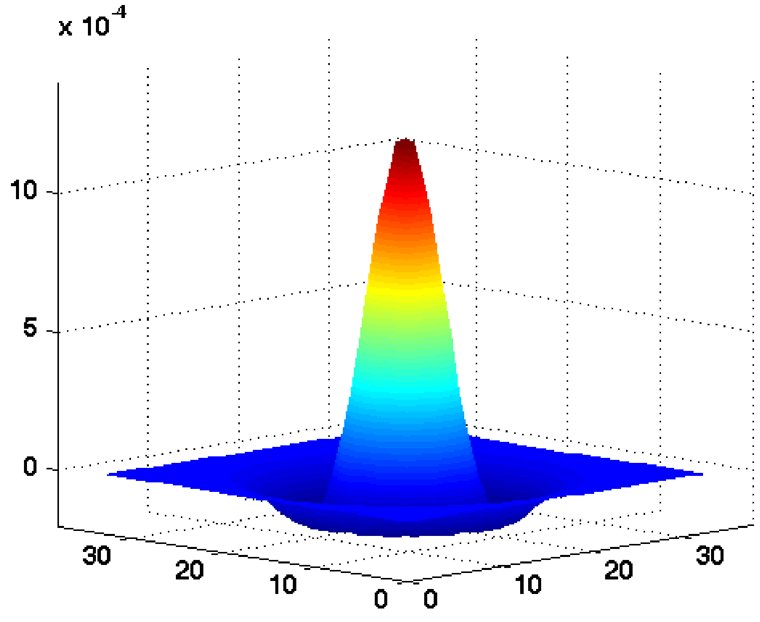
$$\nabla^2(f * g) = f * \nabla^2 g$$

with g the Gaussian kernel and * the convolution

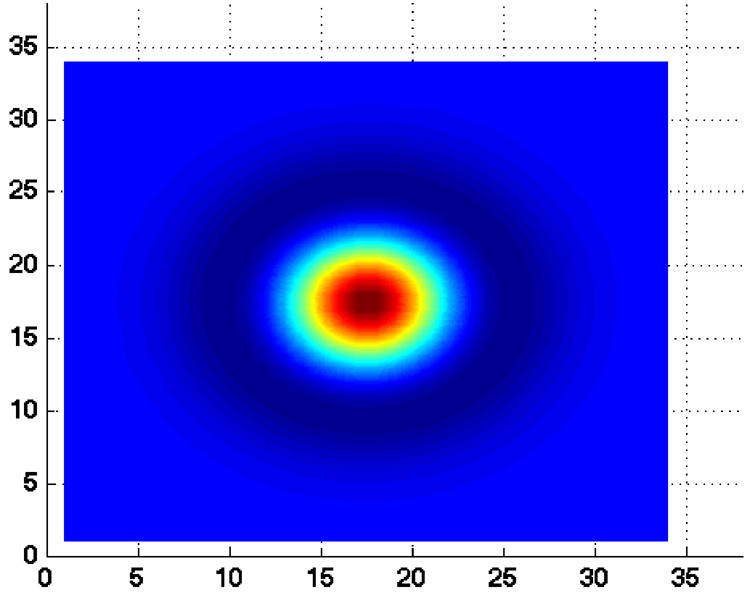
This convolution can be further expanded, in the 2D case, as

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

Laplacian of Gaussian (LoG) transfer function (Mexican hat)



(a)

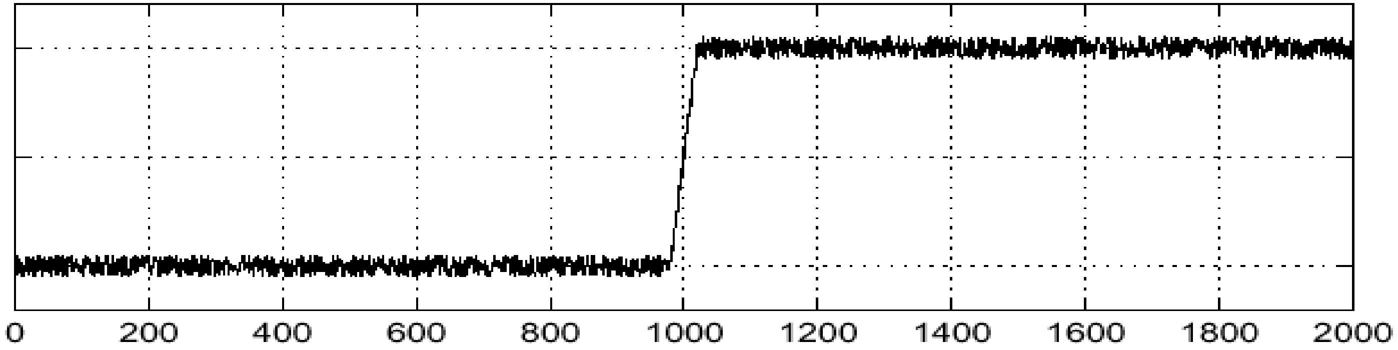


(b)

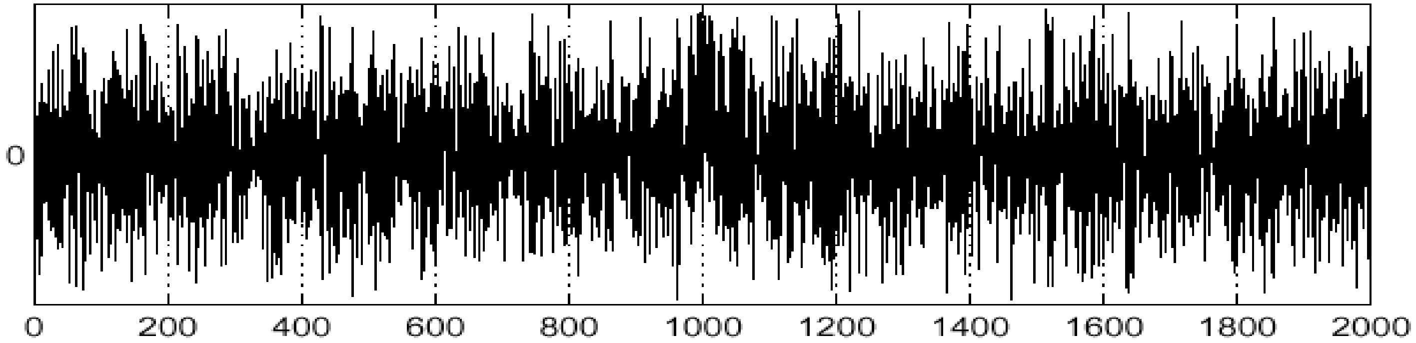
Effect of Noise

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

$$f(x)$$

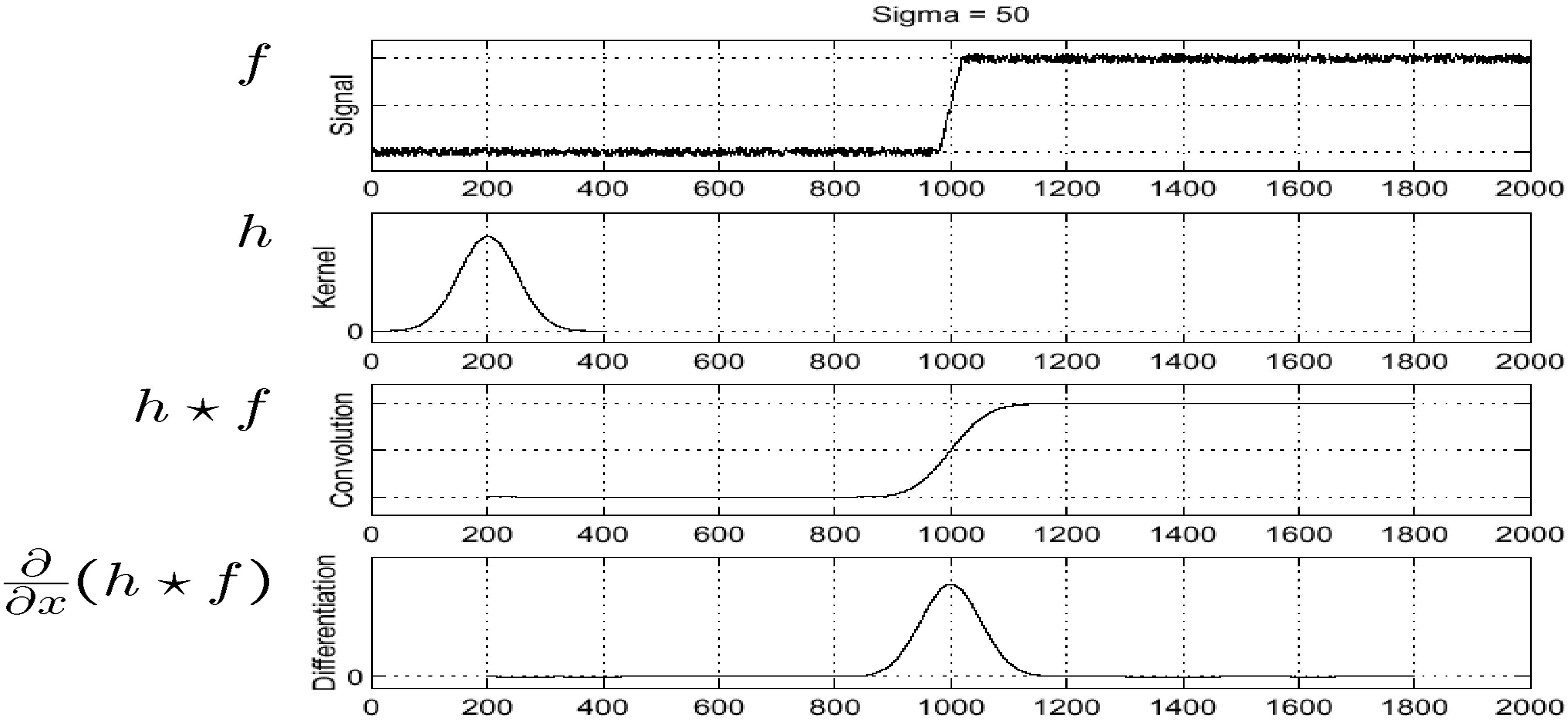


$$\frac{d}{dx}f(x)$$



Where is the edge??

Solution



Where is the edge?

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

Derivative Theorem of Convolution

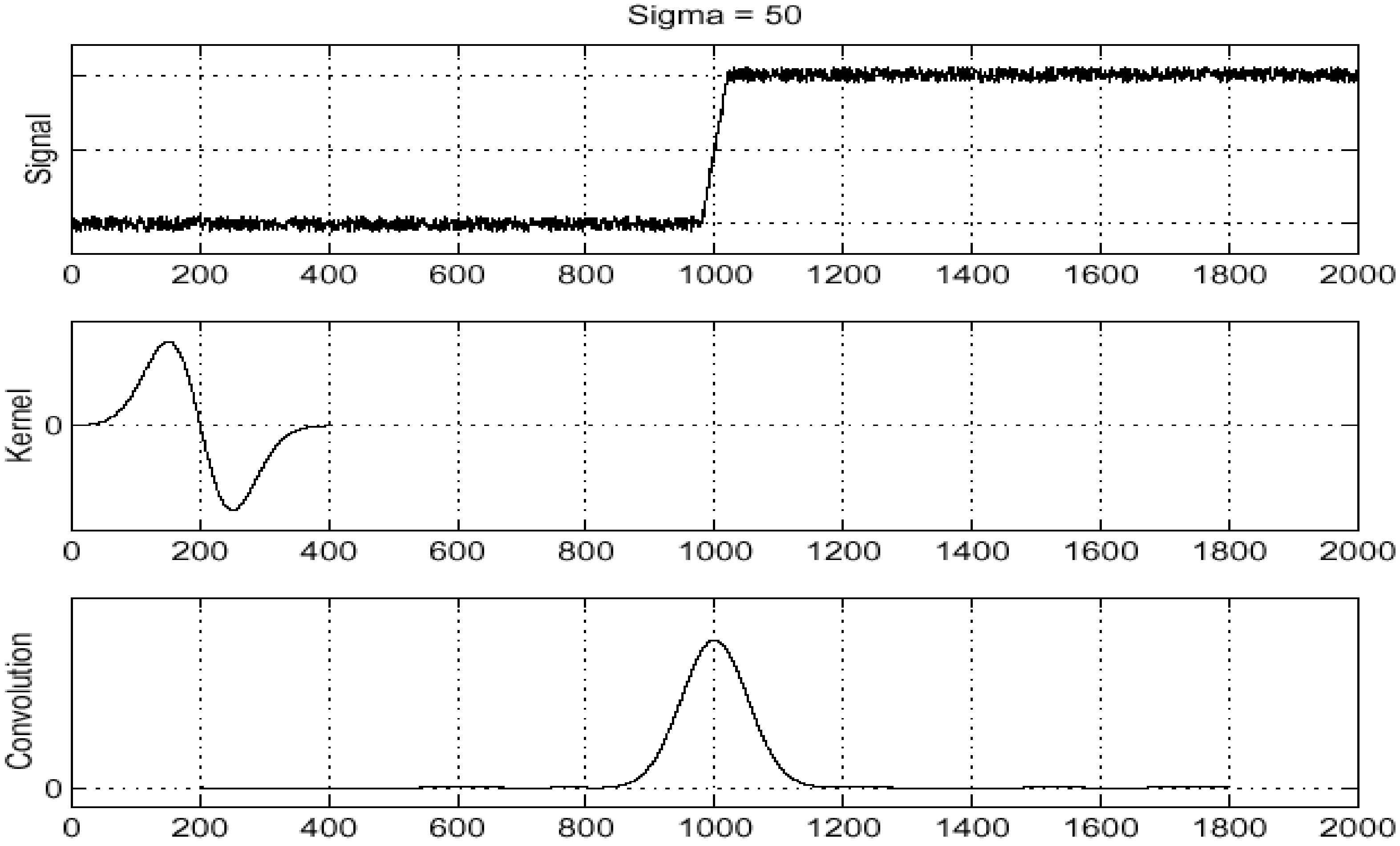
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

…saves us one operation.

f

$\frac{\partial}{\partial x}h$

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



Derivative Theorem of Convolution

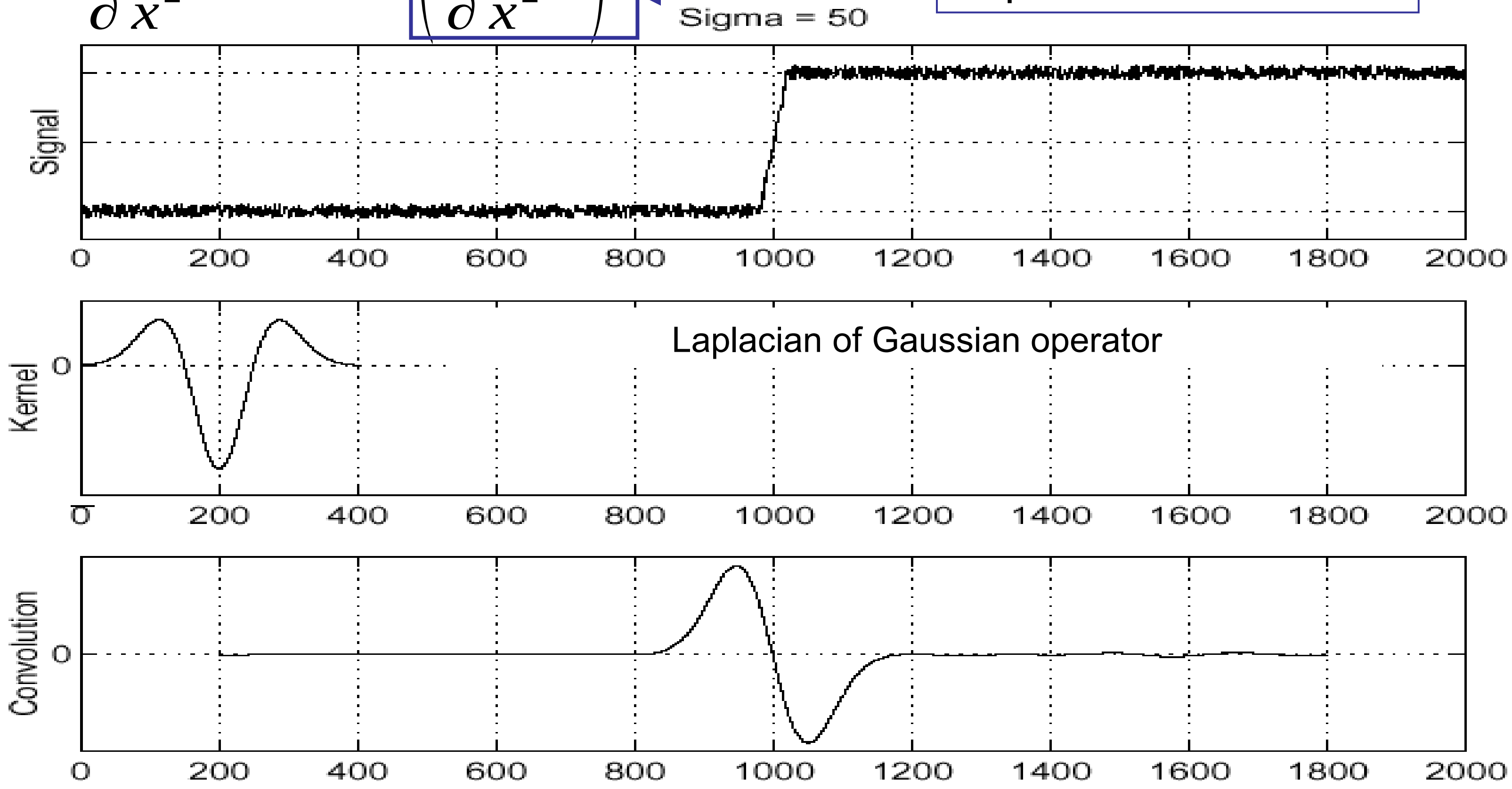
$$\frac{\partial^2}{\partial x^2}(h * f) = \boxed{\left(\frac{\partial^2}{\partial x^2}h\right)} * f$$

Laplacian of Gaussian

f

$\frac{\partial^2}{\partial x^2}h$

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



Where is the edge?

Zero-crossings of bottom graph !

Laplacian of Gaussian

Advantages:

- Easy to detect edges and their various orientations
- There is fixed characteristics in all directions

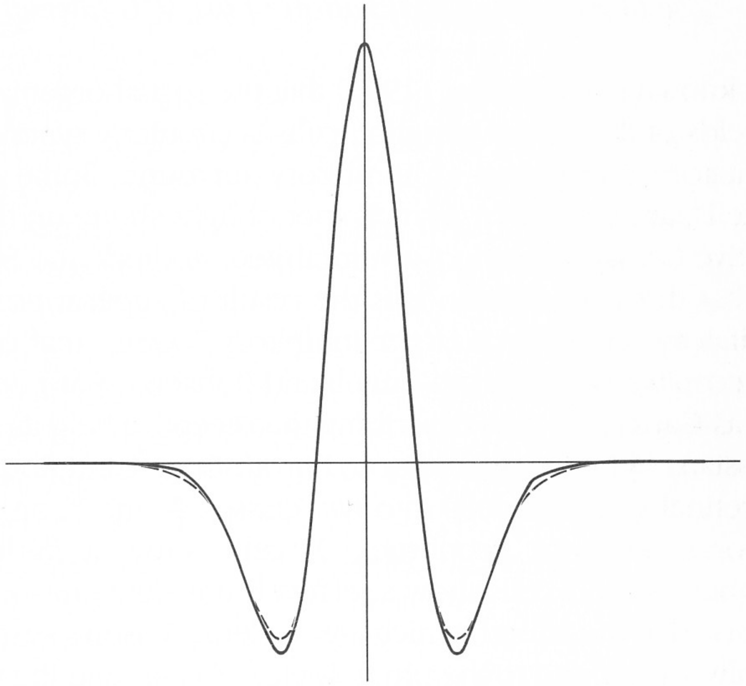
Limitations:

- Very sensitive to noise
- The localization error may be severe at curved edges
- It generates noisy responses that do not correspond to edges, so-called “false edges”

Difference of Gaussian

The difference of Gaussian (DoG) of image can be written as

$$f * g_{(1)} - f * g_{(2)} = f * (g_{(1)} - g_{(2)})$$



Canny Edge Detection

It is a gaussian-based operator in detecting edges. This operator is not susceptible to noise. It extracts image features without affecting or altering the feature.

Canny edge detector have advanced algorithm derived from the previous work of Laplacian of Gaussian operator. It is widely used as an optimal edge detection technique. It detects edges based on three criteria:

- Low error rate
- Edge points must be accurately localized
- There should be just one single edge response

Canny Edge Operator

- Smooth image *I* with 2D Gaussian: $G * I$

- Find local edge normal directions for each pixel

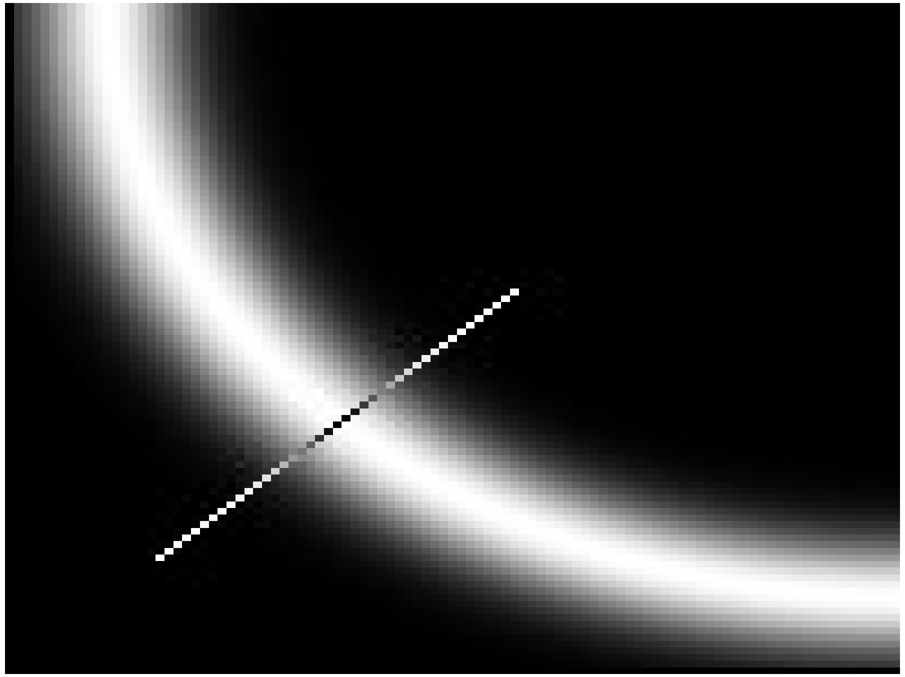
$$\overline{n} = \frac{\nabla(G * I)}{\left|\nabla(G * I)\right|}$$

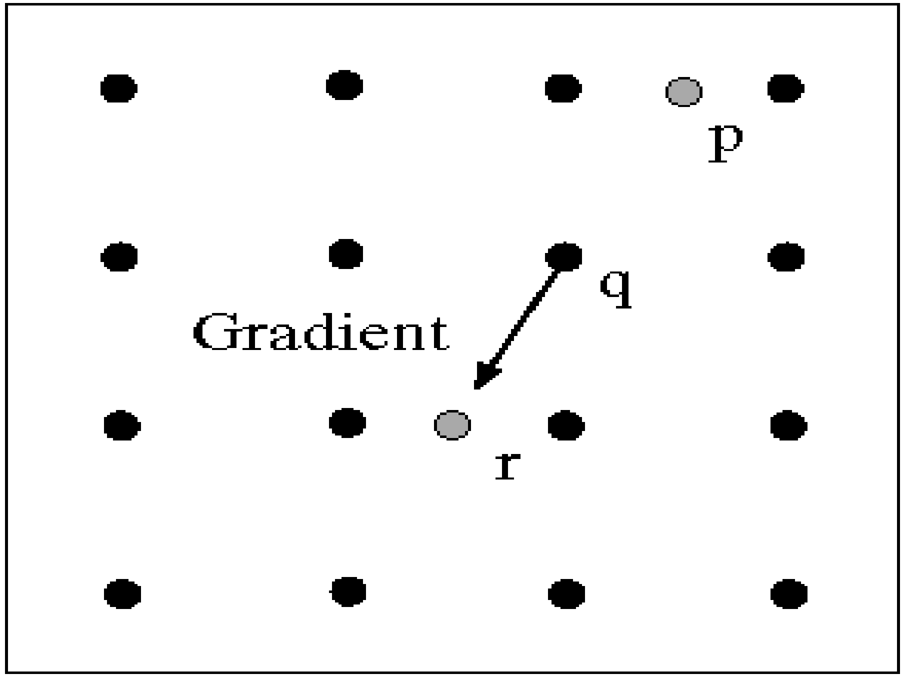
- Compute edge magnitudes $\left|\nabla(G * I)\right|$

- Locate edges by finding zero-crossings along the edge normal directions (**non-maximum suppression**)

$$\frac{\partial^2 (G * I)}{\partial \overline{n}^2} = 0$$

Canny Edge Operator





- Check if pixel is local maximum along gradient direction
  - requires checking interpolated pixels p and r

Canny Edge Operator

Lenna

Edge

Magnitude of Gradient

Edge linking and boundary detection

Goal of edge detection: to produce an image containing *only the edges* of the original image.

However, due to the many technical challenges (noise, shadows, occlusion, etc,), most edge detection algorithms will output an image containing fragmented edges.

Additional processing is needed to turn fragmented edge segments into useful lines and object boundaries.

Hough transform: a global method for edge linking and boundary detection.

Hough Transform

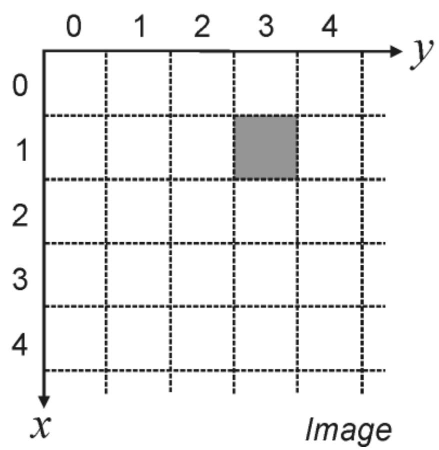
- A mathematical method designed to find lines in images.
  - It can be used for linking the results of edge detection, turning potentially sparse, broken, or isolated edges into useful lines that correspond to the actual edges in the image.

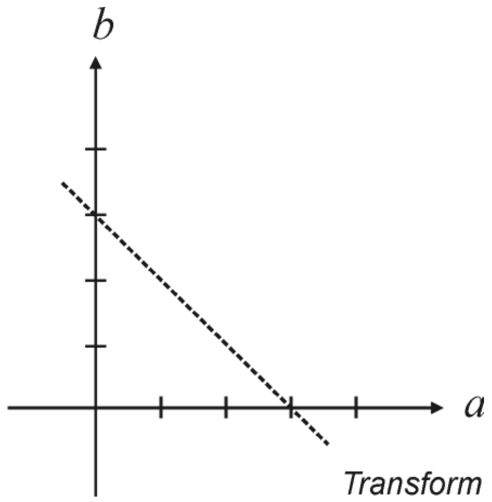
Hough Transform

Let *(x,y)* be the coordinates of a point in a binary image (containing threshold edge detection results).

The Hough transform stores in an *accumulator array* all pairs *(a,b)* that satisfy the equation $y = ax + b$. The *(a,b)* array is called the *transform array*.

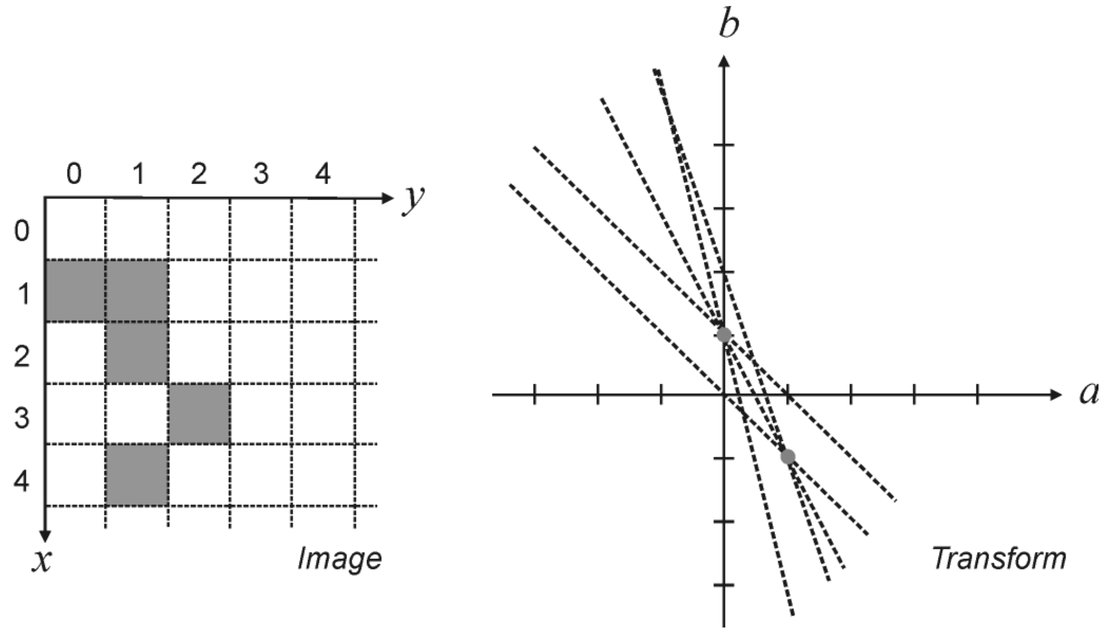
Example:, the point $(x,y) = (1,3)$ in the input image will result in the equation $b = -a + 3$, which can be plotted as a line that represents all pairs *(a,b)* that satisfy this equation.





Hough Transform

- Since each point in the image will map to a line in the transform domain, repeating the process for other points will result in many intersecting lines, one per point.
- The meaning of two or more lines intersecting in the transform domain is that the points to which they correspond are aligned in the image.
- The points with the greatest number of intersections in the transform domain correspond to the longest lines in the image.



Hough Transform