



# FEATURE EXTRACTION

Aziz M. Qaroush

Birzeit University

First Semester  
2015/2016

# Outline

2

- Element of Image Analysis
- Feature Extraction Overview
- Image Main features
  - ▣ Local and Texture Feature
  - ▣ Color Feature
  - ▣ Shape feature
  - ▣ Features Post processing/Analysis

# Element of Image Analysis

3

## Preprocess

Image Acquisition, Enhancement, and Restoration



## Intermediate process

Feature extraction & Image segmentation



## High level process

Image interpretation and recognition

# Features Extraction

4

- Feature = “point of interest” for image description
  - ▣ Features should contain information required to distinguish between classes
    - Should be insensitive to irrelevant variability in the input
  
- Main goal of feature extraction
  - ▣ Obtain the most relevant information from the original data
  - ▣ Represent that information in a lower dimensionality space.

# Features Extraction

5

- Feature extraction is a special form of dimensionality reduction
  - ▣ When the input data is too large to be processed and it is suspected to be redundant (much data, but not much information) then the input data will be transformed into a reduced representation set of features (also named features vector).
- Used by classifiers to recognize the input unit
- Used in many applications such as
  - ▣ Character recognition
  - ▣ Reading bank deposit slips
  - ▣ data entry
  - ▣ Image retrieval
  - ▣ .....

# Features Extraction Classification

6

□ Features can be classified as:

▣ **General features:**

- Application independent features such as **color, texture, and shape**.
- According to the abstraction level, they can be further divided into:
  - **Pixel-level features:** Features calculated at each pixel, e.g. color, location.
  - **Local features:** Features calculated over the results of subdivision of the image band on image segmentation or edge detection.
  - **Global features:** Features calculated over the entire image or just regular sub-area of an image.

# Features Extraction Classification

7

- Features can be classified as:
  - **Domain-specific features:**
    - **Application dependent features:** such as human faces, fingerprints, Characters, and conceptual features.
      - These features are often a synthesis of low-level features for a specific domain.

# Features Extraction Classification

8

- On the other hand, all features can be coarsely classified into:
  - ▣ **Low-level features:** features can be extracted directly from the original images
    - Edges
    - Corners
    - Interest points
  - ▣ **High-level features:** high-level feature extraction must be based on low level features
    - Shape
    - Template Matching



# Characteristics of good features

9

- **Identifiability:** shapes which are found perceptually similar by human have the same feature different from the others.
- **Repeatability:** The same feature can be found in several images despite geometric (**Translation, rotation and scale invariance**) and photometric (**Intensity**) transformations
- **Noise resistance:** features must be as robust as possible against noise, i.e., they must be the same whichever be the strength of the noise in a give range that affects the pattern.

# Efficiency of features

10

- **Occultation invariance:** when some parts of a shape are occulted by other objects, the feature of the remaining part must not change compared to the original shape.
- **Statistically independent:** two features must be statistically independent. This represents compactness of the representation.
- **Reliable:** as long as one deals with the same pattern, the extracted features must remain the same.

# What is best method for feature extraction

11

- ❑ It all depends on your application at hand. Few things you should keep in mind are:
  - ❑ Feature extraction is highly subjective in nature
  - ❑ There is no generic feature extraction scheme which works in all cases.
    - ❑ What kind of problem are you trying to solve? e.g. classification, regression, clustering, etc.
    - ❑ Do you have a lot of data?
    - ❑ Do your data have very high dimensionality?
    - ❑ Is your data labelled?
    - ❑ Do you want to use a very computationally intensive method or something rather inexpensive?

# Image Features

12

- Image Main Features:
  - **Local Features**
  - Color Features
  - Shape Features

# Local Features – Motivation

13

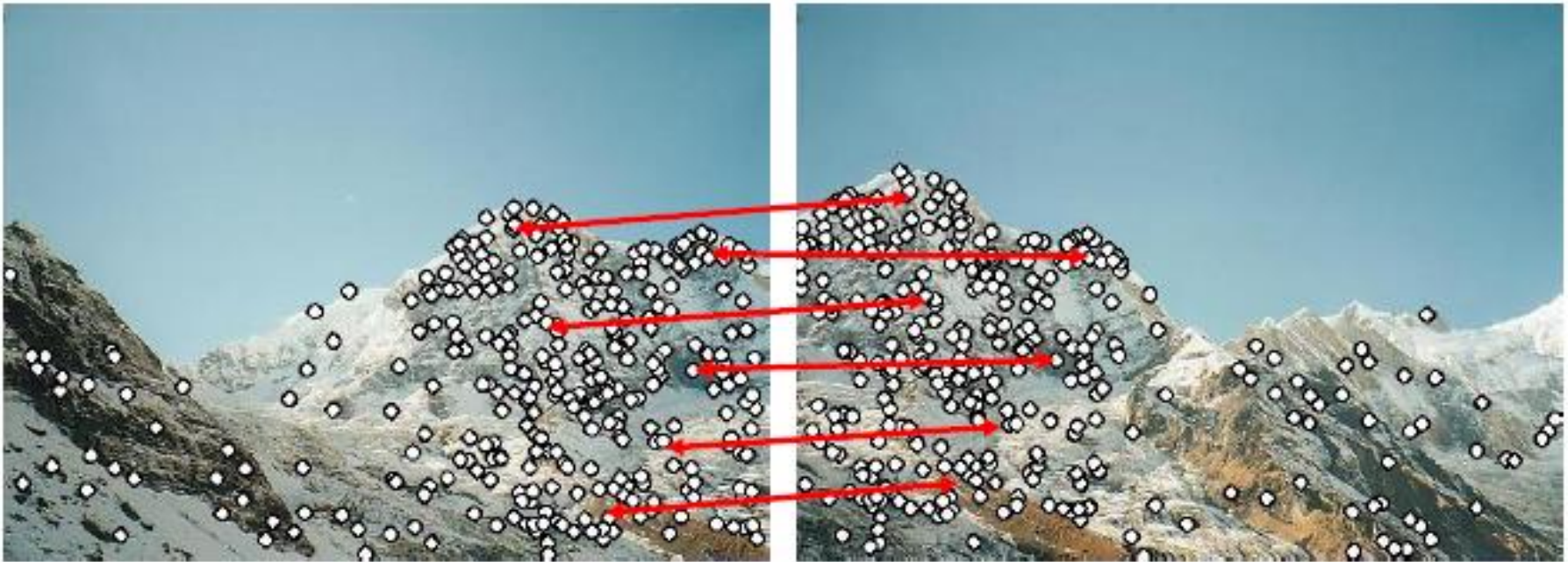
- Panorama stitching
  - ▣ We have two images – how do we combine them?



# Local Features – Motivation

14

- Panorama stitching
  - ▣ We have two images – how do we combine them?



Extract and match features

# Why extract features?

15

- Panorama stitching
  - ▣ We have two images – how do we combine them?



Align images

# Local Features

16

- Features that can be extracted **automatically** from an image **without any shape information** (information about *spatial* relationships)
- Can be used in high-level feature extraction, where we find shapes in Images.
- Types
  - ▣ Edge
  - ▣ Corner
  - ▣ Interest points



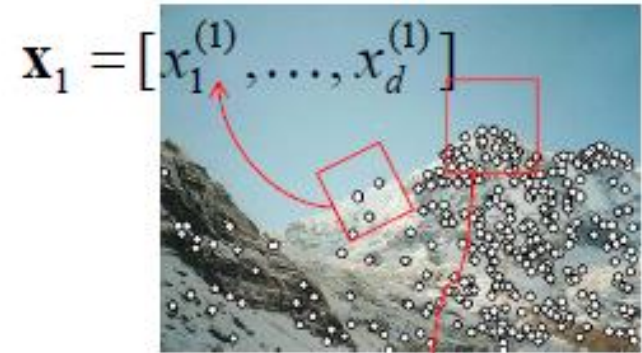
# Local Features extraction: main components

17

1. **Detection:** Identify the interest points

2. **Description:** Extract feature vector descriptor surrounding each interest point.

3. **Matching:** Determine correspondence between descriptors in two views



$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$



# Local Features

- ▶ **Edge**
- ▶ **Corner**
- ▶ **Interest Points**

# Local Features - Edge

19

- **Edges can be classified according to intensity profiles into**
  - **Step Edge (ideal edge)**
    - *involves transition between two intensity levels over a distance of one pixel*
    - *Occur mostly in computer generated images*
  - **Ramp Edge**
    - *The transition between two intensity levels occur over a distance that is greater than one pixel*
    - *Appear in real images as a result of noise and focusing limitations of imaging devices*
  - **Roof Edge**
    - *Essentially, they represent blurred lines that pass through a region*



a b c

**FIGURE 10.8**

From left to right, models (ideal representations) of a step, a ramp, and a roof edge, and their corresponding intensity profiles.

# Edge Detection

20

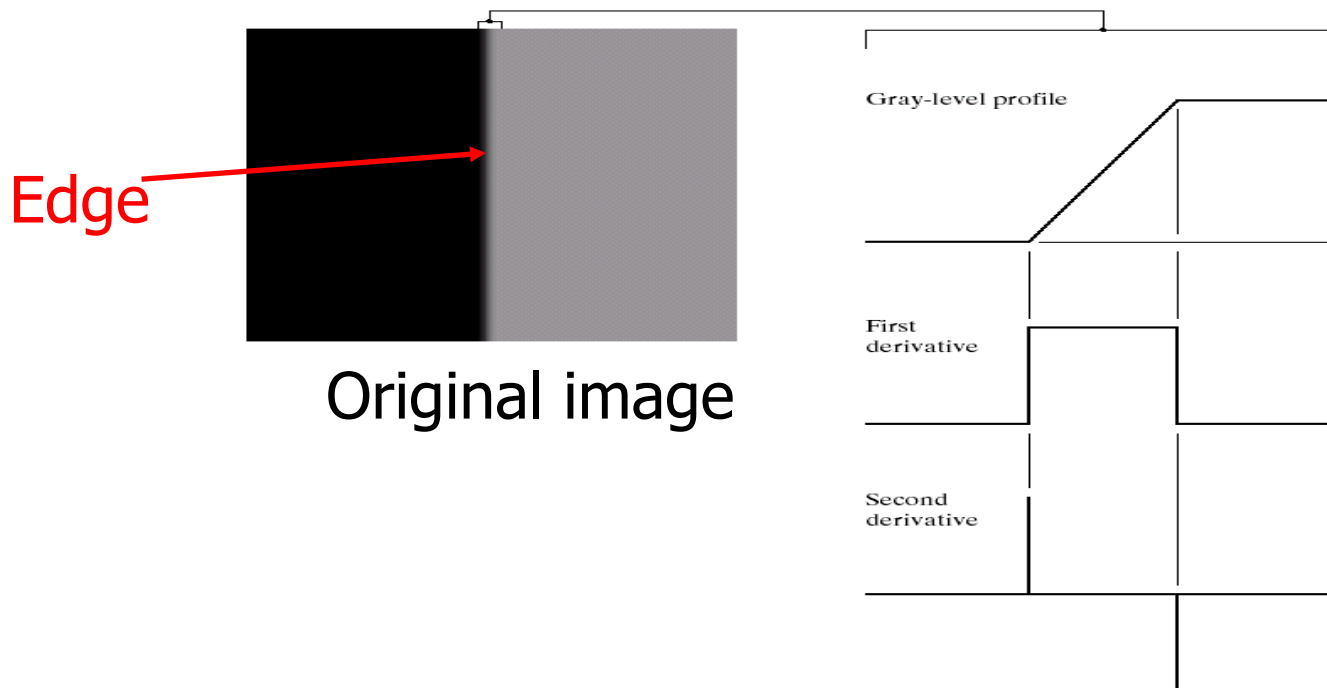
- It is not unusual to find the three types of edges in one image



# Edge Detection

21

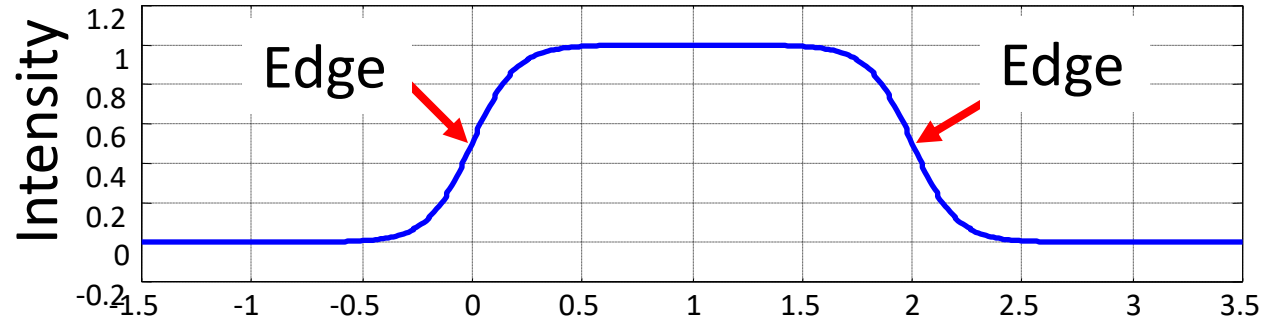
- Detection of edges can be through the use of first-order or second-order derivatives
  - *For the first derivative, the magnitude can be used to detect the presence of an edge*
  - *For the second derivative, the sign of the second derivative is used to detect the presence of the edge*



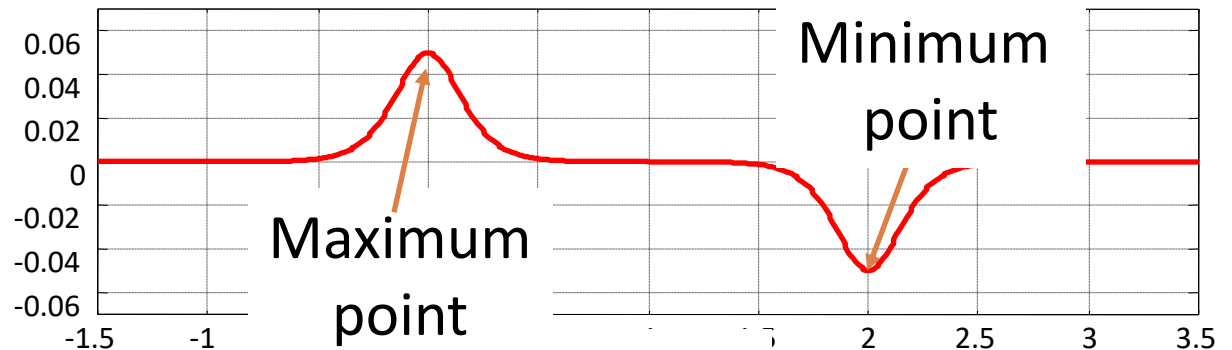
# Edge Detection

22

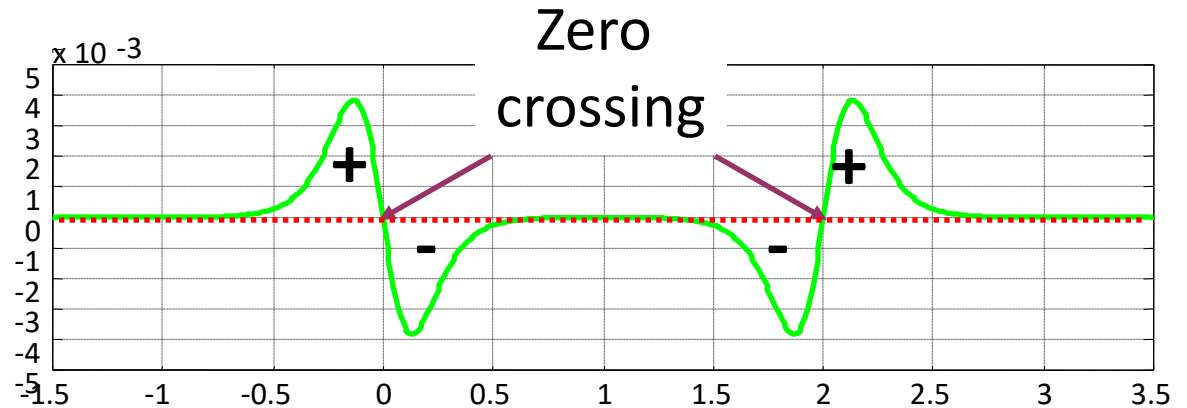
Gray level profile



The 1<sup>st</sup> derivative



The 2<sup>nd</sup> derivative



Therefore, for detecting edges, we can apply zero crossing detection to the 2<sup>nd</sup> derivative image or thresholding the absolute of the 1<sup>st</sup> derivative image

# Edge Detection Using Gradient

23

- The gradient is a powerful tool in finding the strength and direction of edges. The gradient at pixel  $(x,y)$  is defined as

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

- The magnitude of the gradient measures the strength of the edge (maximum rate of change)

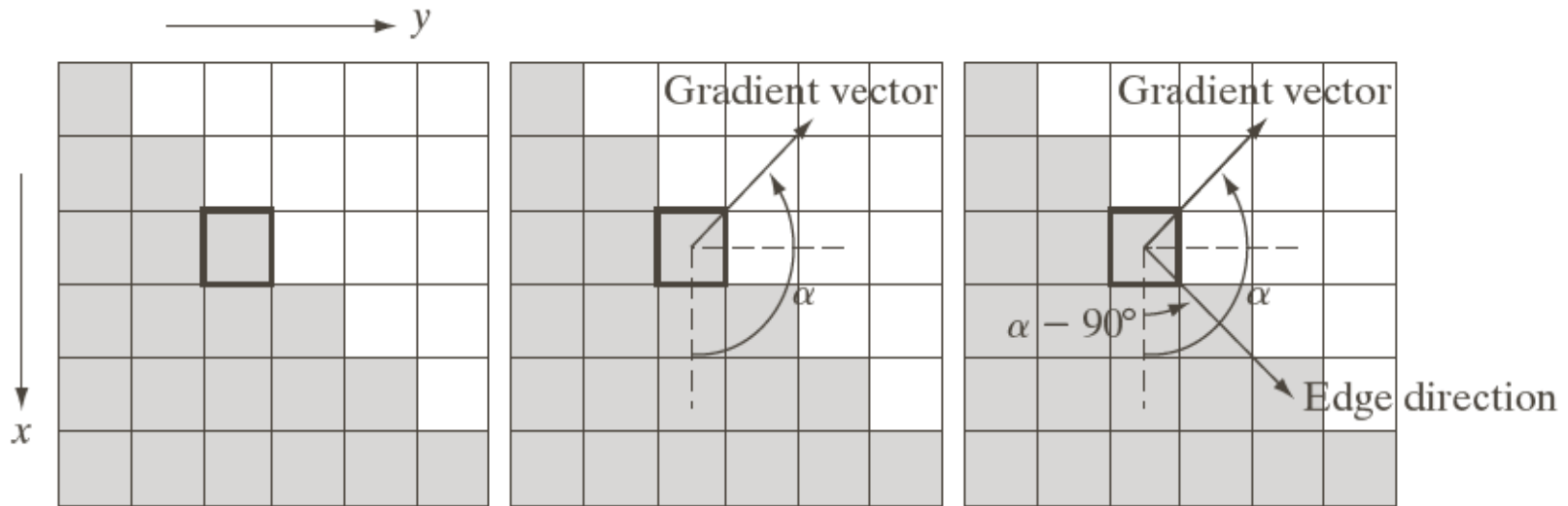
$$M(x,y) = \left[ \left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2 \right]^{1/2} \quad |\nabla f| \approx |G_x| + |G_y|$$

- The direction of the gradient is perpendicular to the edge direction

$$\alpha = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$

# Edge Detection Using Gradient

24



$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$$

$$\alpha = \tan^{-1}\left(\frac{G_y}{G_x}\right) = 135^\circ$$

- All edge pixels have the same gradient magnitude and direction



# Edge Detection Using Gradient

25

- **Gradient Masks to compute gradient at  $Z_5$** 
  - **Discrete 1<sup>st</sup> Derivative**

$$G_x = \frac{\partial f}{\partial x} = f(x+1, y) - f(x, y) \\ = Z_8 - Z_5$$

-1
1

$$G_y = \frac{\partial f}{\partial y} = f(x, y+1) - f(x, y) \\ = Z_6 - Z_5$$

-1	1
----	---

- Not efficient in detecting diagonal edges

- **Roberts Cross-gradient Operator**

-1	0
0	1

Horizontal Operator

0	-1
1	0

Vertical Operator

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Pixel z5 and its neighbours

$$G_x(x, y) = z_9 - z_5$$

$$G_y(x, y) = z_8 - z_6$$

- 2x2 masks are not as good as symmetric masks which capture information from opposite sides around the center point

# Edge Detection Using Gradient

26

- **Gradient Masks to compute gradient at  $Z_5$**

- **Prewitt Operators**

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

Mask to Compute  $G_x$

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

Mask to Compute  $G_y$

$$G_x = \frac{\partial f}{\partial x} = (Z_7 + Z_8 + Z_9) - (Z_1 + Z_2 + Z_3)$$

$$G_y = \frac{\partial f}{\partial y} = (Z_3 + Z_6 + Z_9) - (Z_1 + Z_4 + Z_7)$$

- **Sobel Operators**

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

Mask to Compute  $G_x$

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

Mask to Compute  $G_y$

$$G_x = \frac{\partial f}{\partial x} = (Z_7 + 2Z_8 + Z_9) - (Z_1 + 2Z_2 + Z_3)$$

$$G_y = \frac{\partial f}{\partial y} = (Z_3 + 2Z_6 + Z_9) - (Z_1 + 2Z_4 + Z_7)$$

- They have better response than Prewitt masks and have better smoothing which is essential to reduce the effect of noise

# Edge Detection Using Gradient

27

- **Gradient Masks to compute gradient at  $Z_5$** 
  - Prewitt and Sobel masks shown before give the strongest response for horizontal and vertical edges. We can modify these masks to obtain better response for diagonal edges **Prewitt Operators**

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

Prewitt Diagonal Masks

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

Sobel Diagonal Masks

# Edge Detection Using Gradient

28



Image



$G_x$  computed using  
Sobel Operator



$G_y$  computed using  
Sobel Operator



$|G_x| + |G_y|$



Angle of Gradient

# Edge Detection Using Gradient

29

- **Example – continued**

- Note that in the previous slide, the edges of the wall bricks were successfully detected. However, this might not be desirable if we are interested in the main edges
- We can eliminate such small edges (which might be considered as noise) by
  - Smoothing the image before computing the gradient
  - Thresholding the gradient image
  - Smoothing followed by thresholding



Gradient Image ( $|G_x| + |G_y|$ ) without smoothing



Gradient Image ( $|G_x| + |G_y|$ ) after the original image was smoothed by 5x5 mask

# Edge Detection Using Gradient

30

- **Example – continued**



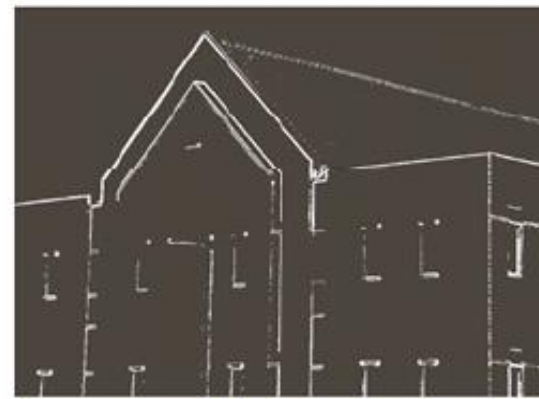
Gradient Image ( $|G_x| + |G_y|$ )  
without smoothing



Thresholded Gradient Image



Gradient Image ( $|G_x| + |G_y|$ ) after the original  
image was smoothed by 5x5 mask



Thresholded Gradient Image (better  
connectivity for edges)

# Edge Detection Using Gradient

31

- **Example – continued**

- The Sobel masks used in the previous slides were those that have stronger response for vertical and horizontal directions. How about diagonal directions?



Gx computed using Sobel Operator (Horizontal)



Gy computed using Sobel Operator (vertical)



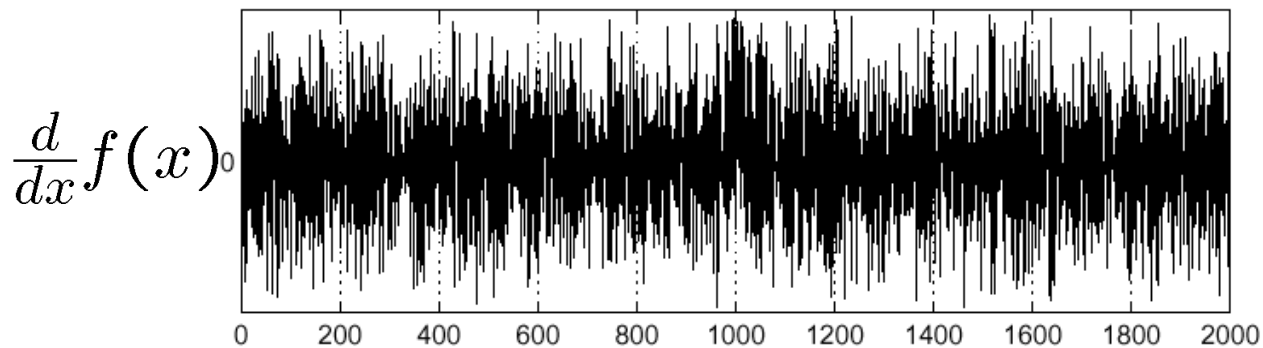
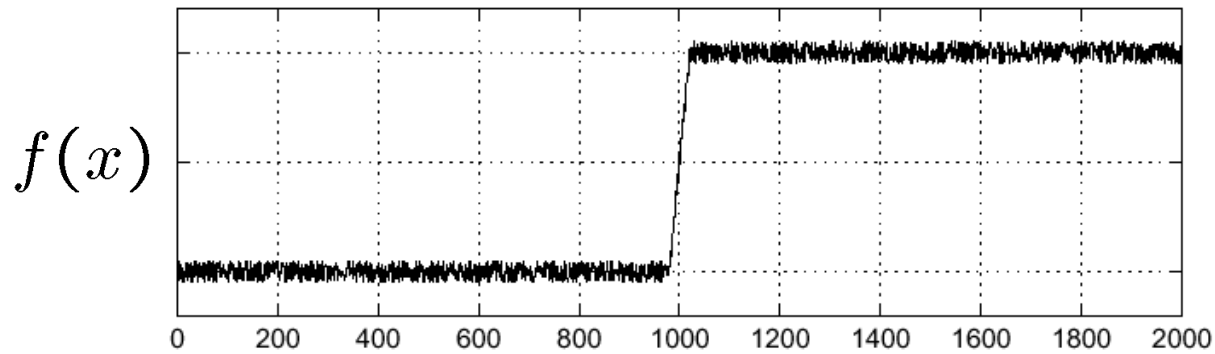
Gx computed using Diagonal Sobel Operator (+45)



Gx computed using Diagonal Sobel Operator (-45)

# Effects of noise

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



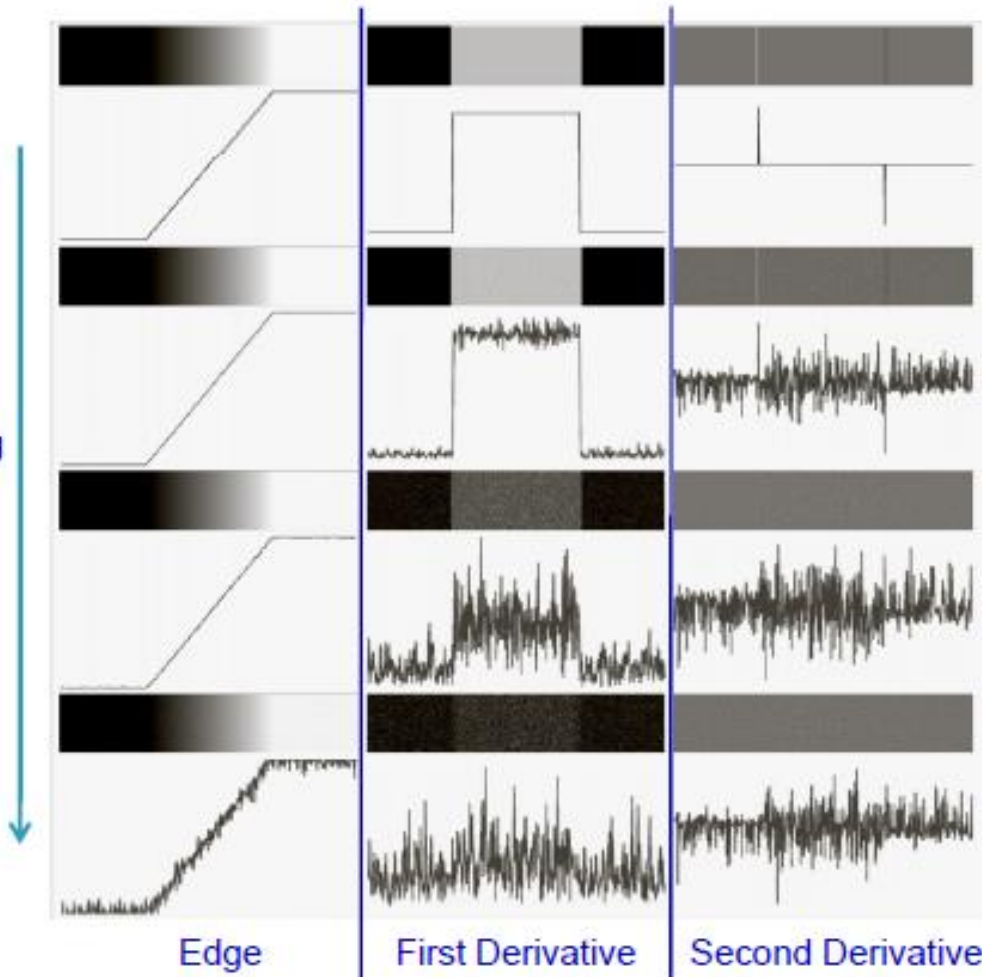
Where is the edge?



# Edge Detection with Noise

33

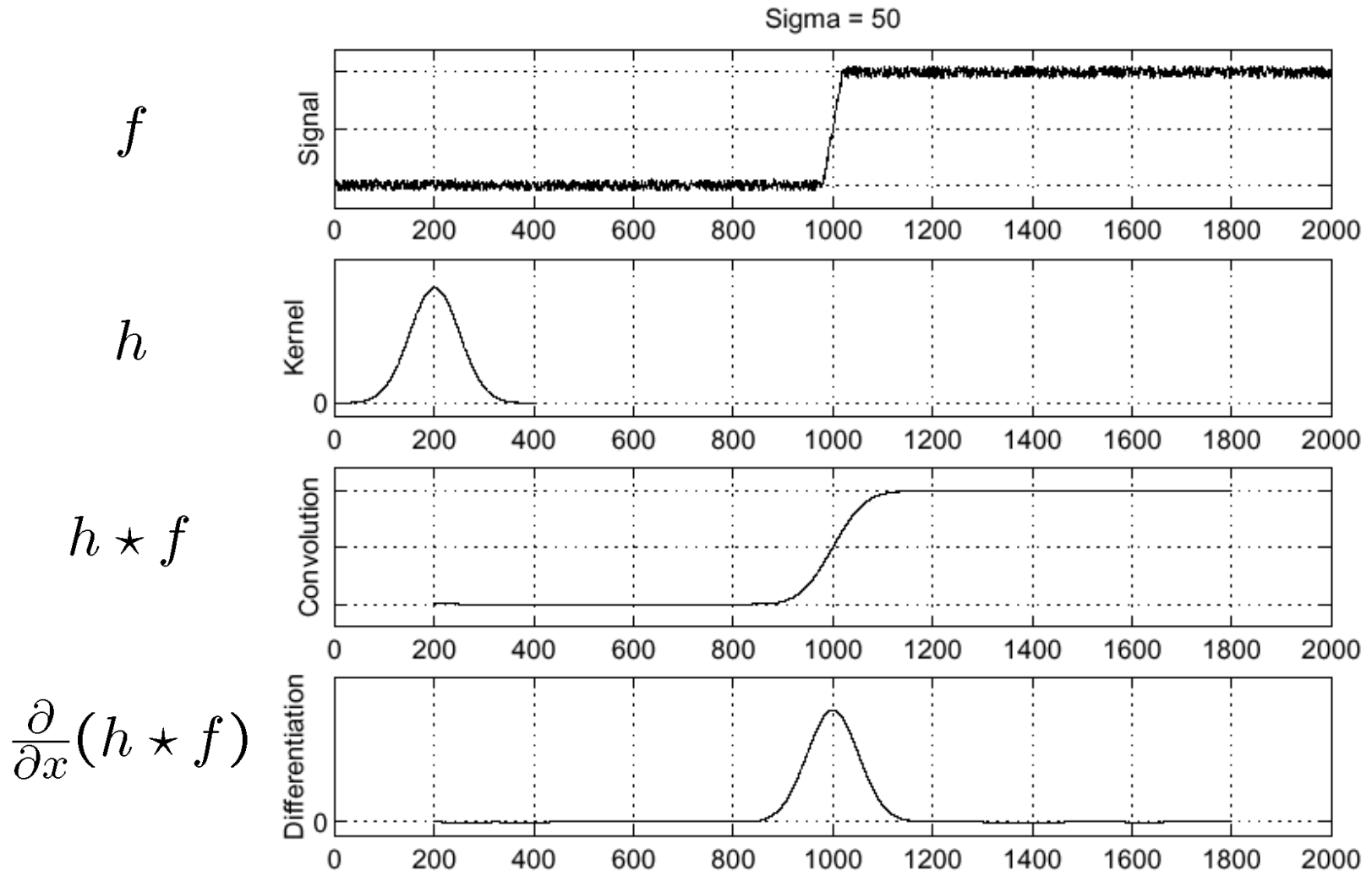
- Although it sounds straight forward to detect edges using first-order or second-order derivatives, the presence of noise may affect this operation significantly depending on noise level



**In general, detecting edges, involves :**

- 1) Smoothing to reduce the sensitivity of derivatives to noise
- 2) Detection of all possible edge points
- 3) Edge localization to select from the candidate point those that comprise the edge

# Solution: smooth first



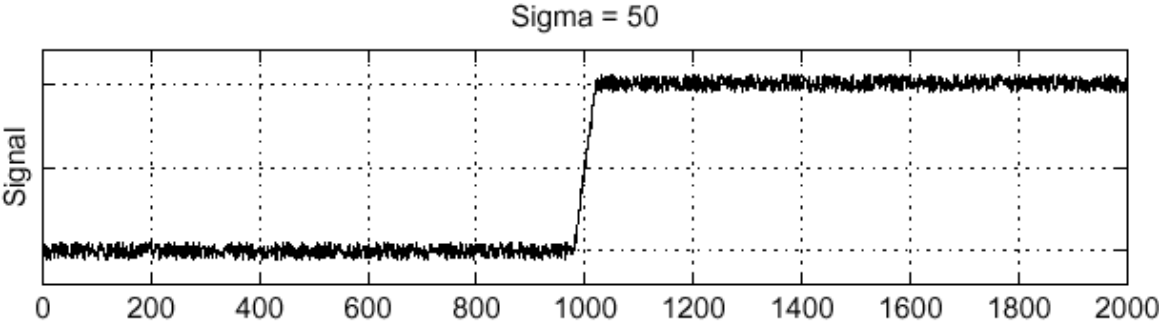
Where is the edge? Look for peaks in  $\frac{\partial}{\partial x}(h \star f)$

# Derivative theorem of convolution

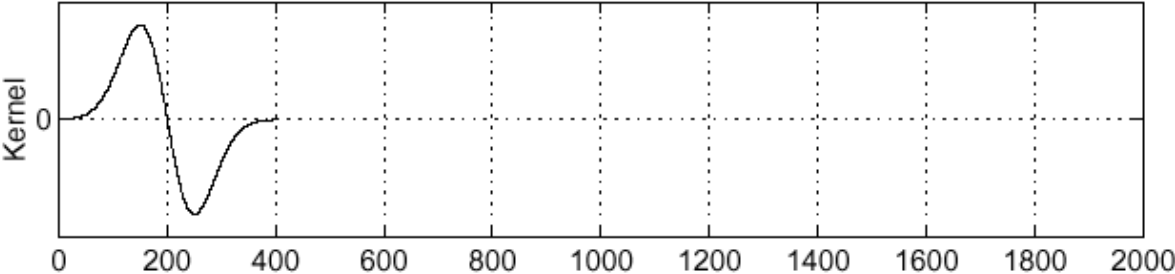
$$\frac{\partial}{\partial x}(h \star f) = \left(\frac{\partial}{\partial x}h\right) \star f$$

This saves us one operation:

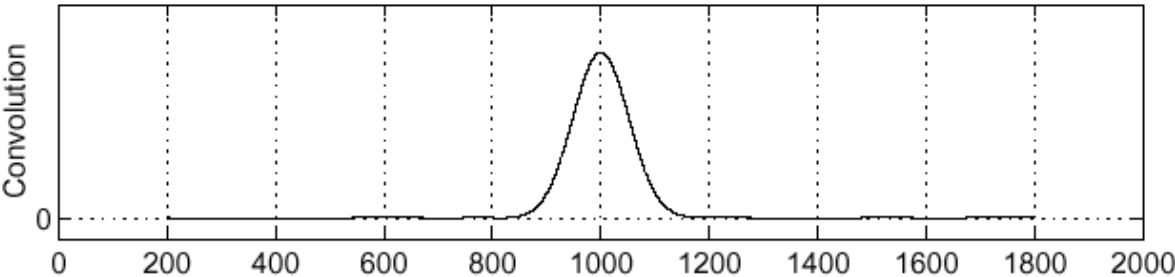
$f$



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



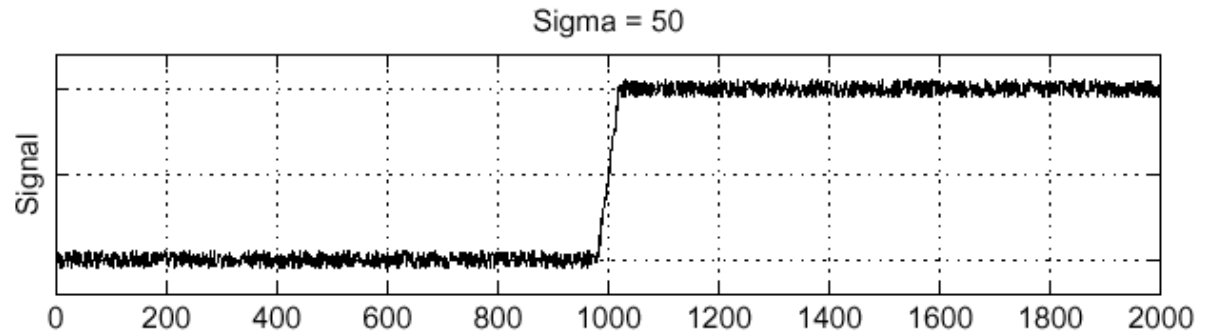
$\left(\frac{\partial}{\partial x}h\right) \star f$



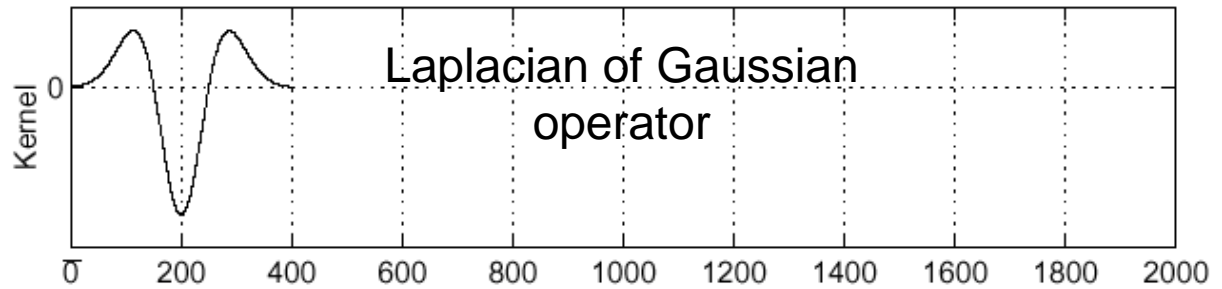
# Laplacian of Gaussian

- Look for zero-crossings of  $\frac{\partial^2}{\partial x^2}(h \star f)$

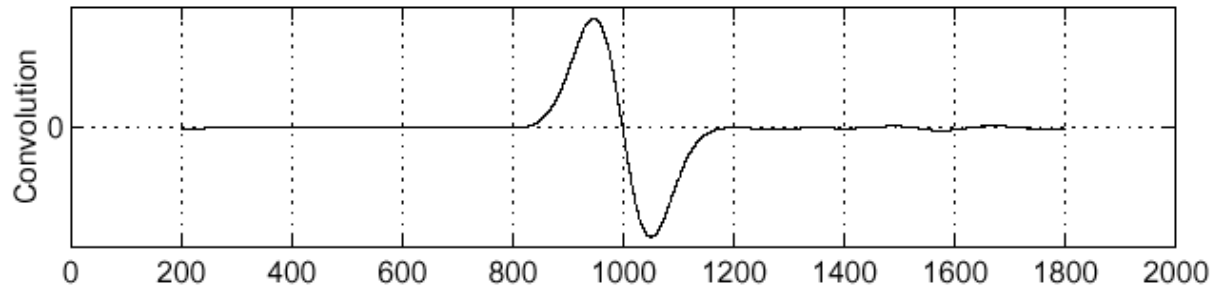
$f$



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

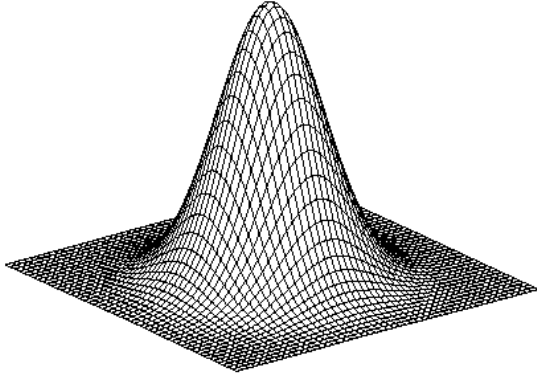


$(\frac{\partial^2}{\partial x^2}h) \star f$

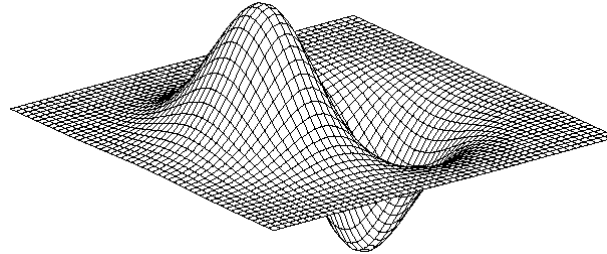


# 2D edge detection filters

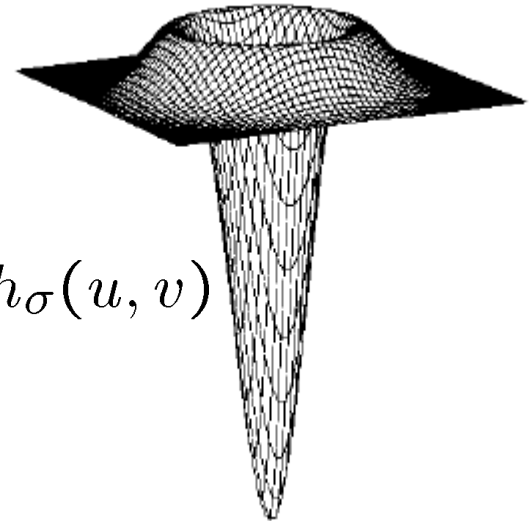
Gaussian



derivative of Gaussian



Laplacian of Gaussian



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



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

$\nabla^2$  is the **Laplacian** operator:

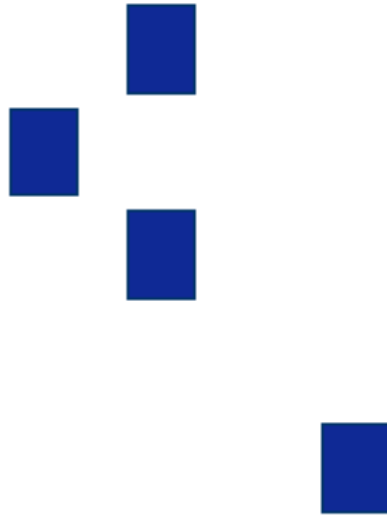
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

# Quality of an Edge

38



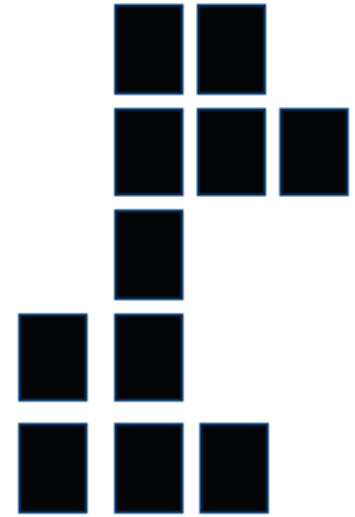
True edge



Poor robustness to noise



Poor localization



Too many responses

# Advanced Edge Detection Method - Canny Edge Detector

39

- The Canny algorithm has three goals:
  - ▣ **Good Detection:** The optimal detector must minimize the probability of false positives as well as false negatives.
  - ▣ **Good Localization:** The edges detected must be as close as possible to the true edges.
  - ▣ **Single Response Constraint:** The detector must return one point only for each edge point.

# Canny Algorithm

40

- The Process of Canny edge detection algorithm can be broken down to 5 different steps:
  1. Apply Gaussian filter to smooth the image in order to remove the noise
  2. Find the intensity gradients of the image
  3. Apply non-maximum suppression to get rid of spurious response to edge detection
  4. Apply double threshold to determine potential edges
  5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.



# Canny Algorithm – Step 1

41

- Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian smoothing filter.

$$\frac{1}{159}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

# Canny Algorithm – Step 2

42

- The second step is to use Sobel masks to find the edge gradient strength and direction for each pixel.
  - ▣ The magnitude, or edge strength, of the gradient is then approximated using the formula:  $|G| = |G_x| + |G_y|$

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

$G_x$

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

$G_y$

- ▣ The direction of the edge is computed using the gradient in the x and y directions

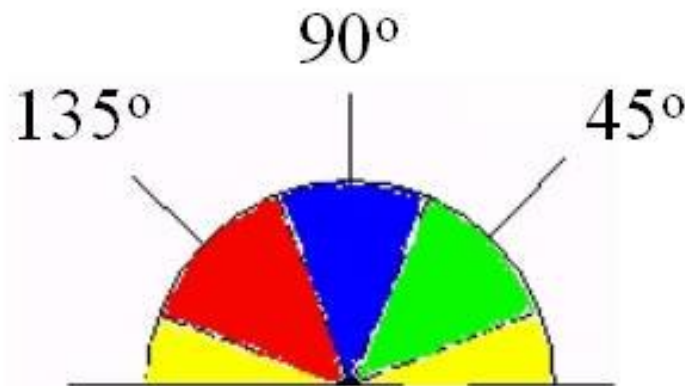
$$\theta[i, j] = \tan^{-1}(G_y / G_x)$$

# Canny Algorithm – Step 3

43

## □ Gradient Orientation

- ▣ Reduce angle of Gradient  $\theta[i,j]$  to one of the 4 sectors
- ▣ Check the 3x3 region of each  $M[i,j]$ 
  - Any edge direction falling within the **yellow range** (0 to 22.5 & 157.5 to 180 degrees) is set to 0 degrees. Any edge direction falling in the **green range** (22.5 to 67.5 degrees) is set to 45 degrees. Any edge direction falling in the **blue range** (67.5 to 112.5 degrees) is set to 90 degrees. And finally, any edge direction falling within the **red range** (112.5 to 157.5 degrees) is set to 135 degrees.



# Canny Algorithm – Step 4

44

- The edge extracted from the gradient value is still quite blurred.
- Non-maximum suppression can help to suppress all the gradient values to 0 except the local maximal, which indicates location with the sharpest change of intensity value.
- The algorithm for each pixel in the gradient image is:
  - ▣ Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
  - ▣ If the edge strength of the current pixel is the largest compared to the other pixels in the mask with the same direction (i.e., the pixel that is pointing in the  $y$  direction, it will be compared to the pixel above and below it in the vertical axis), the value will be preserved. Otherwise, the value will be suppressed.

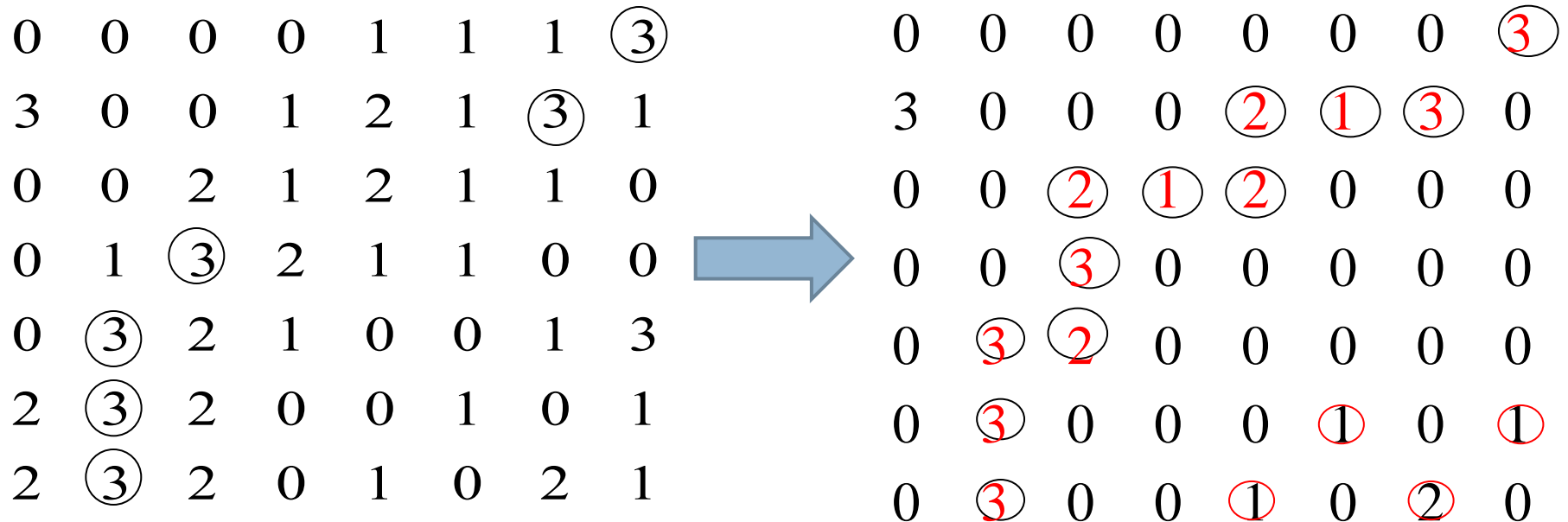
# Canny Algorithm – Step 4

45

## □ Non-maximum Suppression

□ For each pixel (i,j):

- Find the direction  $d_k$ , which best approximates the direction
- Check the 3x3 region of each  $M[i,j]$
- If  $M[i,j]$  is smaller than at least one of its two neighbors along  $d_k$ , assign  $I[i,j]=0$ ; otherwise assign  $I[i,j]=M[i,j]$



# Canny Algorithm – Step 4

46

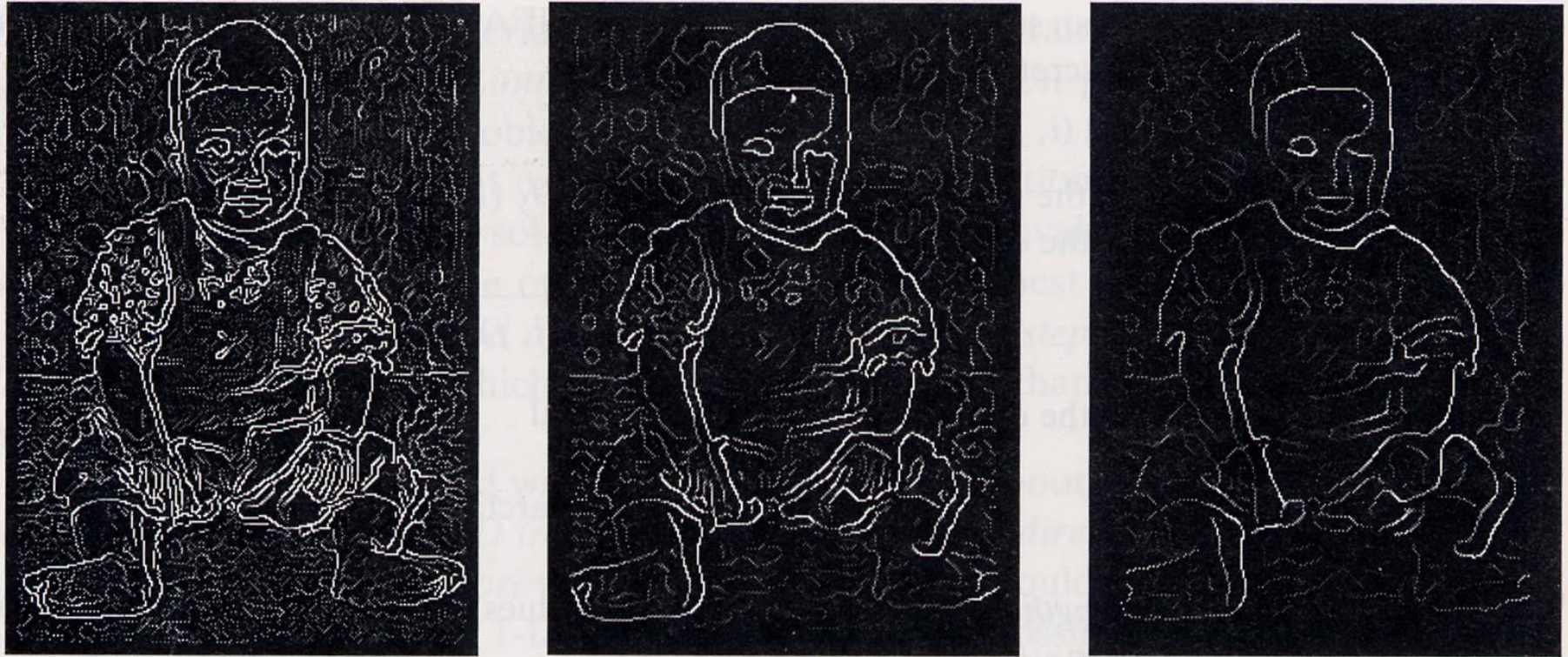


Figure 4.5 Strength images output by `CANNY_ENHANCER` run on Figure 4.1, after nonmaximum suppression, showing the effect of varying the filter's size, that is, the standard deviation,  $\sigma_f$ , of the Gaussian. Left to right:  $\sigma_f = 1, 2, 3$  pixel.

# Canny Algorithm – Step 5

47

## □ Hysteresis Thresholding

- ▣ The image output by NONMAX- SUPPRESSION  $I[i,j]$  still contains the local maxima created by noise. How do we get rid of these?
- ▣ Reduce number of false edges by applying a threshold  $T$ 
  - All values below  $T$  are changed to 0
  - Selecting a good values 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

# Canny Algorithm – Step 5

48

## □ Hysteresis Thresholding

### ▣ Double Thresholding

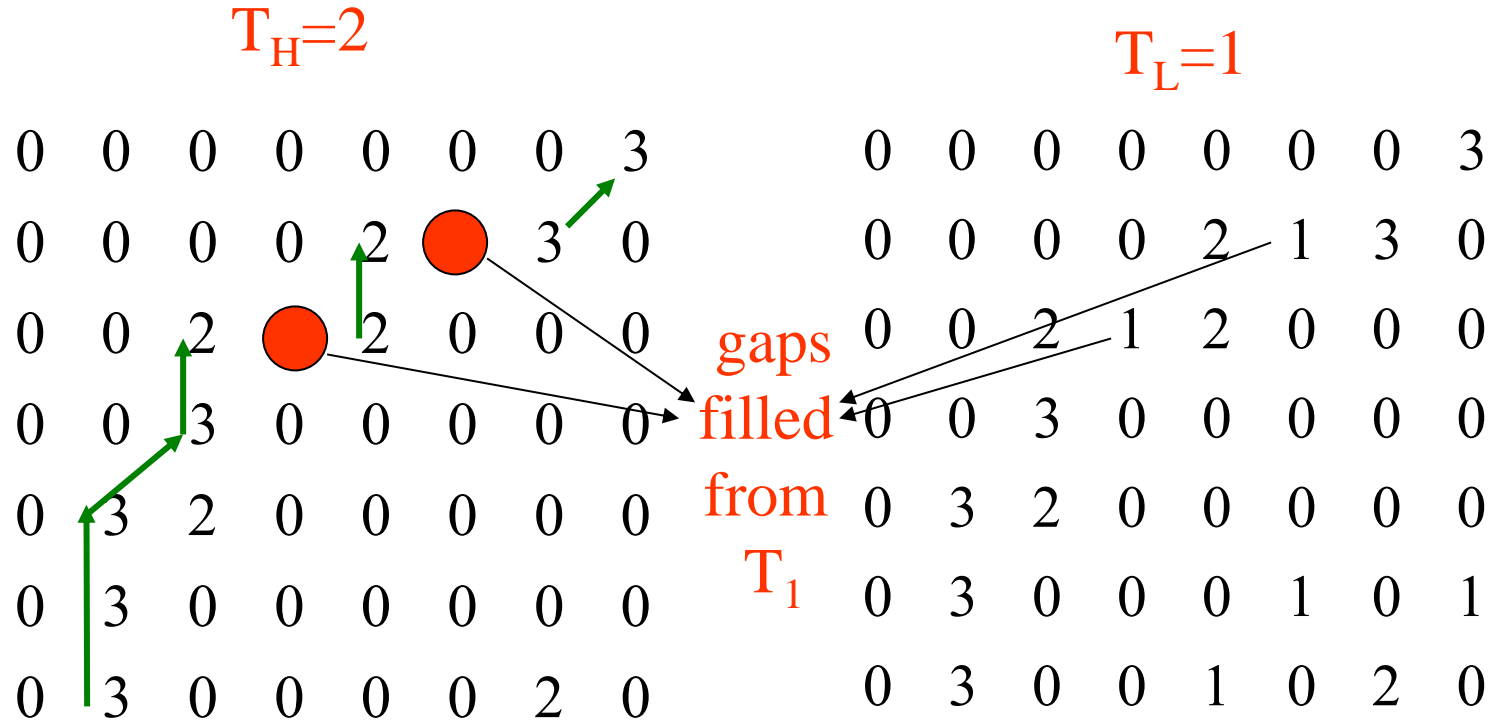
- Two threshold values,  $T_L$  and  $T_H$  are applied to  $I[i,j]$ .
- Here  $T_L < T_H$
- Two images in the output
- The image from  $T_L$  contains fewer edges but has gaps in the contours
- The image from  $T_L$  has many false edges
- **Combine** the results from  $T_L$  and  $T_H$
- Link the edges of  $T_H$  into contours until we reach a gap
- link the edge from  $T_H$  with edge pixels from a  $T_L$  contour until a  $T_H$  edge is found again



# Canny Algorithm – Step 5

49

## □ Hysteresis Thresholding



- A  $T_H$  contour has pixels along the green arrows
- **Linking:** search in a 3x3 of each pixel and connect the pixel at the center with the one having greater value
- Search in the direction of the edge (direction of Gradient)

# Canny Algorithm – Step 5

50

## □ Hysteresis Thresholding



Gaussian  
smoothing



(a) Hysteresis thresholding,  
upper level = 40,  
lower level = 10



(b) Uniform thresholding,  
level = 40

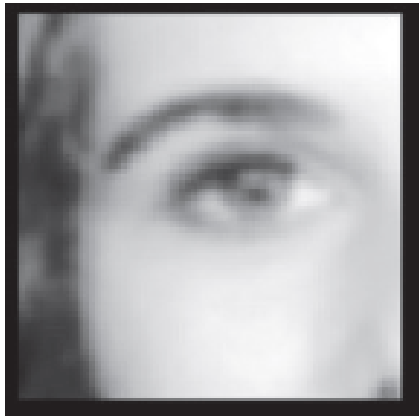


(c) Uniform thresholding,  
level = 10

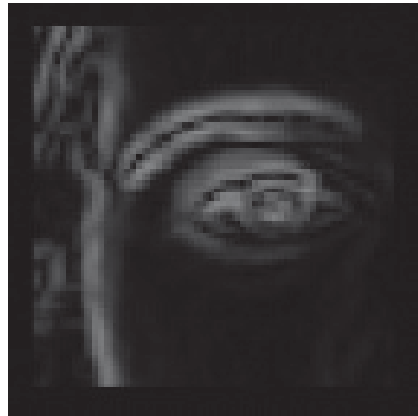
# Canny Algorithm

51

## □ Stages in Canny edge detection - Example



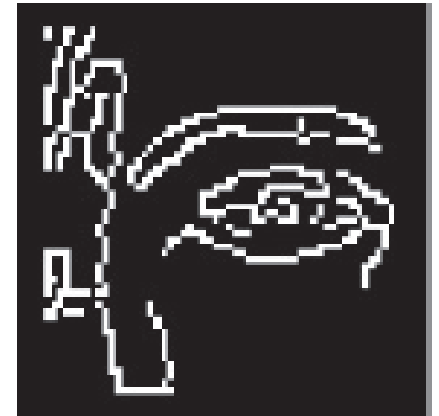
(a) Gaussian smoothing



(b) Sobel edge detection



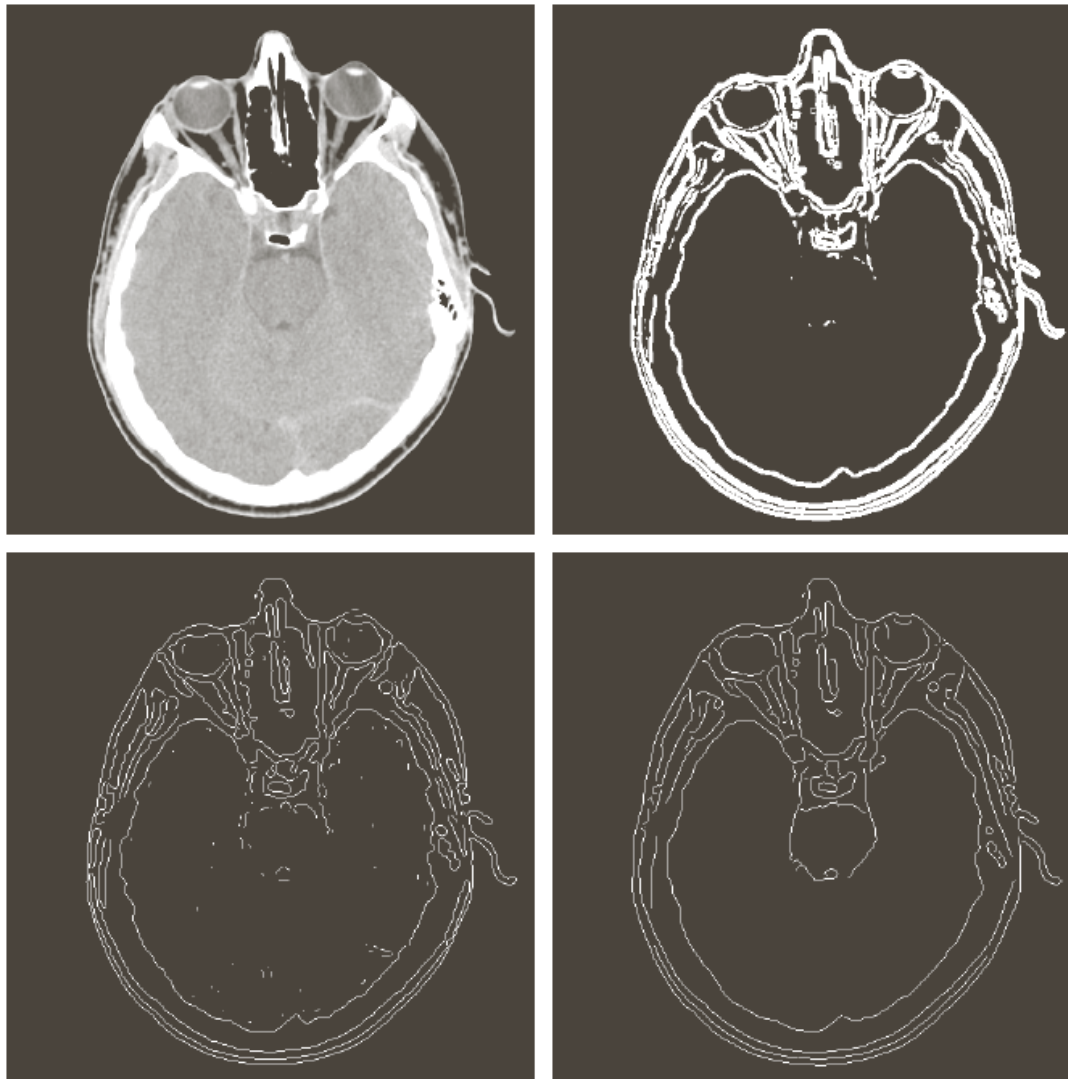
(c) Non-maximum suppression



(d) Hysteresis thresholding

# Canny Edge Detection

52



a	b
c	d

**FIGURE 10.26**

(a) Original head CT image of size  $512 \times 512$  pixels, with intensity values scaled to the range  $[0, 1]$ .

(b) Thresholded gradient of smoothed image.

(c) Image obtained using the Marr-Hildreth algorithm.

(d) Image obtained using the Canny algorithm.

(Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

# Canny Edge Detection Summary

53

- The performance of the Canny algorithm depends heavily on the adjustable parameters, , which is  $\delta$  and the threshold values, 'T1' and 'T2'.
  - ▣ The bigger the value for  $\delta$ , the larger the size of the Gaussian filter becomes. This implies more blurring, necessary for noisy images, as well as detecting larger edges.
  - ▣ However, the larger the scale of the Gaussian, the less accurate is the localization of the edge.
  - ▣ The user can tailor the algorithm by adjusting these parameters to adapt to different environments.
  - ▣ Canny's edge detection algorithm is computationally more expensive compared to Sobel, Prewitt and Robert's operator. However, the Canny's edge detection algorithm performs better than all these operators under almost all scenarios

# Edge Detection Summery

54

- Since edge detection is the initial step in object recognition, it is important to know the differences between edge detection techniques.
- ▣ Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise.
  - The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image.
  - An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels of these images to help distinguish valid image contents from visual artifacts introduced by noise.

# Edge Detection Summery



Original (a)



Sobel (b)



(c)

Prewitt



(d)

Roberts

# Edge Detection Summery

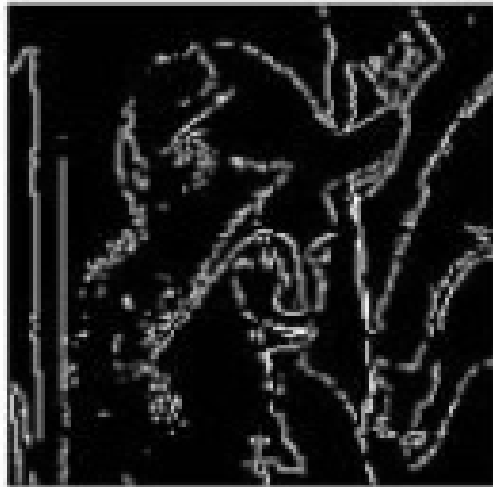
56



Original (a)



Canny (b)



Roberts (c)

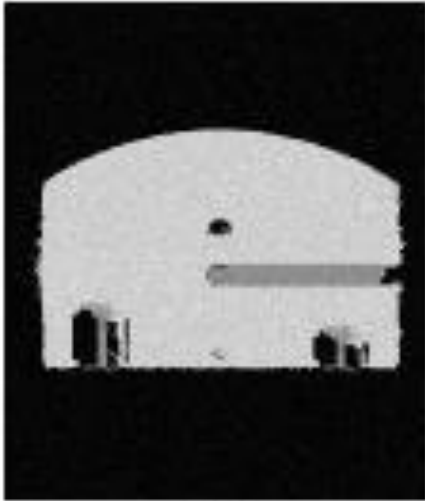


Sobel (e)

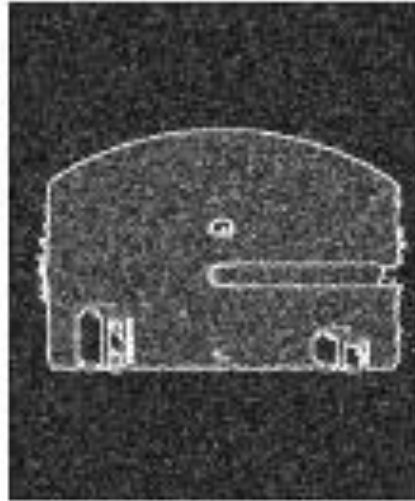


# Edge Detection Summary

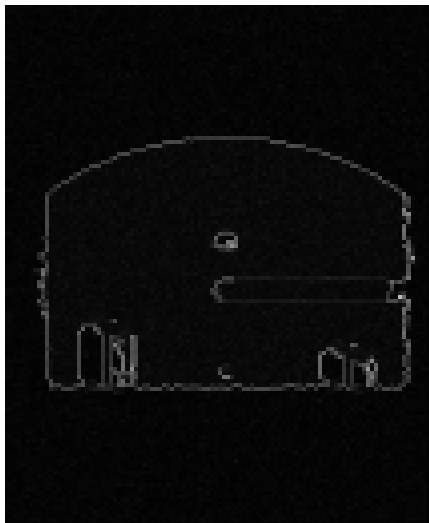
57



(a)



(b)



(c)



(d)

- (a) Original Image with Noise
- (b) Sobel
- (c) Robert
- (d) Canny

# Edge Detection Summery

58

## □ SOFT COMPUTING APPROACHES

- *Fuzzy based Approach*
- *Genetic Algorithm Approach*
- *Neural Network Approach*

- Soft computing approaches, are applied on a real life example image of nature scene

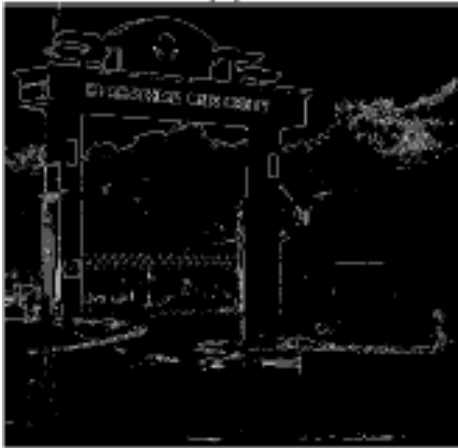
# Edge Detection Summery



Original



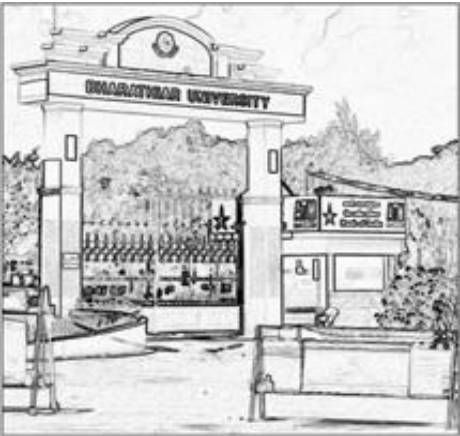
Roberts



Sobel



Fuzzy



Genetics



Neural Network

# Edge Linking and Boundary Detection

60

- Ideally, edge detection should produce the sets of pixels that form the edges in the image
- Practically, this is not always correct !
  - Some detected pixels correspond to noise
  - Some of the edges pixels are not detected due to breaks in the edges
    - Nonuniform illumination
    - Efficiency of the detection method
- It is a common practice to follow the detection process by *linking process* to fill in the breaks and to remove non-edge pixels
- Three common approaches
  - Local - Assumes knowledge about edge points in a local region
  - Regional – Requires that the points on the boundary be known
  - Global – Operates on the entire edge image

# Local Edge Linking

61

- It is based on analyzing a small neighborhood about every candidate edge pixel in the edge image. All pixels that are similar according to some criteria are linked to form an edge using these similar pixels
- Two principle properties used to measure similarity of edge pixels are the magnitude and direction of gradient
  - A pixel  $(s,t)$  in a small neighborhood  $S_{xy}$  around pixel  $(x,y)$  is similar to pixel  $(x,y)$  by gradient magnitude if
$$|M(s,t) - M(x,y)| \leq E$$
  - A pixel  $(s,t)$  in a small neighborhood  $S_{xy}$  around pixel  $(x,y)$  is similar to pixel  $(x,y)$  by gradient direction if
$$|\alpha(s,t) - \alpha(x,y)| \leq A$$
  - The pixel  $(s,t)$  in  $S_{xy}$  is linked to pixel  $(x,y)$  if the previous two conditions are satisfied

# Local Edge Linking

62

- The previous approach is expensive as it requires the investigation of each edge pixel
- Alternatively, we simplify by following these steps
  - 1) Compute the gradient magnitude and angle arrays,  $M(x,y)$  and  $\alpha(x,y)$  of the edge image  $f(x,y)$
  - 2) Form the binary image  $g(x,y)$  by using

$$g(x,y) = \begin{cases} 1, & \text{if } M(x,y) > T_M \text{ and } \alpha(x,y) = A \pm T_A \\ 0, & \text{otherwise} \end{cases}$$

- 3) Scan the rows of  $g(x,y)$  and fill (set to 1) all gaps (sets of 0s) in each row that don't exceed a specified length  $K$ . A gap is bounded by 1s from both ends. Processing is done on each row individually
- 4) To detect gaps in any other direction  $\theta$ , rotate  $g(x,y)$  by  $\theta$  and repeat the horizontal scanning in step 3. Once, done, rotate the result by  $-\theta$

# Local Edge Linking

63

- **Example 10.5.**

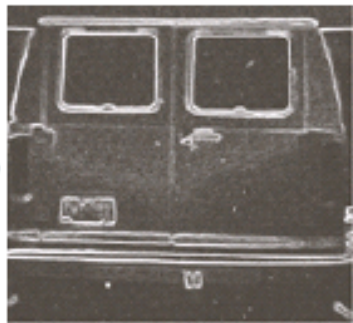
$T_M = 30\%$  of maximum gradient value

$A = 90^\circ$     $T_A = 45^\circ$

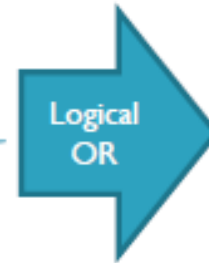
$K = 25$  pixels



Original



Gradient Image



# Global Edge Linking and Hough Transform

64

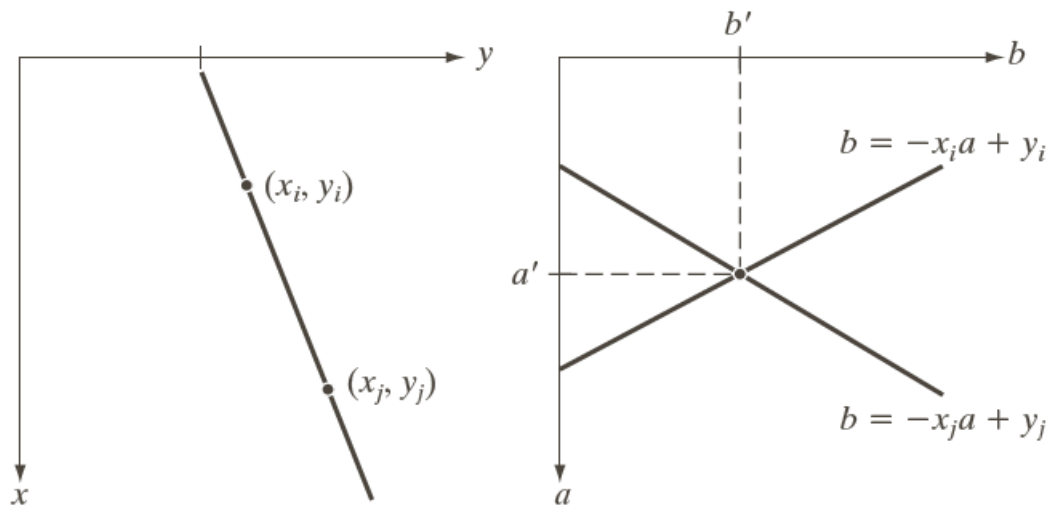
- The input is the edge image with all edge pixels being candidates for linking
- Acceptance or elimination of pixels is based on some global properties, usually a specified shape such as a line or circle
- Given  $n$  edge points, suppose we want to find subsets of these points that lie on straight lines.
- A simple approach is find all possible lines between every pair of points then pick those lines that are close to particular lines
- However, this is computationally expensive !! It requires finding  $n(n-1)/2$  lines and performing  $n^2(n-1)/2$  comparisons !



# Hough Transform

65

- Consider a point  $(x_i, y_i)$  in the  $xy$ -plane and the general equation of a straight line  $y_i = a x_i + b$
- Infinite number of lines pass through this point with different values of  $a$  and  $b$
- If we rewrite the line equation such that  $b = -a x_i + y_i$  and consider the  $ab$ -plane (the parameter space), then we have the line equation for a fixed pair  $(x_i, y_i)$
- If we consider another point  $(x_j, y_j)$  that lie on the same line as  $(x_i, y_i)$ , then in the  $ab$ -space the two lines corresponding to these two point will intersect at  $(a', b')$  and so do all the lines for the points on the line in the  $xy$ -plane



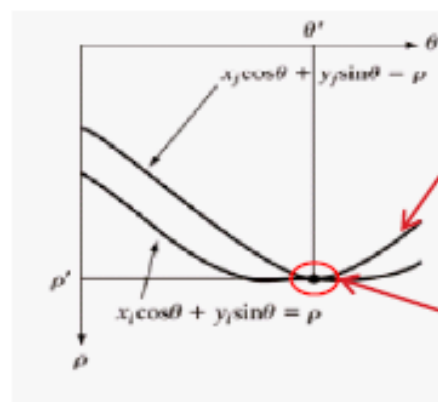
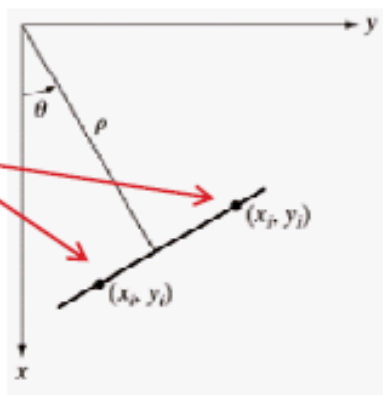
# Hough Transform

66

- In order to find the principle lines in the image, we can compute all the parameter-space lines and then *select those points that have large number of intersections*
- A major difficulty in this approach is that the *slope of the line approaches infinity for vertical and near vertical lines*
- One way around is to use the normal representation of the line in the xy-plane

$$x_i \cos \theta + y_i \sin \theta = \rho$$

Two points that lie on the same line



Representation of the two points in the parameter space

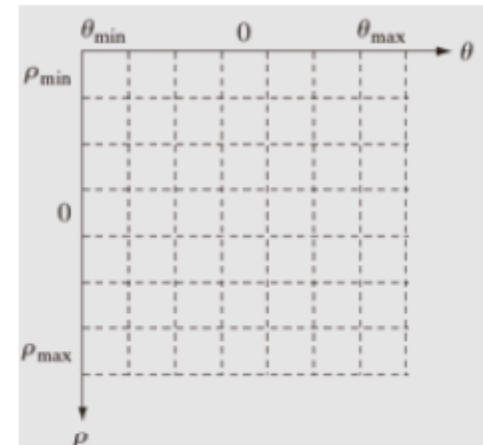
Intersection of the curves that represent the two points

- In this representation, a vertical line is represented in the parameter-space by  $\theta = 90$  and  $\rho$  being the y-intercept.
- Points that lie on the same line have their normal representation intersect at  $(\theta', \rho')$ . As the number of points that lie on the same line increases, we will have more curve crossing in the parameter space at the point  $(\theta', \rho')$

# Hough Transform

67

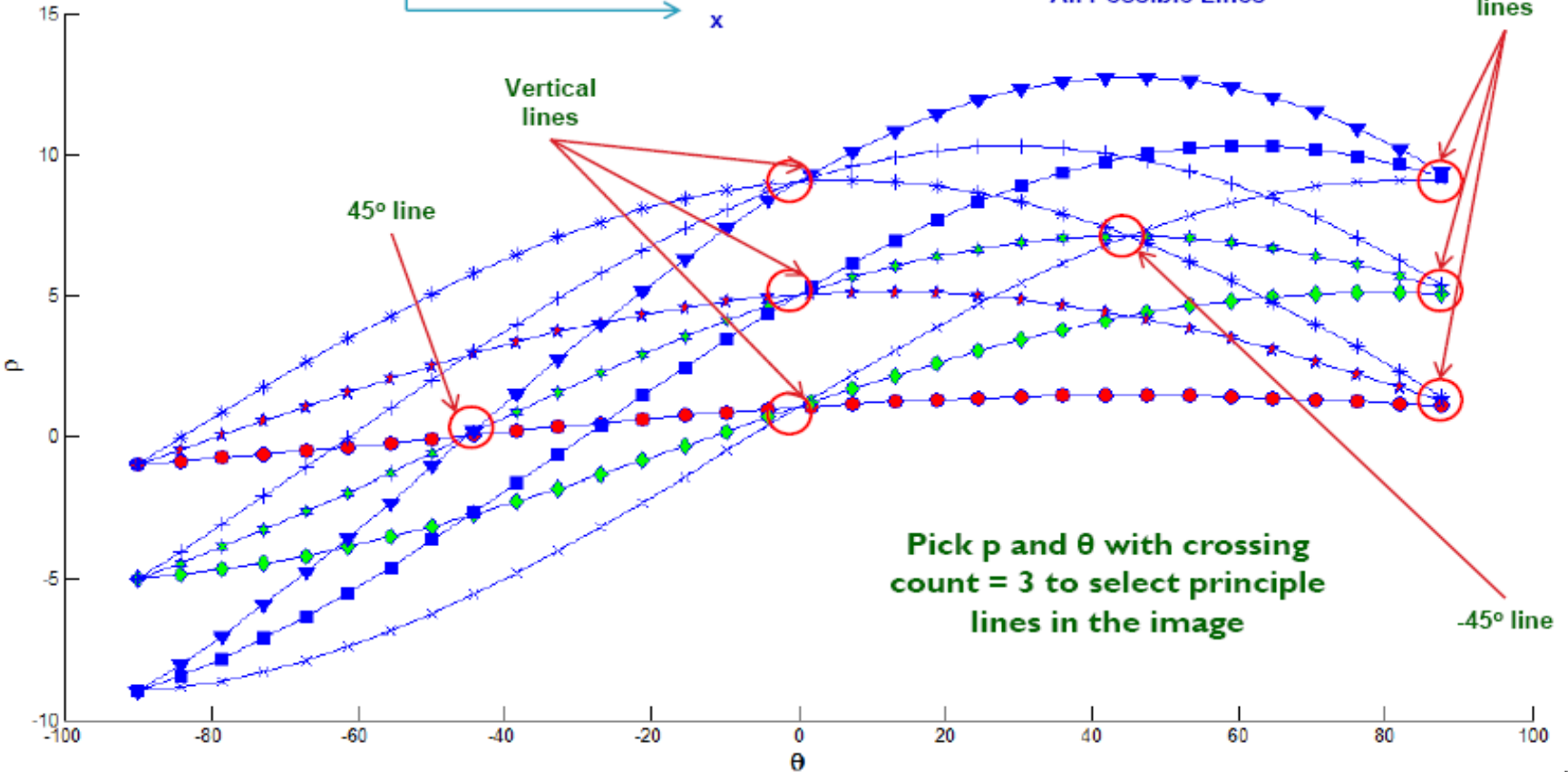
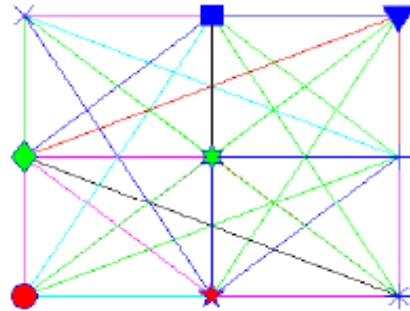
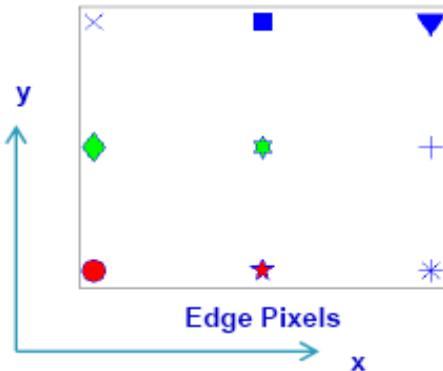
- **Procedure for finding principle lines in an image**
  1. Subdivide the  $p\theta$  parameter space into so-called accumulator cells and initialize them to zero. The range of each of the parameters is specified by
    - a.  $-90 \leq \theta \leq 90$
    - b.  $-D \leq p \leq D$  with  $D$  being the maximum diagonal length in the image
  2. For every edge/line candidate pixel  $(x_i, y_i)$  in the edge image, we substitute every possible value of  $\theta$  in  $p = x_i \cos \theta + y_i \sin \theta$ . Round the values of  $p$  to the allowed subdivision values on the  $p$ -axis
  3. If  $\theta_m$  results in  $p_n$ , then increment the count in cell  $(m, n)$  by one, i.e.  $A(m, n) = A(m, n) + 1$
  4. Select accumulator cells with counts greater than certain threshold and find the corresponding  $\theta$  and for these cells
  5. Draw the lines in the  $xy$ -plane for each pair of pair of  $\theta$  and  $p$  found in Step 4



Subdividing the parameter space to generate accumulator cells

# Hough Transform

• **Example**



# Hough Transform

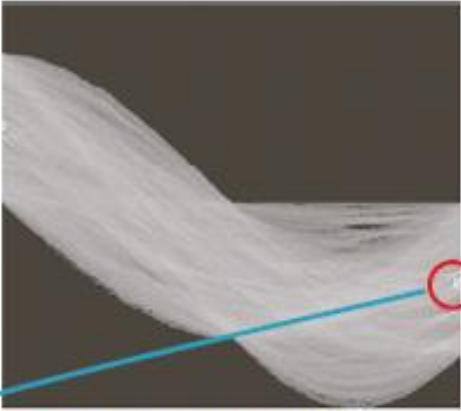
□ **Example:** We want to segment the two edges of the principle runway



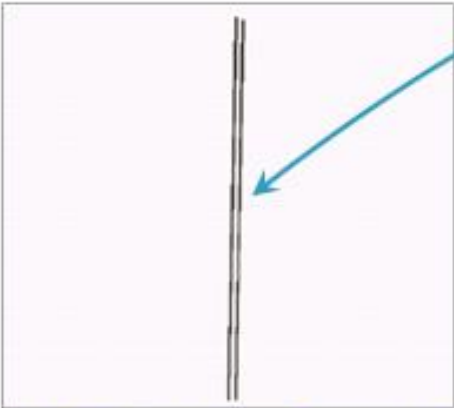
Image



Edge Image



Parameter Space  
Representation of Edge Points



The Two Principle Lines Detected after  
Thresholding the Parameter Space



The Two Principle Lines Superimposed on  
Original Image

# Are Edge Invariant to Transformations?

70

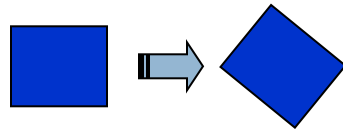
## □ Invariance:

- We want features to be detected despite geometric or photometric changes in the image: if we have two transformed versions of the same image, features should be detected in corresponding locations

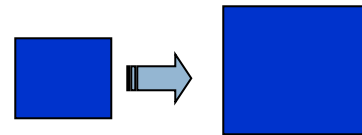
## □ Models of Image Change - Transformation

### □ Geometry

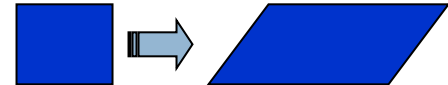
- Rotation



- Similarity (rotation + uniform scale)



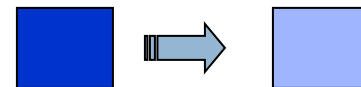
- Affine (scale dependent on direction)



valid for: orthographic camera, locally planar object

### □ Photometry

- Affine intensity change ( $I \rightarrow aI + b$ )



# Are Edges Invariant to Transformations?

71

- Edges are usually defined as sets of points in the image which have a strong gradient magnitude
- Edges can be invariant to brightness changes but typically not invariant to other transformations



# Local Features

- ▶ **Edge**
- ▶ **Texture Features**
  - ▶ **Corner**
  - ▶ **Interest Points**



# Texture Features: What's in the image?

73

- Texture is a tactile or visual characteristic of a surface.
- In general, color is usually a pixel property while texture can only be measured from a group of pixels.
- Aim: Texture gives us information about the spatial arrangement of the colors or intensities in an image.
- To find a unique way of representing the underlying characteristics of textures and represent them in some simpler but unique form, so then they can be used to accurately and robustly classify and segment objects.

# Texture Features

74

- Basically, texture representation methods can be classified into two categories:
  - ▣ **Structural approach:** Texture is a set of primitive texels in some regular or repeated relationship.
    - Texel: A small geometric pattern that is repeated frequently on some surface resulting in a texture.
    - Work well for man-made and regular patterns
  - ▣ **Statistical approach:** Texture is a quantitative measure of the arrangement of intensities in a region.
    - More general and easier to compute and is used more often in practice.

# Structural approach

75

- Structural approaches model texture as a set of texture primitives (also called texels (texture elements)) in a particular spatial relationship (also called lattice or grid layout).
- A structural description of a texture includes a description of the primitives and a specification of their placement patterns.
- The primitives must be identifiable and their relationships must be efficiently computable.

# Structural approach

76



A method that involves the detection of interest points, clustering of these points, voting for consistent lattice unit proposals, and iterative fitting of a lattice structure.

# Statistical approach

77

- Usually, segmenting out the texels is difficult or even impossible in real images.
- Instead, numeric quantities or statistics that describe a texture can be computed from the gray tones or colors themselves.
- Statistical methods analyze the spatial distribution of gray values, by computing local features at each point in the image, and deriving a set of statistics from the distributions of the local features.
- This approach can be less intuitive, but is computationally efficient and often works well.

# Statistical approach

78

- Depending on the number of pixels defining the local feature statistical methods can be further classified into
  - ▣ First-order (one pixel)
  - ▣ Second-order (two pixels)
  - ▣ Higher-order (three or more pixels) statistics.
  
- The basic difference is that:
  - ▣ First-order statistics estimate properties (e.g. average and variance) of individual pixel values, ignoring the spatial interaction between image pixels,
  - ▣ Second- and higher-order statistics estimate properties of two or more pixel values occurring at specific locations relative to each other.

# Some Statistical Approaches

79

- Some statistical approaches for texture:
  - ▣ **Corner Detection**
  - ▣ Co- occurrence matrices
  - ▣ Local binary patterns
  - ▣ Statistical moments
  - ▣ Autocorrelation
  - ▣ Markov random fields
  - ▣ Autoregressive models
  - ▣ Mathematical morphology
  - ▣ **Interest points – SIFT, SURF...**
  - ▣ Fourier power spectrum
  - ▣ Gabor filters

# Texture Features - Corner

80

- A corner can be defined as the intersection of two edges.
- Can also be defined as a point for which there are two dominant and different edge directions in a local neighborhood of the point.
- Corner detection is frequently used in **motion detection, image registration, video tracking, image matching, and object recognition.**
- Edge detection that can be used with post-processing to detect corners
  - ▣ Kirsch operator
  - ▣ Frei-Chen masking set.



# Corner Detection

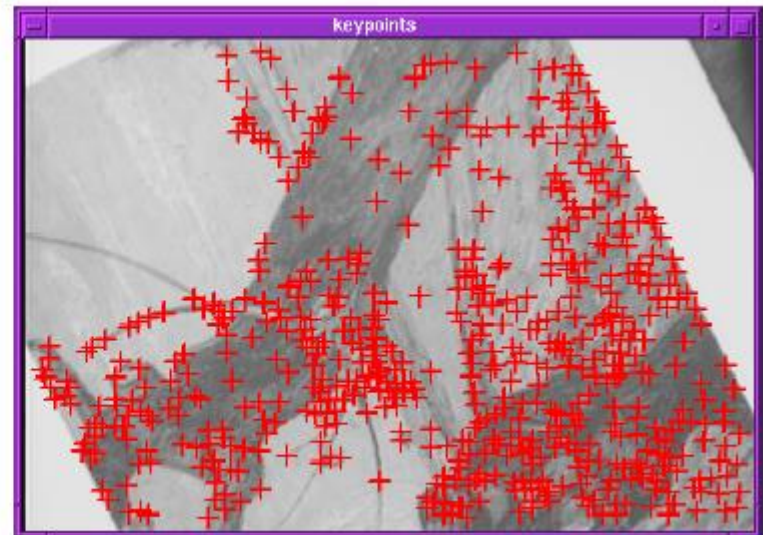
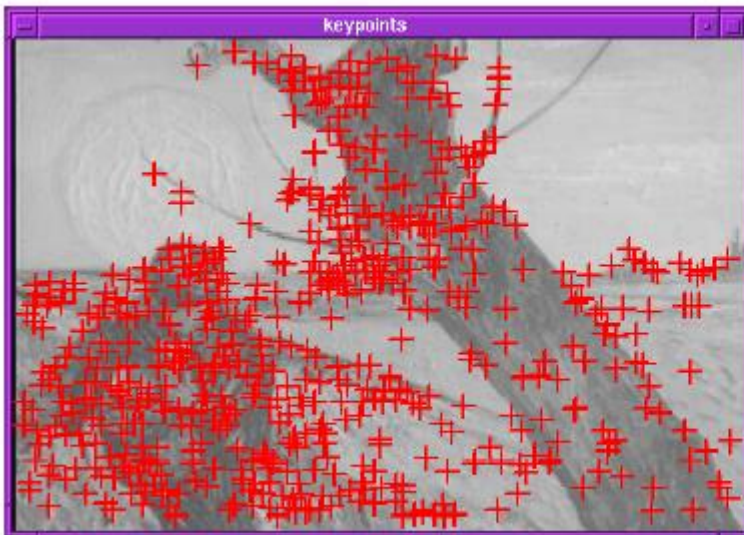
81

- Several proposed approaches for corner detection:
  - Moravec corner detection algorithm
  - **The Harris & Stephens corner detection algorithms**
  - The level curve curvature approach
  - Laplacian of Gaussian, differences of Gaussians and determinant of the Hessian scale-space interest points
  - The Wang and Brady corner detection algorithm
  - The SUSAN corner detector
  - .....
- One determination of the quality of a corner detector is its ability to detect the same corner in multiple similar images, under conditions of **different lighting, translation, rotation, Scaling, and other transforms.**

# Finding Corners

82

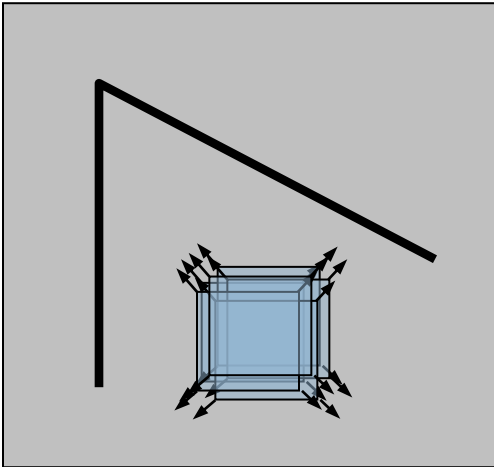
- Key property: in the region around a corner, image gradient has two or more dominant directions
  - ▣ Idea:
    - Exactly at a corner, gradient is ill defined.
    - However, in the region around a corner, gradient has two or more different values.
- Corners are repeatable and distinctive.
- Edge detectors perform poorly at corners



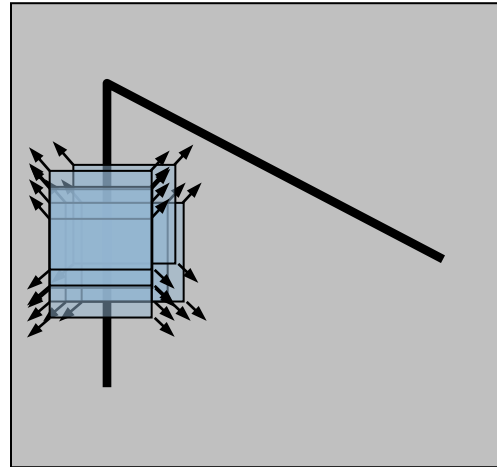
# Harris Detector - The Basic Idea

83

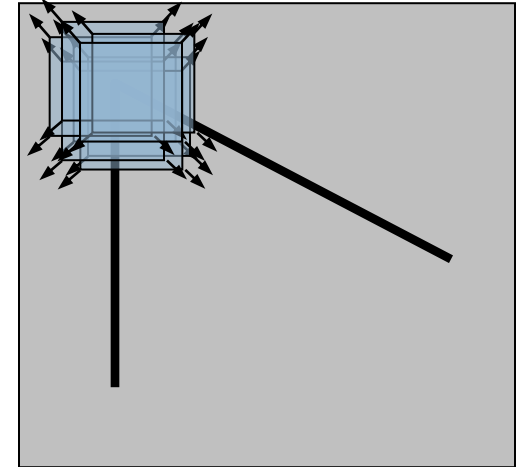
- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



“flat” region:  
no change in  
all directions



“edge”:  
no change along  
the edge direction



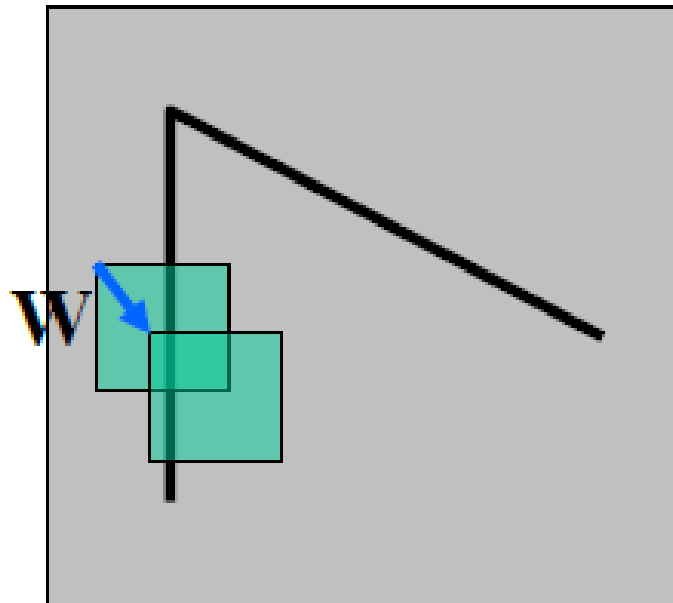
“corner”:  
significant change  
in all directions

Find locations such that the minimum change caused by shifting the window in any direction is large

# Harris Detector - The Basic Idea

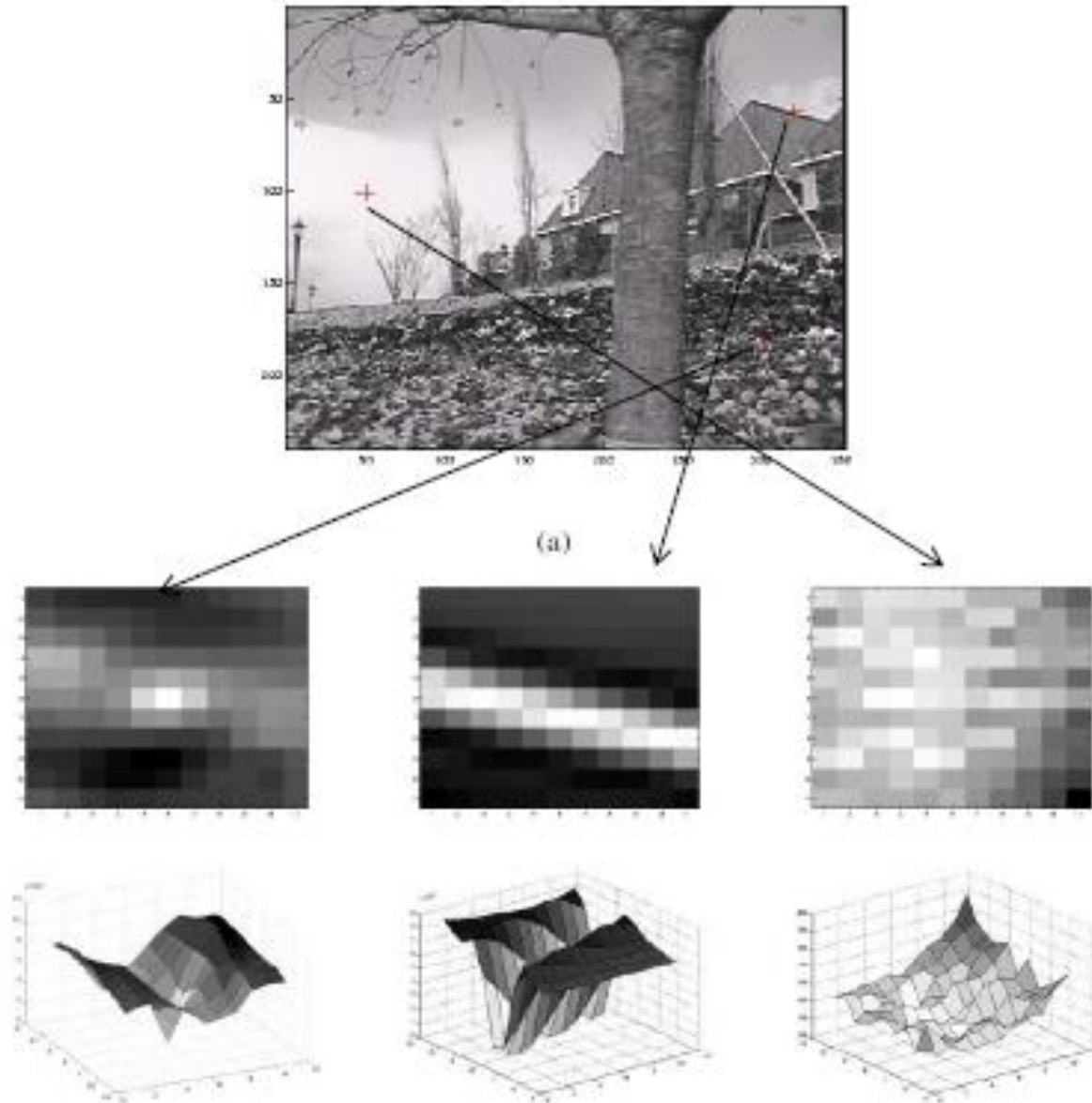
84

- Consider shifting the window  $\mathbf{W}$  by  $(u,v)$ 
  - ▣ How do the pixels in  $\mathbf{W}$  change?
  - ▣ Compare each pixel before and after using the sum of squared differences (SSD)
  - ▣ This defines an SSD “error”  $E(u,v)$ :



# Sum of Squared Differences (SSD) Profile

85



# Harris Detector: Step 1 - Compute the Gradient

- Change of intensity for the shift  $[u, v]$ :

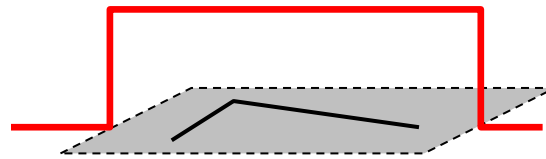
$$E(u, v) = \sum_{x, y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$

Window function

Shifted intensity

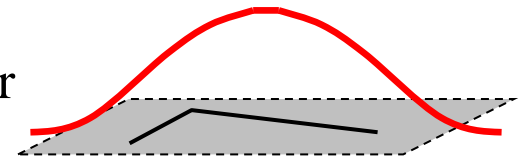
Intensity

Window function  $w(x, y) =$



1 in window, 0 outside

or



Gaussian

# Taylor Series Representation

87

**Taylor series** is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

Express  $I(x + u, y + v)$  at  $(x, y)$ :

$$I(x + u, y + v) = I(x, y) + I_x(x + u - x) + I_y(y + v - y)$$

$$I(x + u, y + v) = I(x, y) + I_x u + I_y v$$

# Harris Detector: Step 1 - Compute the Gradient

## ➤ Small motion assumption

Taylor Series expansion of  $I$ :

$$I(x+u, y+v) = I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \text{higher order terms}$$

If the motion  $(u, v)$  is small, then first order approx. is good

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v$$

$$\approx I(x, y) + [I_x \ I_y] \begin{bmatrix} u \\ v \end{bmatrix}$$

shorthand:  $I_x = \frac{\partial I}{\partial x}$

Plugging this into the formula on the previous slide...



# Harris Detector: Step 1 - Compute the Gradient

89

$$E(u, v) = \sum_{x, y} \left[ \underbrace{I(x+u, y+v)}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}} \right]^2$$

$$E(u, v) = \sum_{x, y} \left[ \underbrace{I(x, y) + uI_x + vI_y}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}} \right]^2$$

Taylor Series

$$E(u, v) = \sum_{x, y} [uI_x + vI_y]^2$$

$$E(u, v) = \sum_{x, y} \left[ (u \quad v) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \right]^2$$

$$M = \sum_{x, y} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

$$E(u, v) = \sum_{x, y} (u \quad v) \begin{pmatrix} I_x \\ I_y \end{pmatrix} (I_x \quad I_y) \begin{pmatrix} u \\ v \end{pmatrix}$$

$$E(u, v) = (u \quad v) \left[ \sum_{x, y} \begin{pmatrix} I_x \\ I_y \end{pmatrix} (I_x \quad I_y) \right] \begin{pmatrix} u \\ v \end{pmatrix}$$

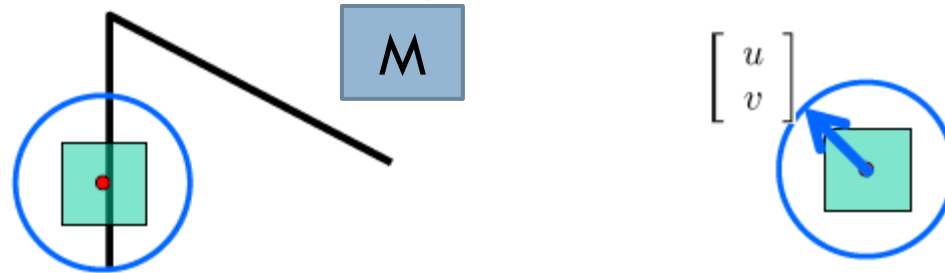
$$E(u, v) = (u \quad v) M \begin{pmatrix} u \\ v \end{pmatrix}$$

# Harris Detector: Step 2 - Compute the Eigenvalues

90

- This can be rewritten:

$$E(u, v) \approx [u \ v] \left( \sum_{(x,y) \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$



- For the example above:

- ▣ You can move the center of the green window to anywhere on the blue unit circle
- ▣ How do we find directions that will result in the largest and smallest  $E$  values?
- ▣ Find these directions by looking at the eigenvectors of  $M$

# Harris Detector: Step 2 - Compute the Eigenvalues

91

- Interpreting the second moment matrix
  - ▣ First, consider the axis-aligned case (gradients are either horizontal or vertical)

$$M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

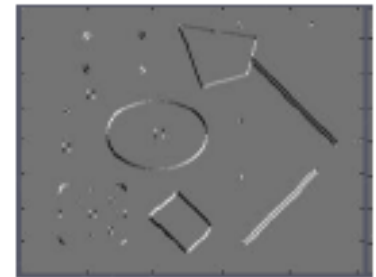
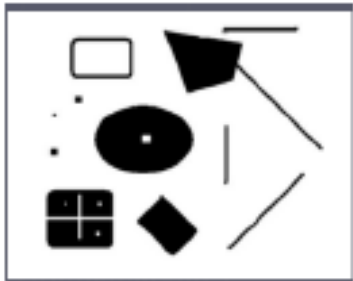
- ▣ If either  $\lambda$  is close to 0, then this is **not** a corner, so look for locations where both are large.
- ▣ If there are two large eigenvalues there is a corner; and if one an edge

# Harris Detector: Step 2 - Compute the Eigenvalues

92

$$M = \sum w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$$

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).



Notation:

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

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

$$I_x I_y \Leftrightarrow \frac{\partial I}{\partial x} \frac{\partial I}{\partial y}$$

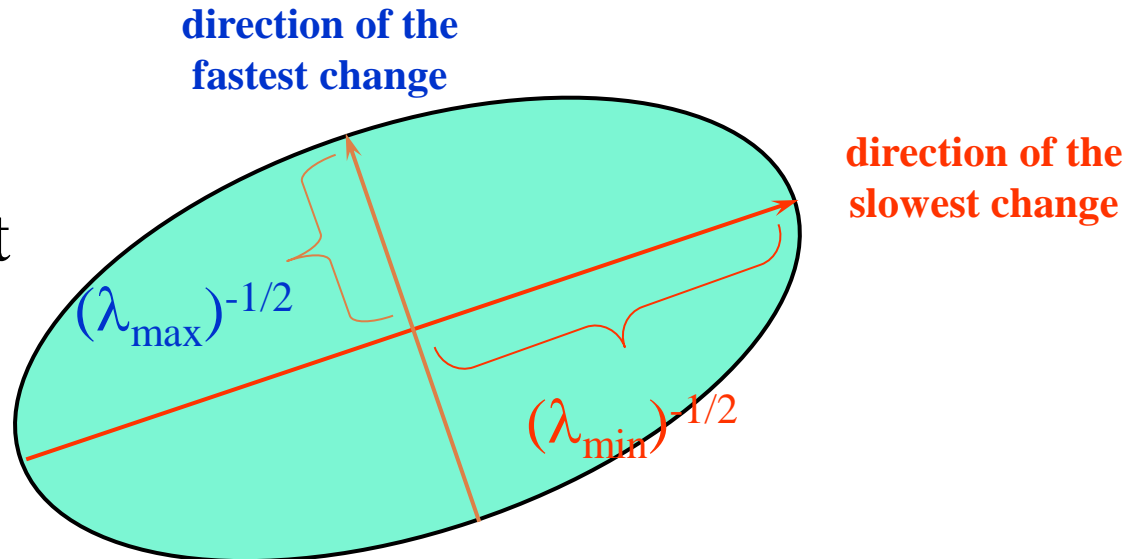
# Harris Detector: Step 2 - Compute the Eigenvalues

- Intensity change in shifting window: eigenvalue analysis

$$E(u, v) \cong [u, v] M \begin{bmatrix} u \\ v \end{bmatrix} \quad \lambda_1, \lambda_2 \text{ eigenvalues of } M$$

- Since  $M$  is symmetric, we have  $M = R^{-1} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$
- We can visualize  $M$  as an ellipse with axis lengths determined by the eigenvalues and orientation determined by  $R$

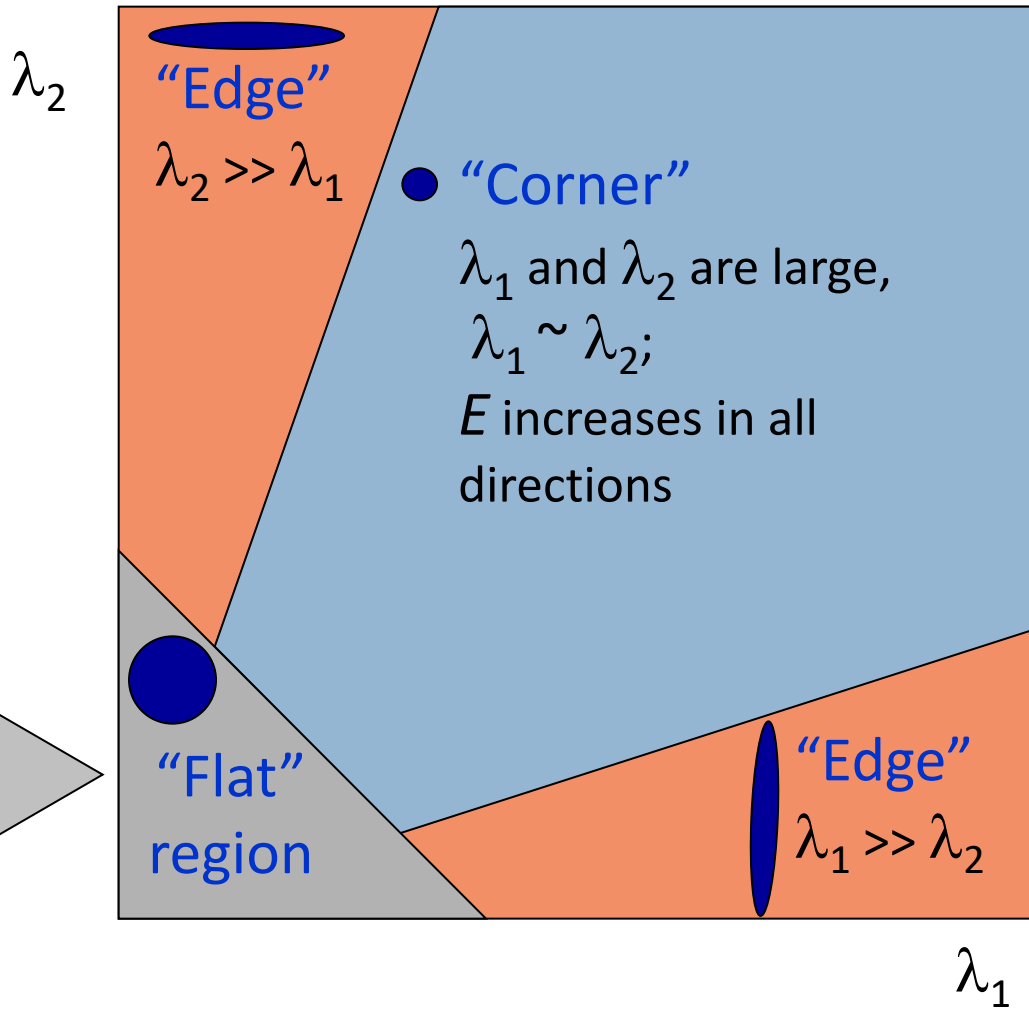
Ellipse  $E(u, v) = \text{const}$



# Harris Detector: Step 3 Classification of Eigenvalues

Classification of image points using eigenvalues of  $M$ :

$\lambda_1$  and  $\lambda_2$  are small;  
 $E$  is almost constant  
in all directions



# Harris Detector: Step 4 Measure Corner Response

- Measure of corner response:

$$R = \det M - k (\text{trace } M)^2$$

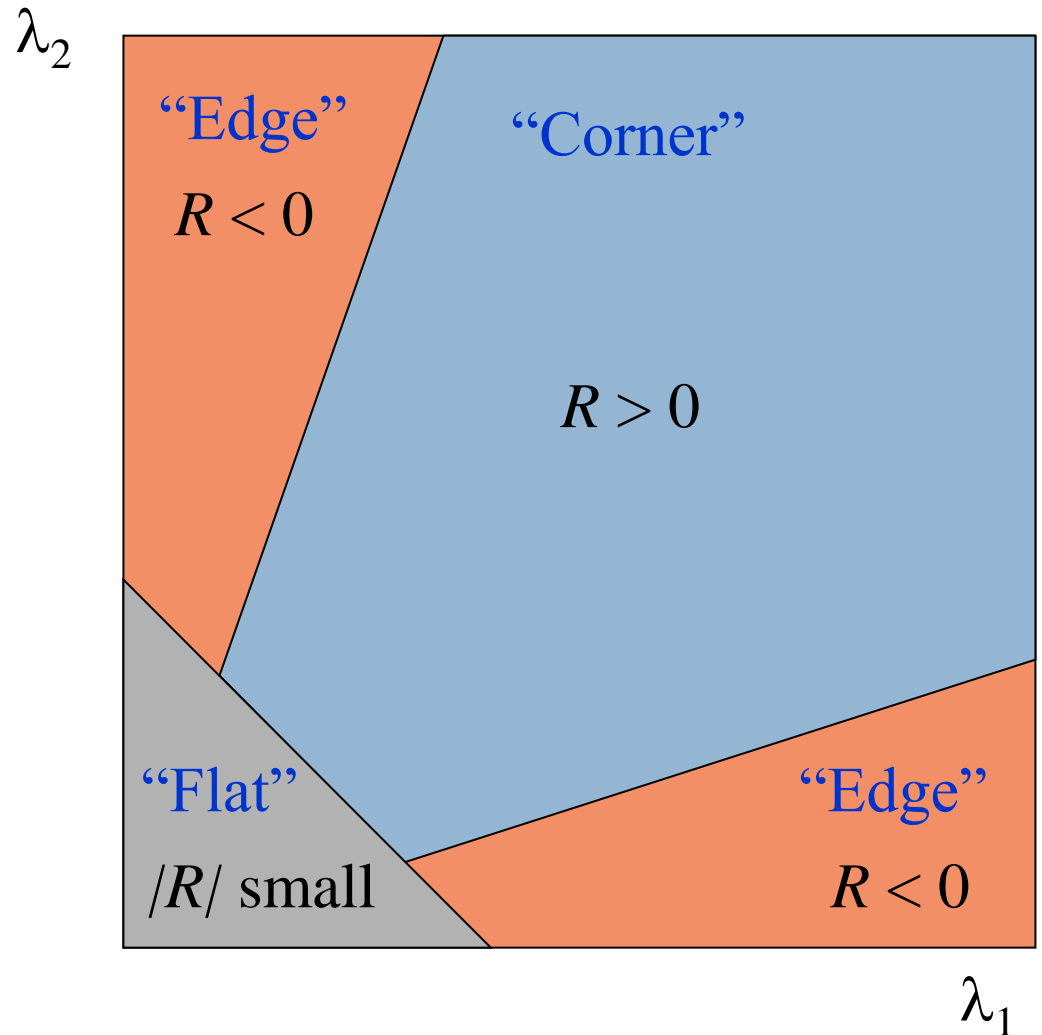
$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

( $k$  – empirical constant,  $k = 0.04-0.06$ )

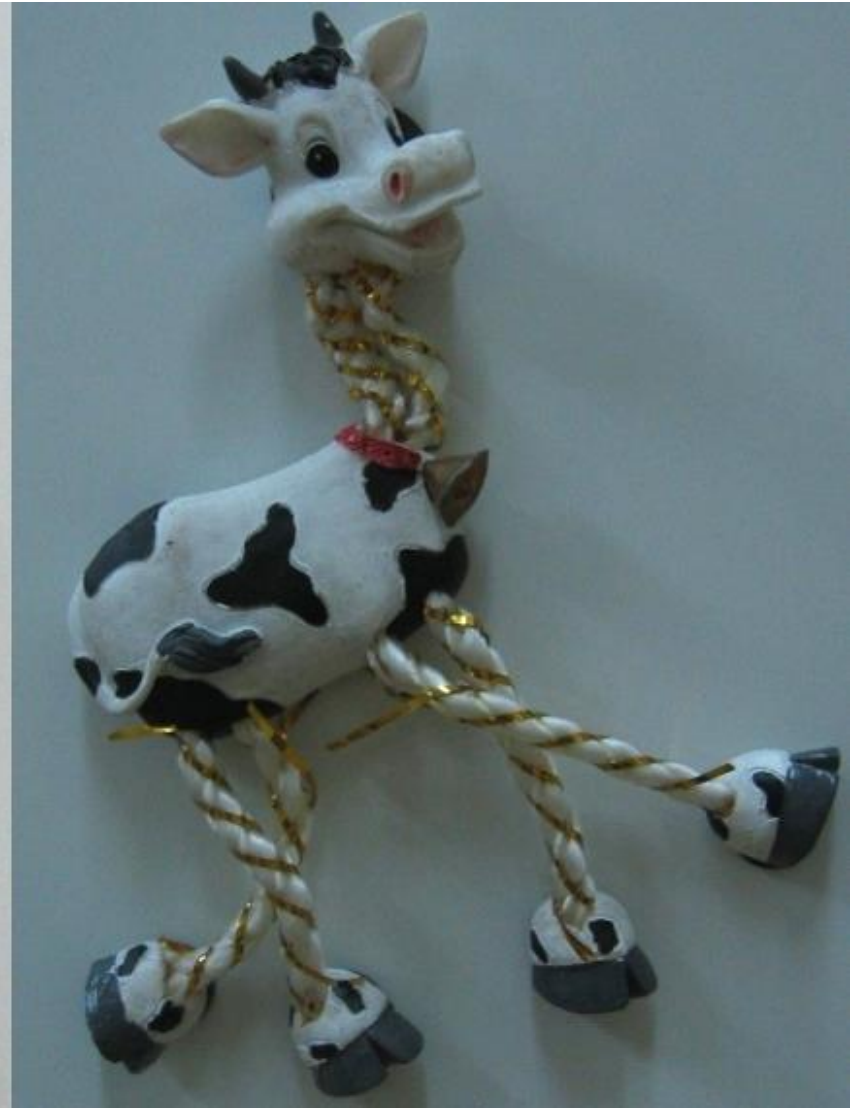
# Harris Detector: Step 4 Measure Corner Response

- $R$  depends only on eigenvalues of  $M$
- $R$  is large for a **corner**
- $R$  is negative with large magnitude for an **edge**
- $|R|$  is small for a **flat** region

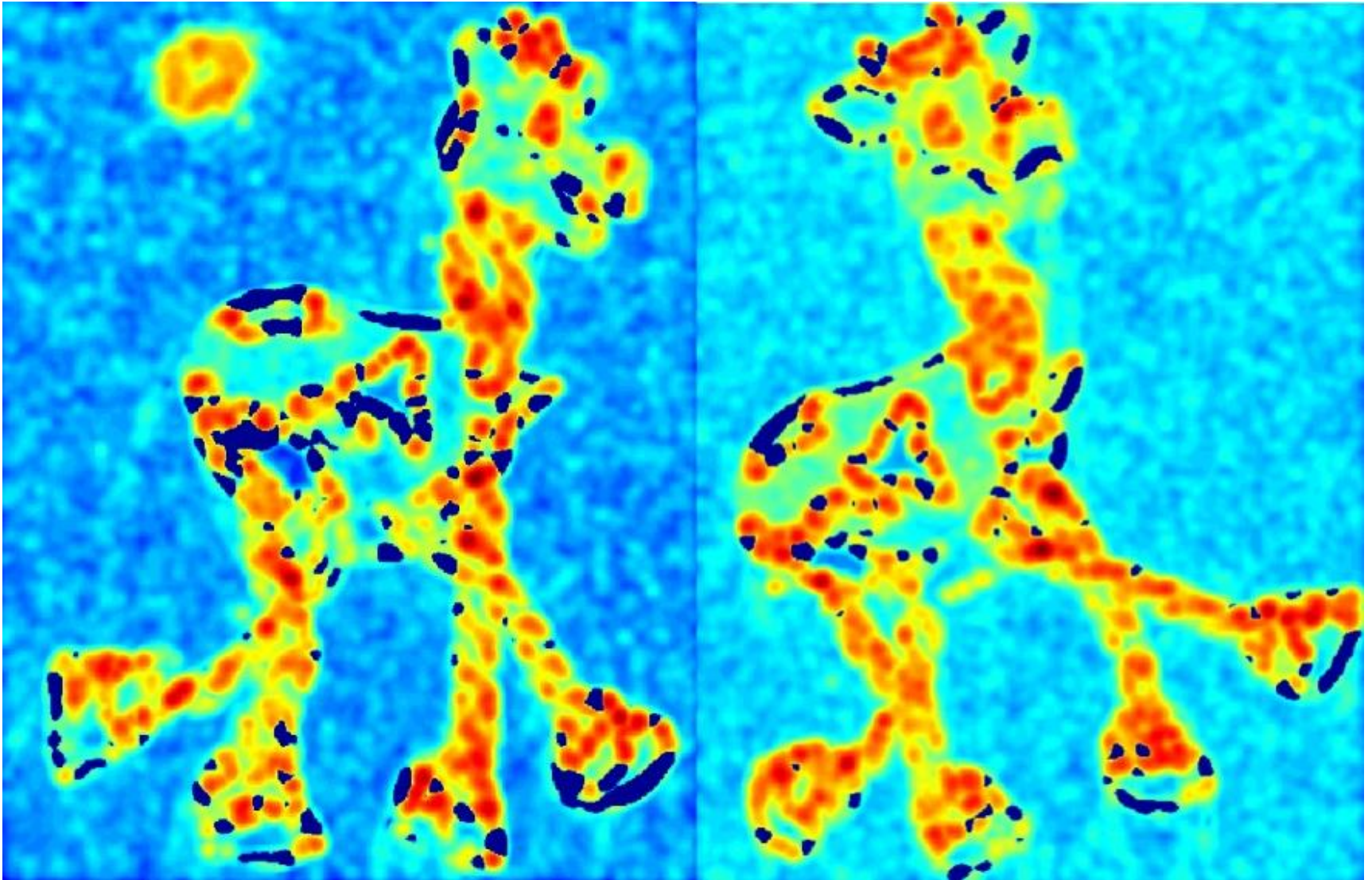




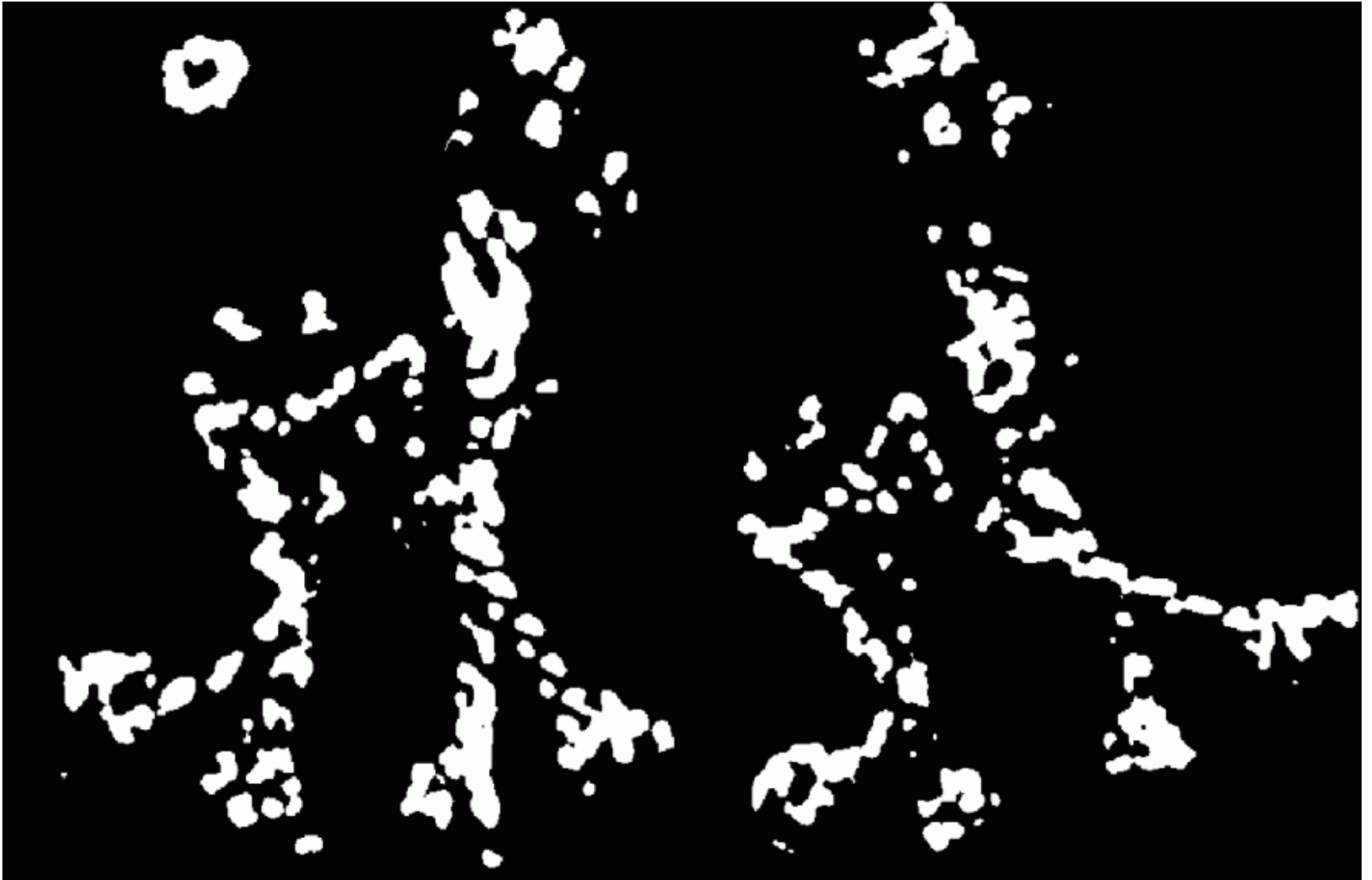
# Harris Detector: Workflow



# Harris Detector: Compute corner response $R$



**Harris Detector: Find points with large corner  
response:  $R > \text{threshold}$**



# Harris Detector: Take only the points of local maxima of $R$



# Harris Detector: Workflow



# Harris Detector: Summary

- Compute Gaussian derivatives at each pixel
- Compute second moment matrix  $M$  in a Gaussian window around each pixel
- Compute corner response function  $R$
- Threshold  $R$
- Find local maxima of response function (nonmaximum suppression)

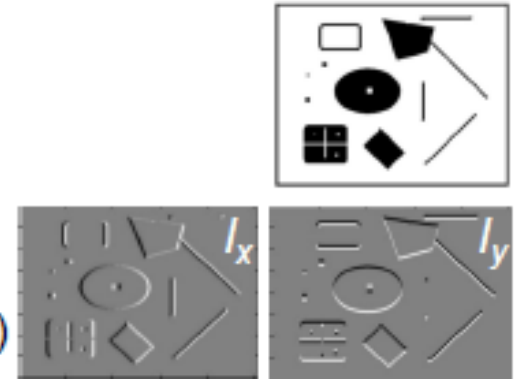
# Harris Detector: Summary

103

- Second moment matrix

$$\mu(\sigma_I, \sigma_D) = g(\sigma_I) * \begin{bmatrix} I_x^2(\sigma_D) & I_x I_y(\sigma_D) \\ I_x I_y(\sigma_D) & I_y^2(\sigma_D) \end{bmatrix}$$

1. Image derivatives  
(optionally, blur first)



2. Square of derivatives



3. Gaussian filter  $g(\sigma)$



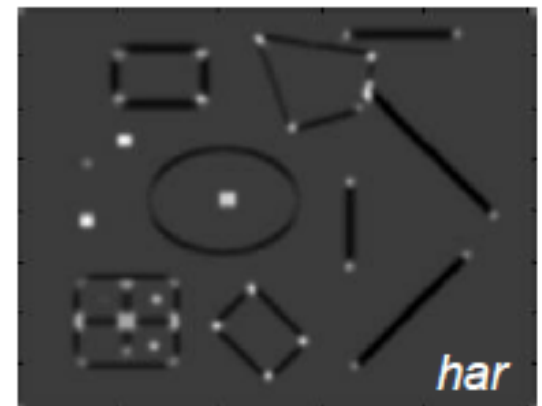
$$\det M = \lambda_1 \lambda_2$$

$$\text{trace } M = \lambda_1 + \lambda_2$$

- 4. Cornerness function – both eigenvalues are strong

$$\text{har} = \det[\mu(\sigma_I, \sigma_D)] - \alpha [\text{trace}(\mu(\sigma_I, \sigma_D))]^2 = g(I_x^2)g(I_y^2) - [g(I_x I_y)]^2 - \alpha [g(I_x^2) + g(I_y^2)]^2$$

- 5. Non-maxima suppression



# Other Version of Harris Detectors

104

$$R = \lambda_1 - \alpha \lambda_2$$

**Triggs**

$$R = \frac{\det(D)}{\text{trace}(D)} = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}$$

**Szeliski** (Harmonic mean)

$$R = \lambda_1$$

**Shi-Tomasi**



# Are Corners Invariant to Transformations?

105

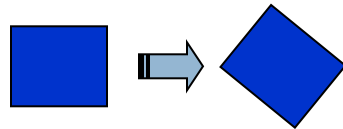
## □ Invariance:

- We want features to be detected despite geometric or photometric changes in the image: if we have two transformed versions of the same image, features should be detected in corresponding locations

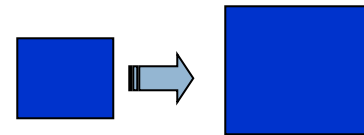
## □ Models of Image Change - Transformation

### □ Geometry

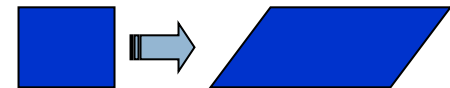
- Rotation



- Similarity (rotation + uniform scale)



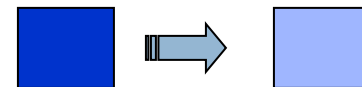
- Affine (scale dependent on direction)



valid for: orthographic camera, locally planar object

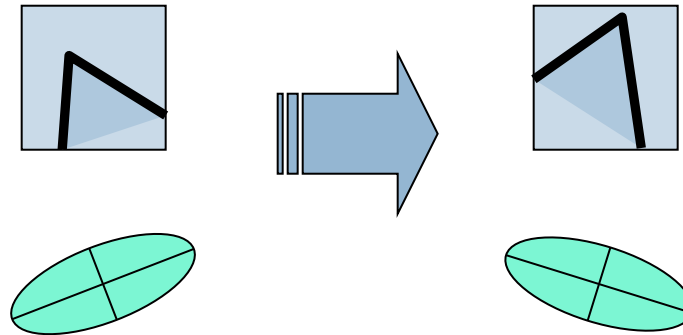
### □ Photometry

- Affine intensity change ( $I \rightarrow aI + b$ )



# Harris Detector: Some Properties

- Rotation invariance



Ellipse rotates but its shape (i.e. eigenvalues) remains the same

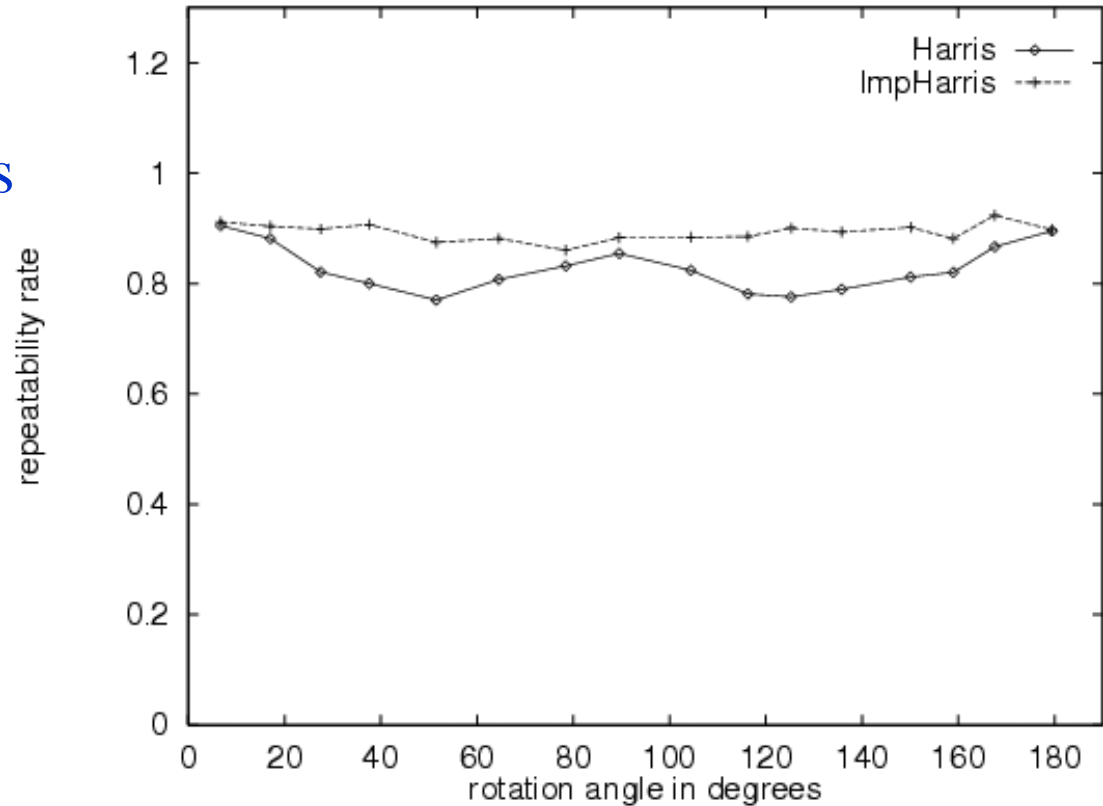
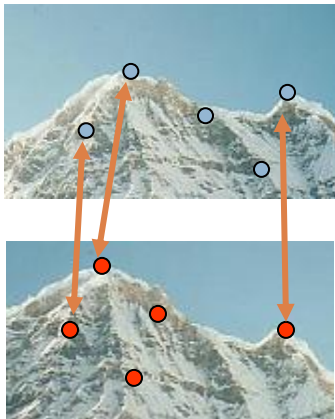
*Corner response  $R$  is invariant to image rotation*

# Harris Detector: Some Properties

## □ Rotation Invariant Detection

Repeatability rate:

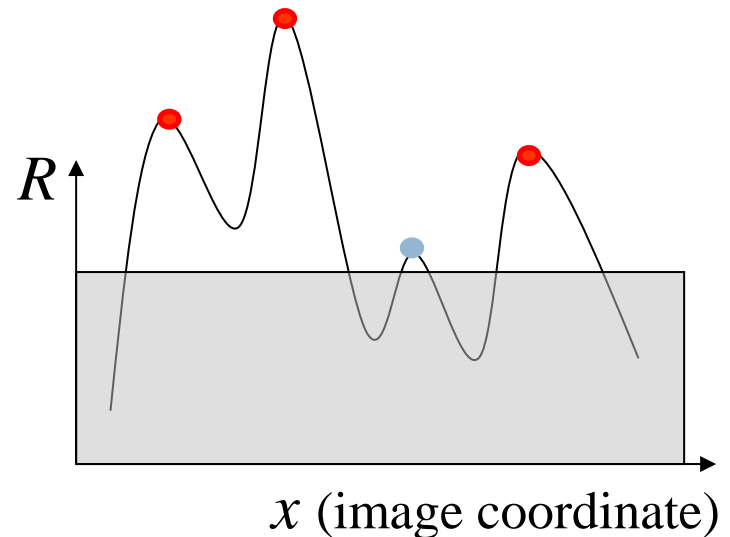
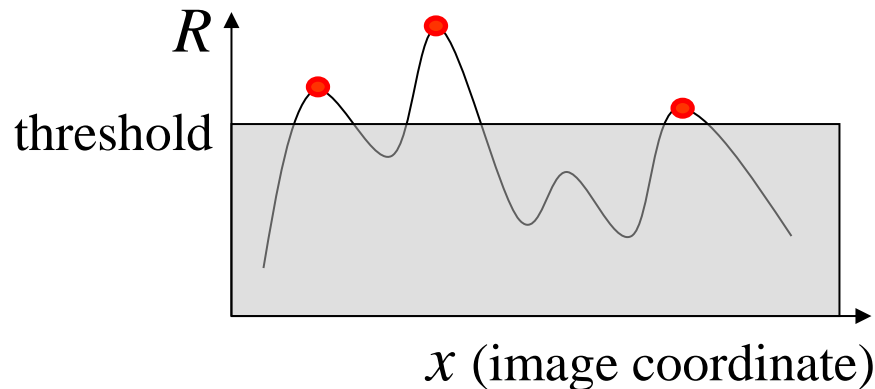
$$\frac{\# \text{ correspondences}}{\# \text{ possible correspondences}}$$



# Harris Detector: Some Properties

## □ Partial invariance to *affine intensity* change

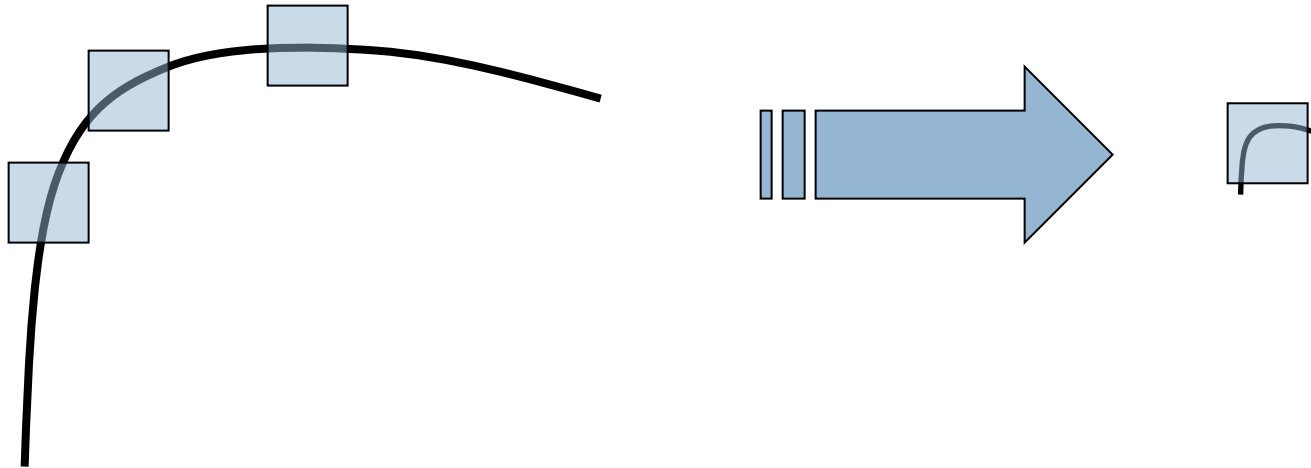
- ✓ Only derivatives are used  $\Rightarrow$  invariance to intensity shift  $I \rightarrow I + b$
- ✓ Intensity scale:  $I \rightarrow a I$



***Partially invariant to affine intensity change***

# Harris Detector: Some Properties

- But: non-invariant to *image scale*!



All points will be  
classified as *edges*

**Corner !**

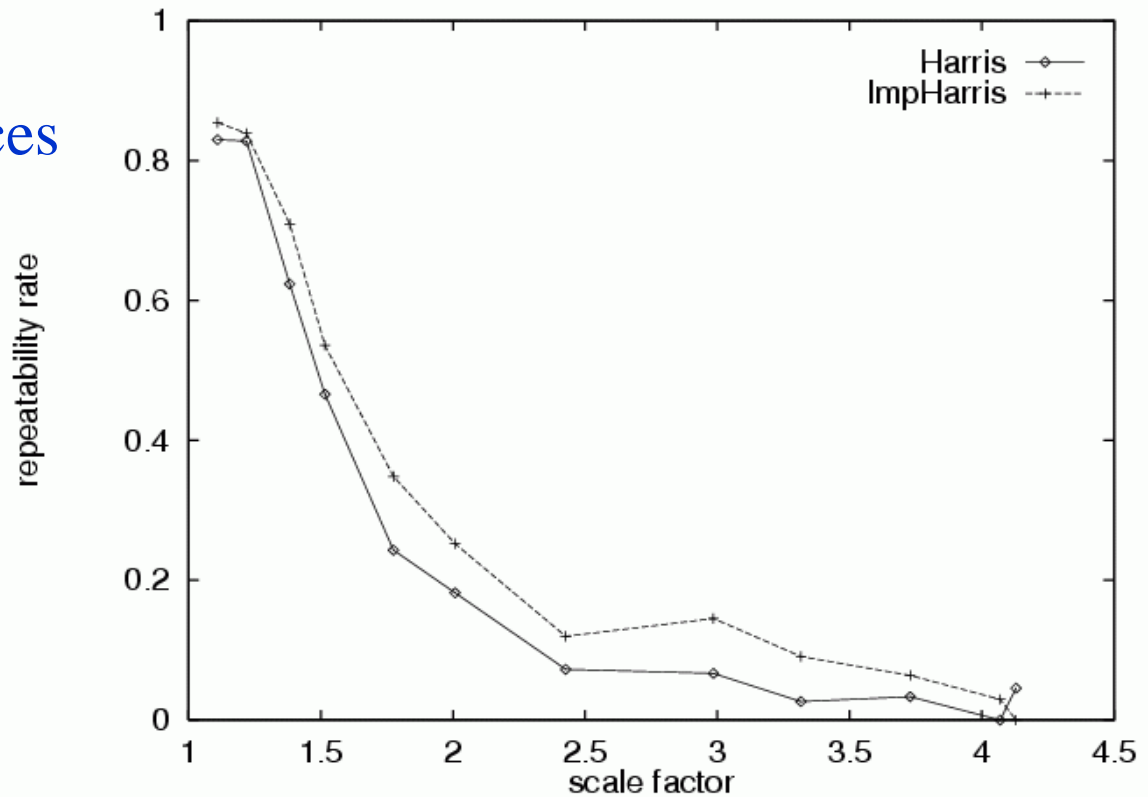
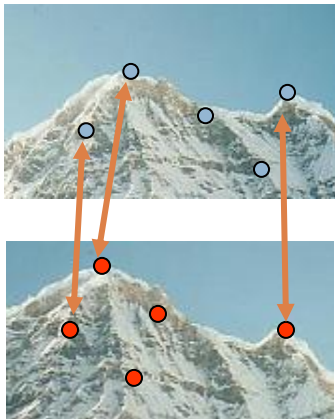
***Not invariant to scaling***

# Harris Detector: Some Properties

- Quality of Harris detector for different scale changes

Repeatability rate:

$$\frac{\# \text{ correspondences}}{\# \text{ possible correspondences}}$$



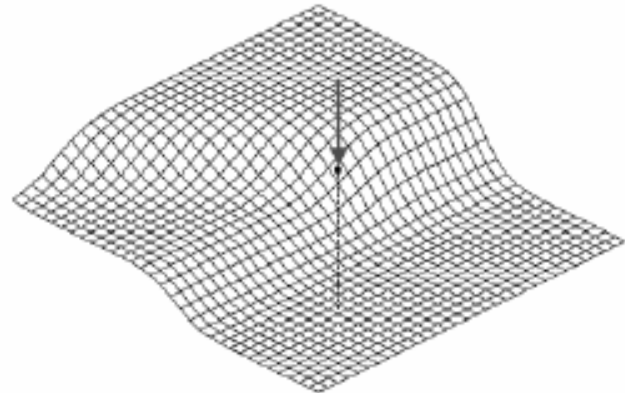
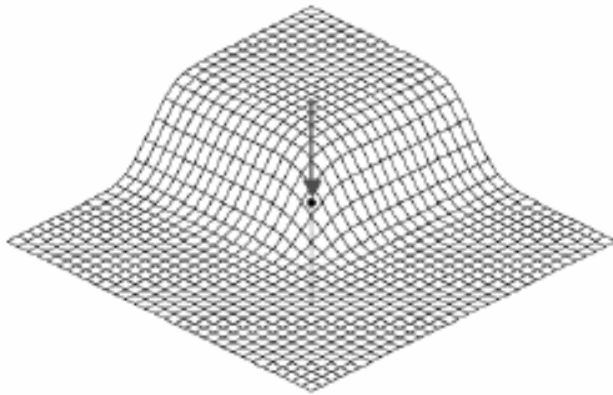
# Local Features

- ▶ Edge
- ▶ Corner
- ▶ **Interest Points**

# Texture extraction by Interest Points

112

- What is an interest point
  - ▣ Expressive texture
    - The point at which the direction of the boundary of object changes abruptly
    - Intersection point between two or more edge segments
  - ▣ Goal: Detect points that are *repeatable* and *distinctive*





# Properties of Interest Point Detectors

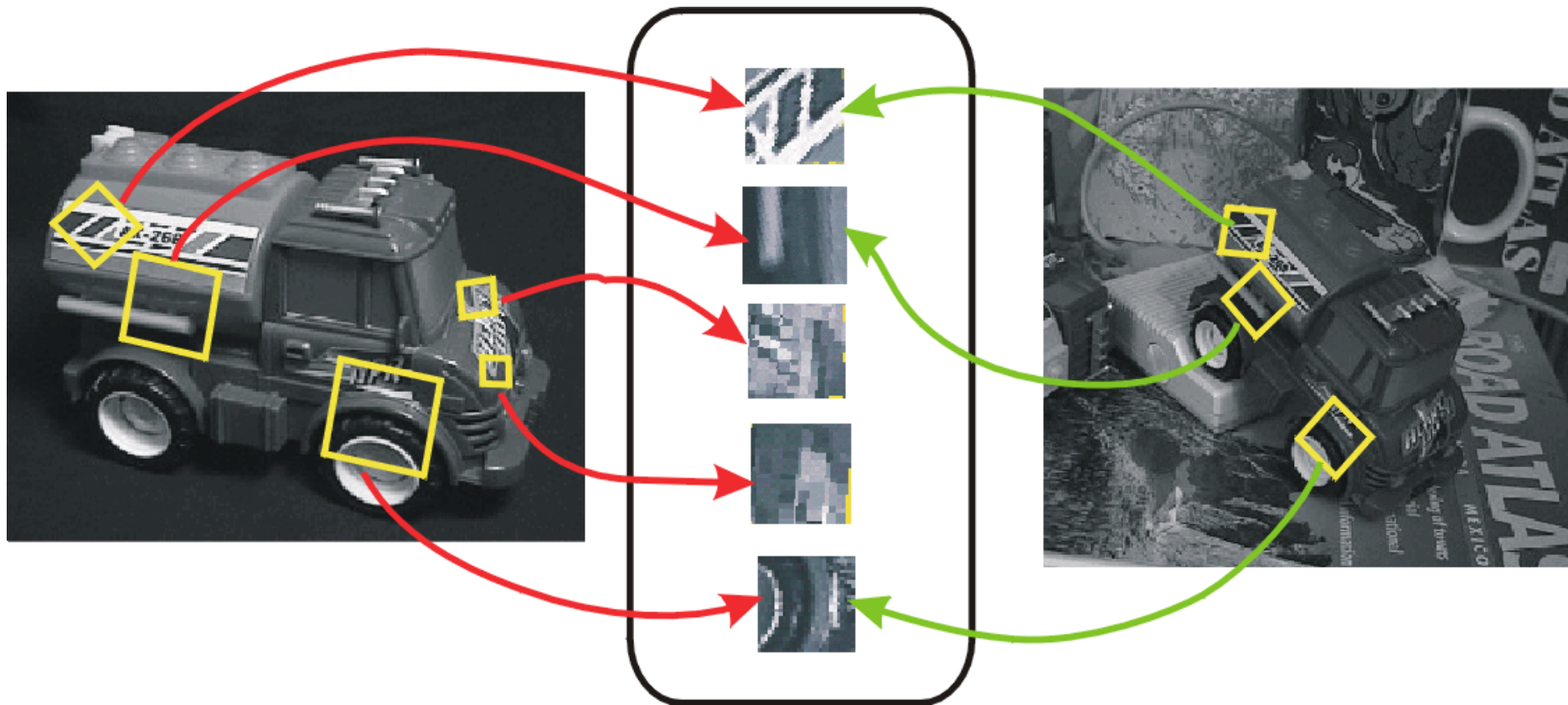
113

- Detect all (or most) true interest points
- No false interest points
- Well localized.
- Robust with respect to noise.
- Efficient detection
- Invariant to transformation

# Interest Point Detection Idea

114

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



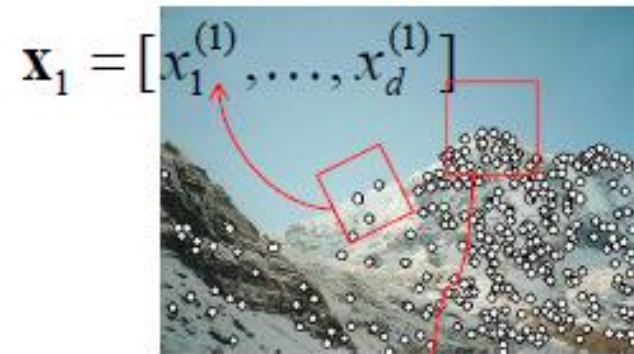
# Local features extraction: main components

115

1. **Detection:** Identify the interest points



2. **Description:** Extract feature vector descriptor surrounding each interest point.



3. **Matching:** Determine correspondence between descriptors in two views

$$\mathbf{x}_2 = [x_1^{(2)}, \dots, x_d^{(2)}]$$



# Key trade-offs

116

## Detection of interest points



**More Repeatable**

Robust detection  
Precise localization

**More Points**

Robust to occlusion  
Works with less texture

## Description of patches



**More Distinctive**

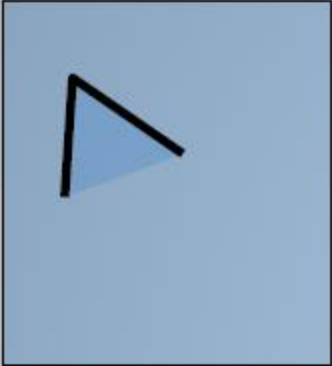
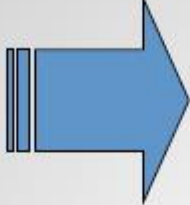
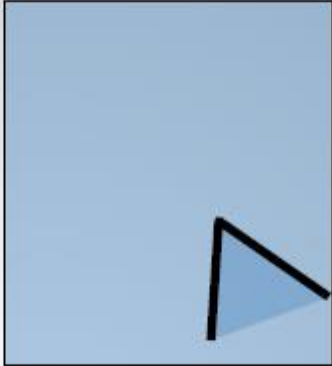


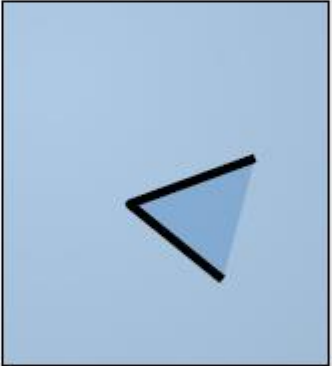
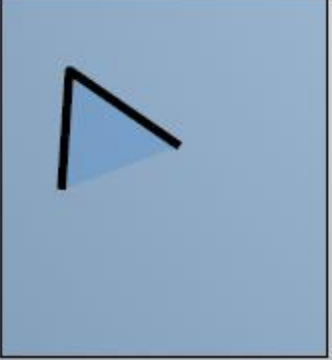

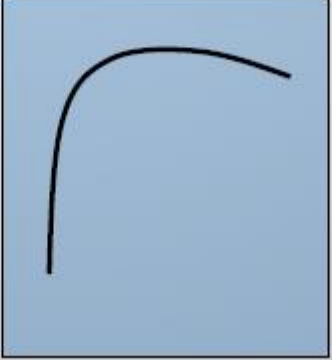
Minimize wrong matches

**More Flexible**

Robust to expected variations  
Maximize correct matches

# Corner as an Interest Point

117

			<p>Corners are invariant to</p> <p>translation ✓</p>
			<p>rotation ✓</p>
			<p>scaling ✗</p>

# Scale Space

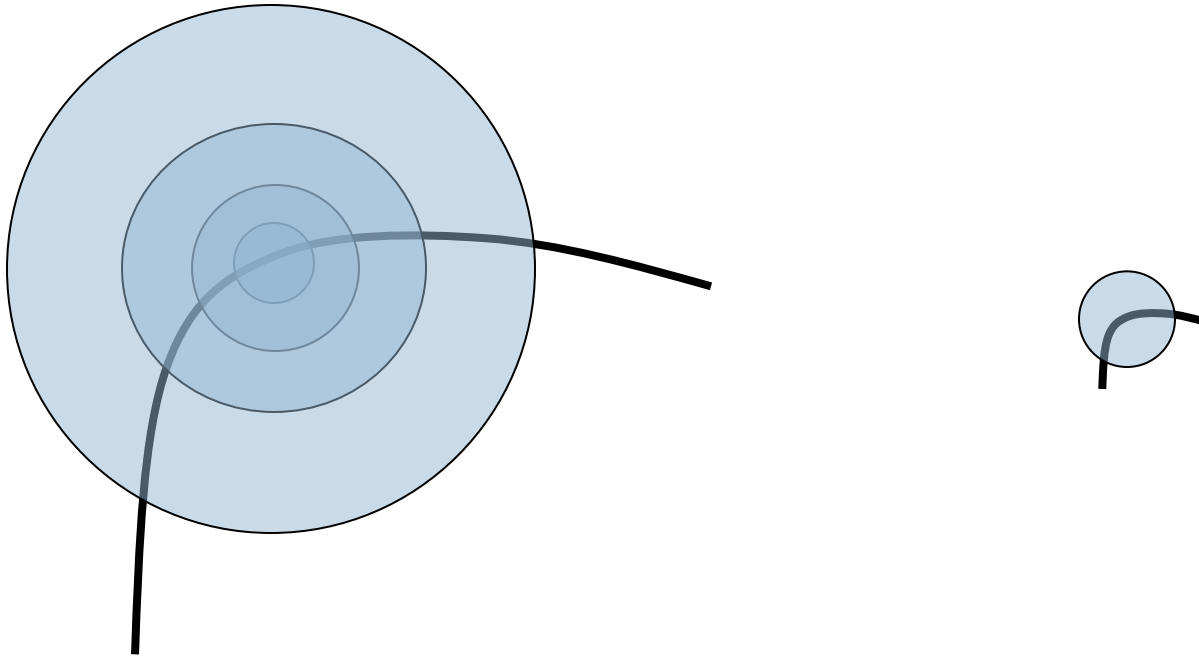
118

- The concept of scale is essential when computing features and descriptors from image data.
- Real-world objects may contain different types of structures at different scales and may therefore appear in different ways depending on the scale of observation.
- When observing objects by a camera or an eye, there is an additional scale problem due to perspective effects, implying that distant objects will appear smaller than nearby objects.
- A vision system intended to operate autonomously on image data acquired from a complex environment must therefore be able to handle and be robust to such scale variations.

# Scale Invariant Detection

119

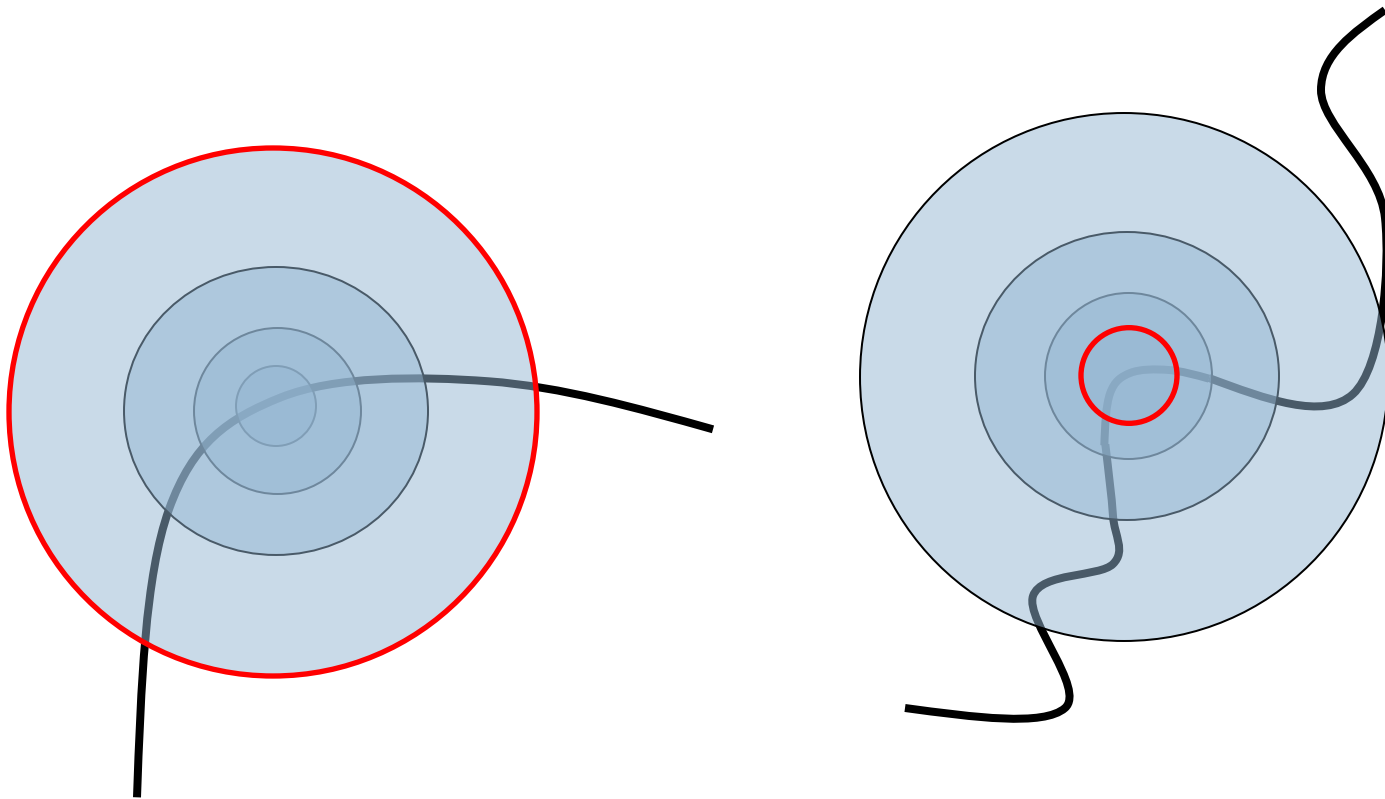
- Consider regions (e.g. circles) of different sizes around a point
- Regions of corresponding sizes will look the same in both images



# Scale Invariant Detection

120

- The problem: how do we choose corresponding circles *independently* in each image?



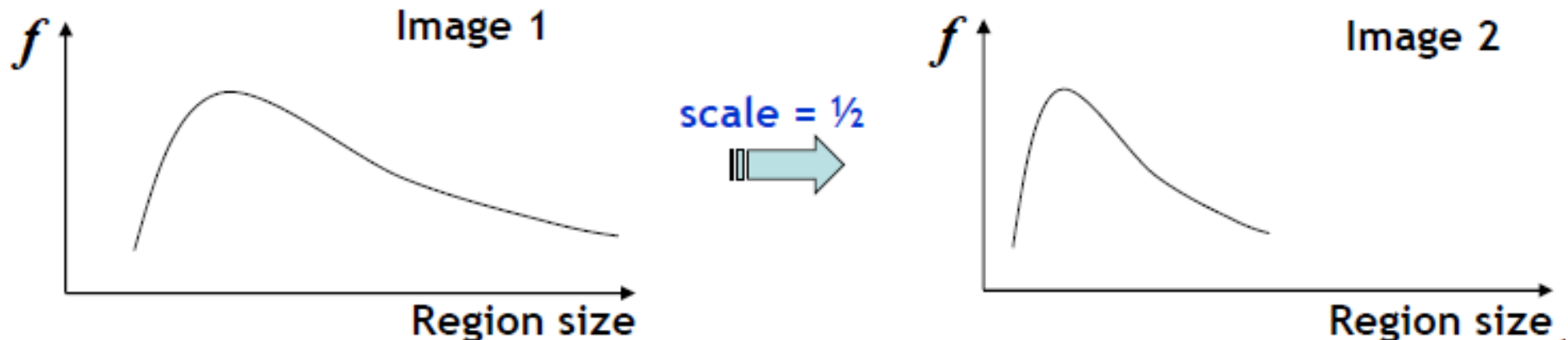


# Scale Invariant Detection

121

## □ Solution:

- ▣ Design a function on the region (circle), which is “scale invariant” (the same for corresponding regions, even if they are at different scales)
  - **Example: average intensity. For corresponding regions (even of different sizes) will be the same.**
- ▣ For a point in one image, we can consider it as a function of region size (circle radius)



# Scale Invariant Detection

122

## Intuition:

- Find scale that gives local maxima of some function  $f$  in both position and scale.

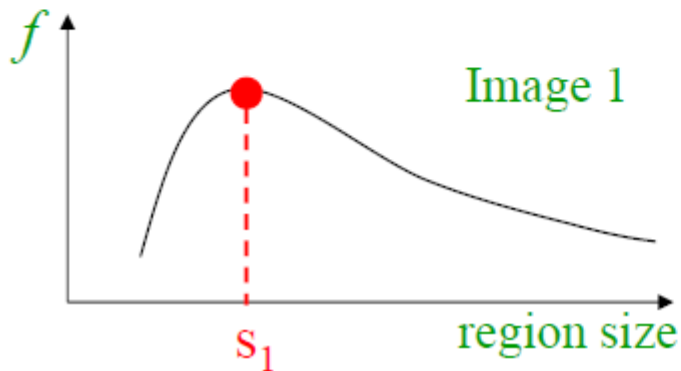
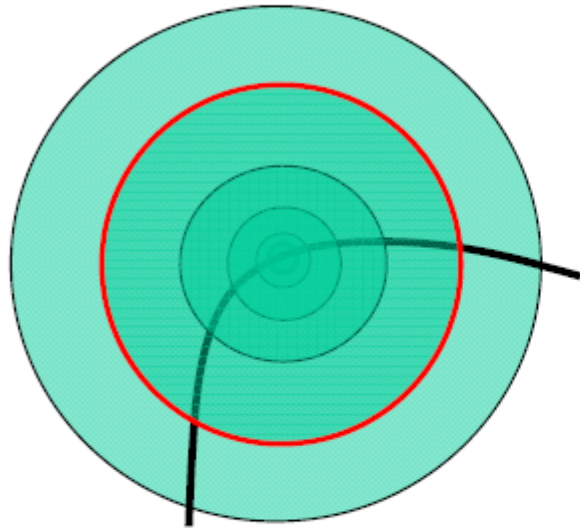


Image 1

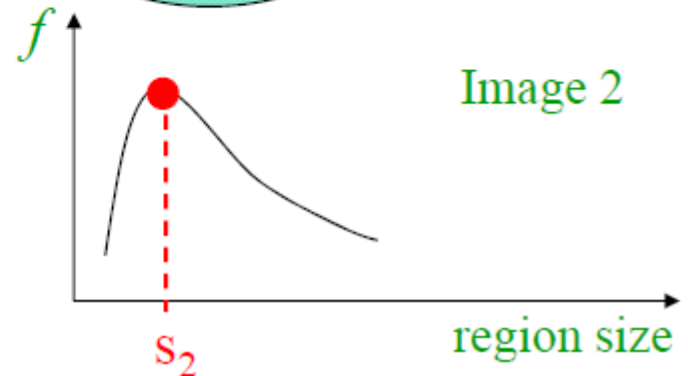
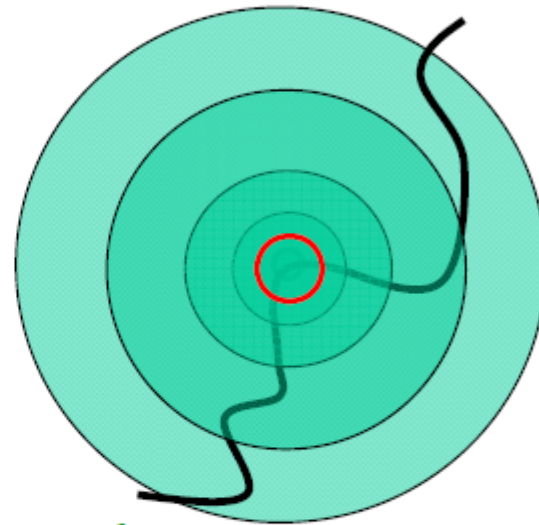


Image 2

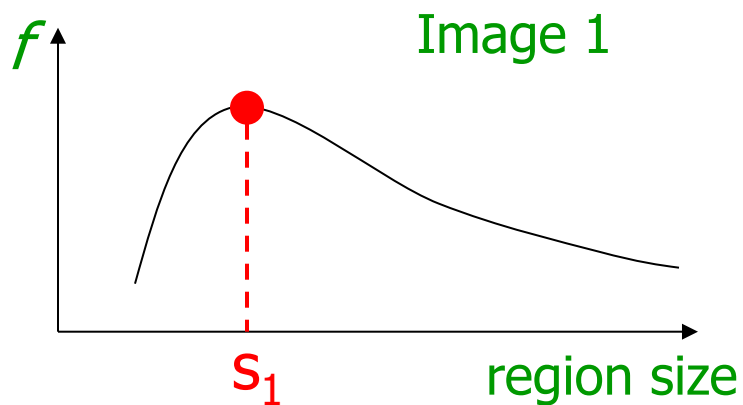
K. Grauman,

# Scale Invariant Detection

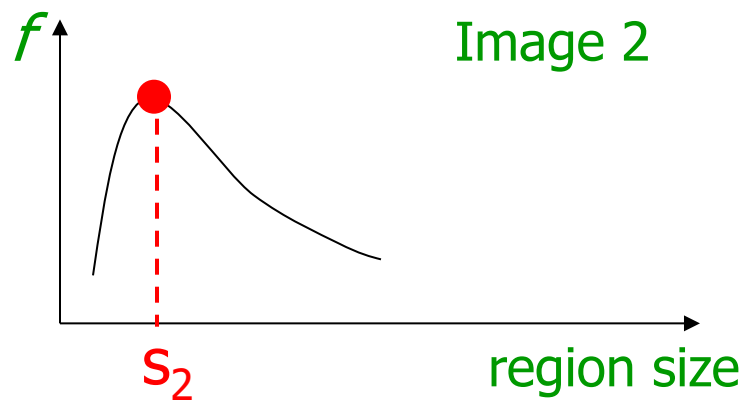
123

- Common approach: Take a local maximum of this function
- **Observation:** region size, for which the maximum is achieved, should be *invariant* to image scale.

**Important: this scale invariant region size is found in each image independently!**



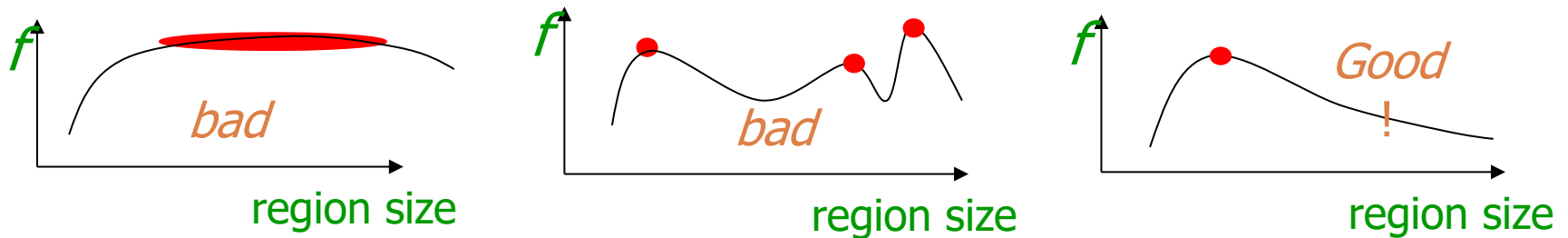
scale = 1/2



# Scale Invariant Detection

124

- A “good” function for scale detection: has one stable sharp peak



- For usual images: a good function would be a one which responds to contrast (sharp local intensity change)

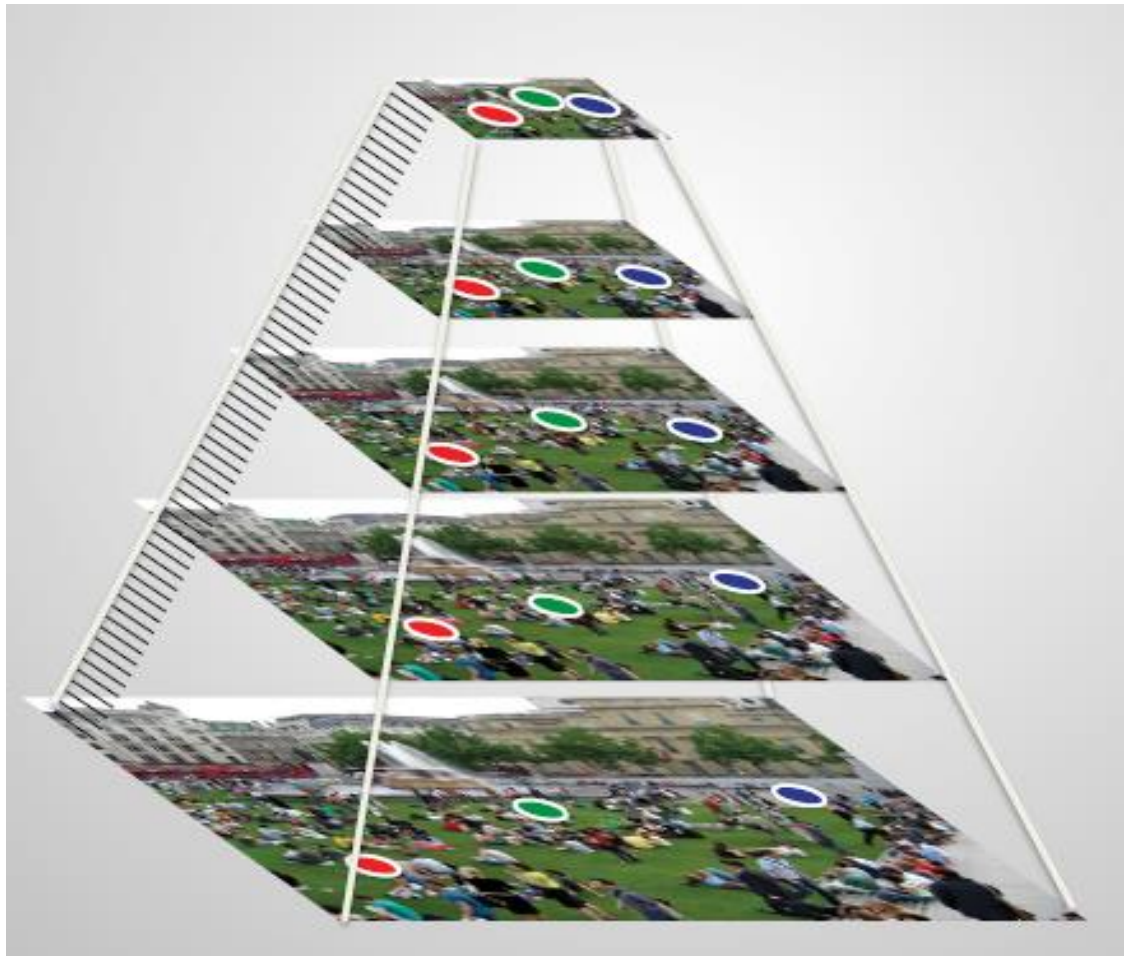
# Scale invariance

- **Requires a method to repeatably select points in location and scale**
- **The** only reasonable scale-space kernel is a Gaussian
- An efficient choice is to detect peaks in the difference of Gaussian pyramid
- Difference-of-Gaussian with constant ratio of scales is a close approximation to scale-normalized Laplacian

# Pyramid

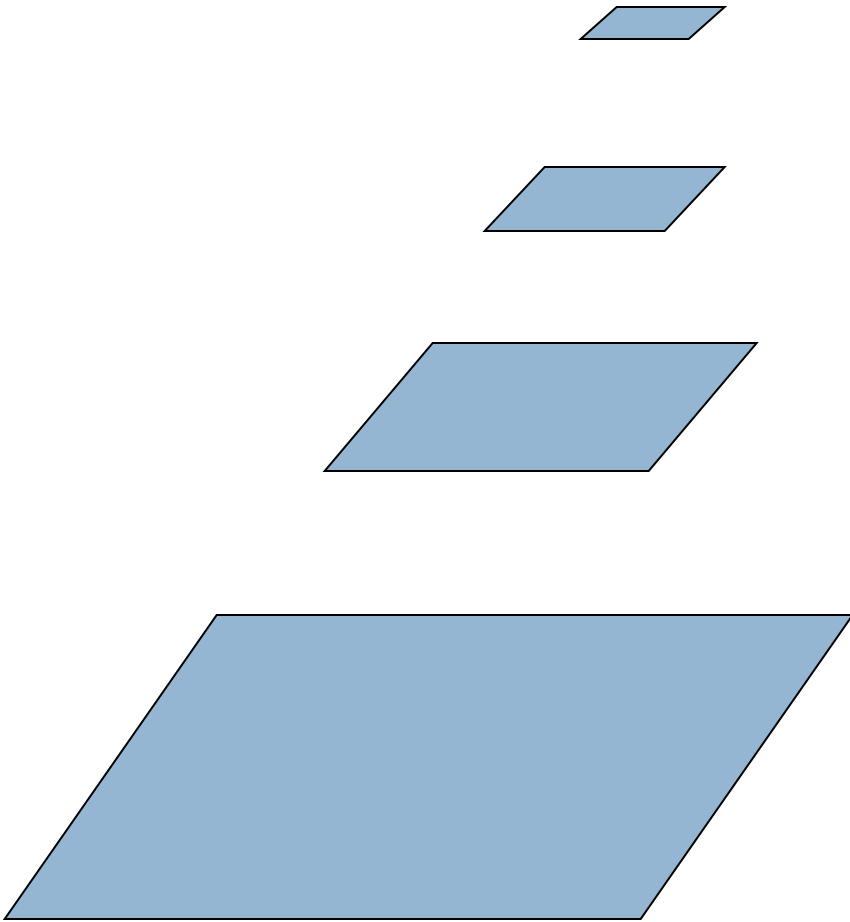
126

- **Pyramid** is one way to represent images in Multi-Scale
- Pyramid can capture global and local features



# Aside: Image Pyramids

127



And so on.

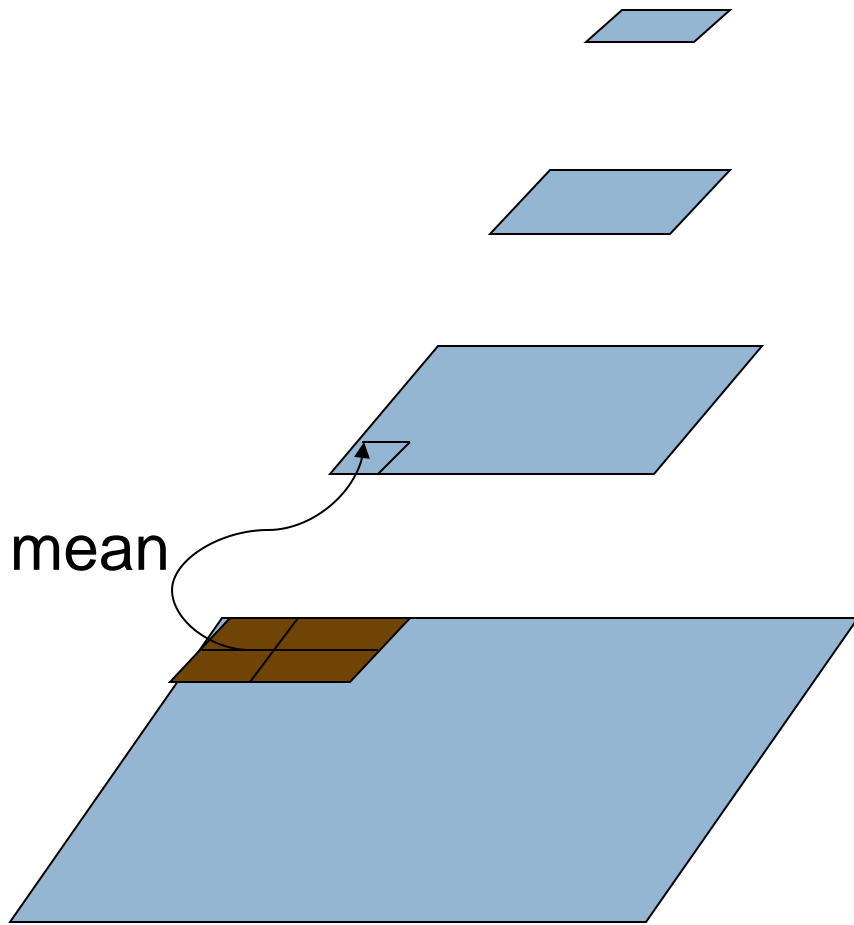
3<sup>rd</sup> level is derived from the 2<sup>nd</sup> level according to the same function

2<sup>nd</sup> level is derived from the original image according to some function

Bottom level is the original image.

# Aside: Mean Pyramid

128



And so on.

At 3<sup>rd</sup> level, each pixel is the mean of 4 pixels in the 2<sup>nd</sup> level.

At 2<sup>nd</sup> level, each pixel is the mean of 4 pixels in the original image.

Bottom level is the original image.



# SIFT - Scale Invariant Feature Transforms

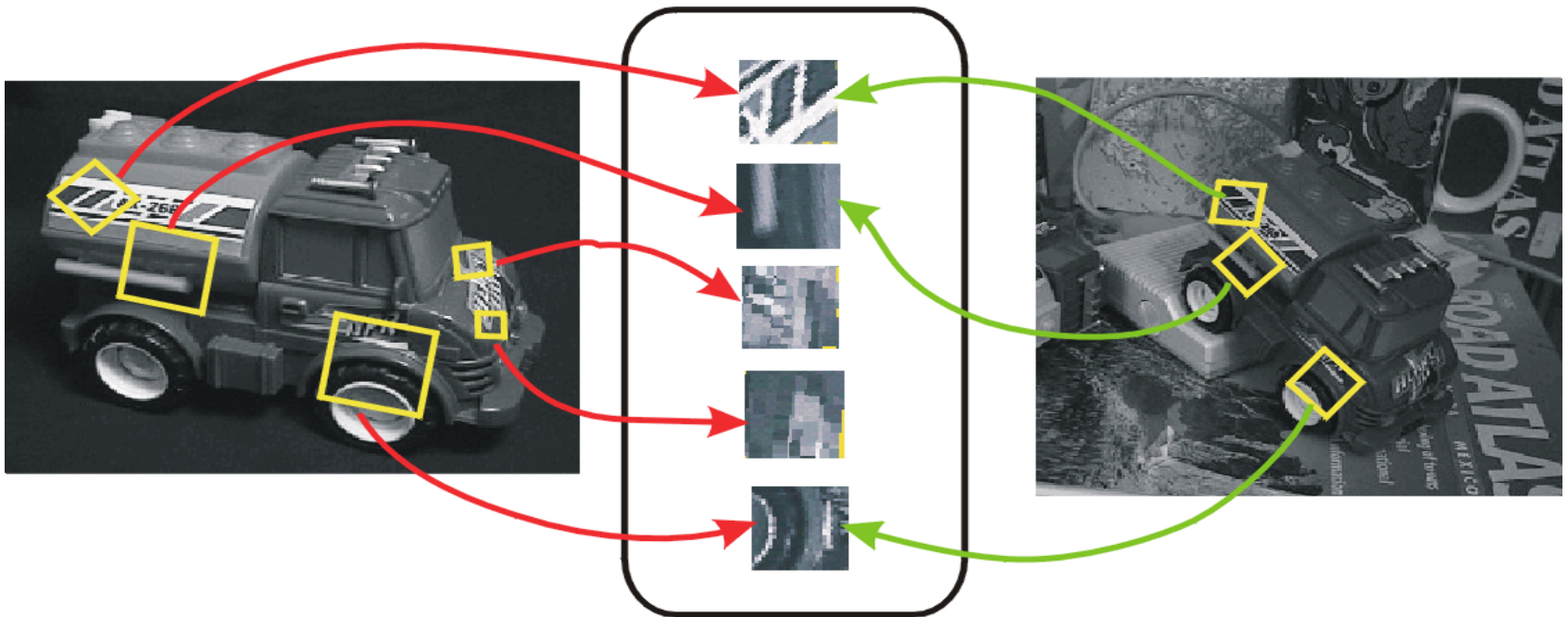
129

- SIFT image features provide a set of features of an object that are not affected by many of the complications experienced in other methods, such as object scaling and rotation.
- While allowing for an object to be recognized in a larger image SIFT image features also allow for objects in multiple images of the same location, taken from different positions within the environment, to be recognized.
- SIFT features are also very resilient to the effects of "noise" in the image.
- The SIFT approach, for image feature generation, takes an image and transforms it into a "large collection of local feature vectors"

# SIFT - Idea

130

- Image content is transformed into local feature coordinates that are invariant to translation, rotation, scale, and other imaging parameters



# Claimed Advantages of SIFT

131

- **Locality:** features are local, so robust to occlusion and clutter (no prior segmentation)
- **Distinctiveness:** individual features can be matched to a large database of objects
- **Quantity:** many features can be generated for even small objects
- **Efficiency:** close to real-time performance
- **Extensibility:** can easily be extended to wide range of differing feature types, with each adding robustness

# Uses for SIFT

132

- Feature points are used also for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Object recognition
  - Indexing and database retrieval
  - Robot navigation
  - ... many others

# Overall Procedure at a High Level

133

- Step 1: Constructing a scale space
- Step 2: Laplacian of Gaussian approximation
- Step 3: Finding Keypoints
- Step 4: Eliminate edges and low contrast regions
- Step 5: Assign an orientation to the keypoints
- Step 6: Generate SIFT features

# Constructing a scale space

134

- This stage of the filtering attempts to identify those locations and scales that are identifiable from different views of the same object.
- This can be efficiently achieved using a "scale space" function.
- The scale space is defined by the function:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

Where:

- \* is the convolution operator,
- L is a blurred image
- G is the Gaussian Blur operator
- I is the input image.
- x,y are the location coordinates
- $\sigma$  is the "scale" parameter. Think of it as the amount of blur. Greater the value, greater the blur.

# Constructing a scale space

135

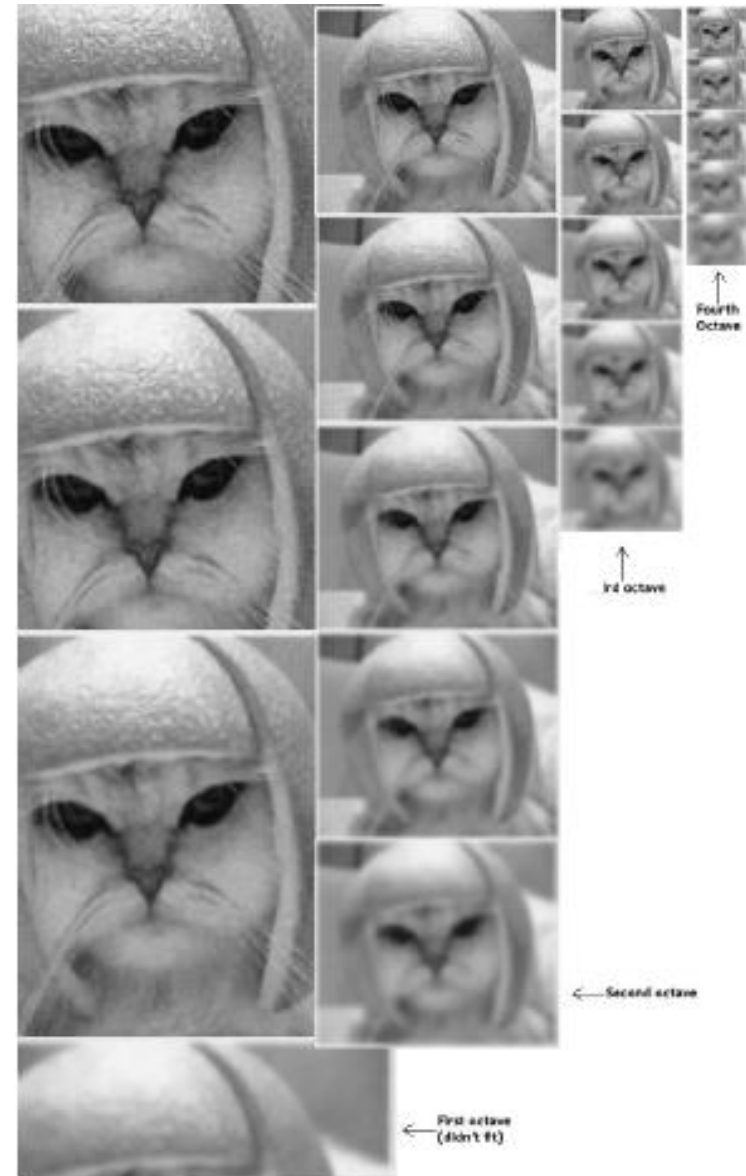
□ Example:



# Constructing a scale space

136

- SIFT takes scale spaces to the next level.
- Resize the original image to half size. And you generate blurred out images again. And you keep repeating.





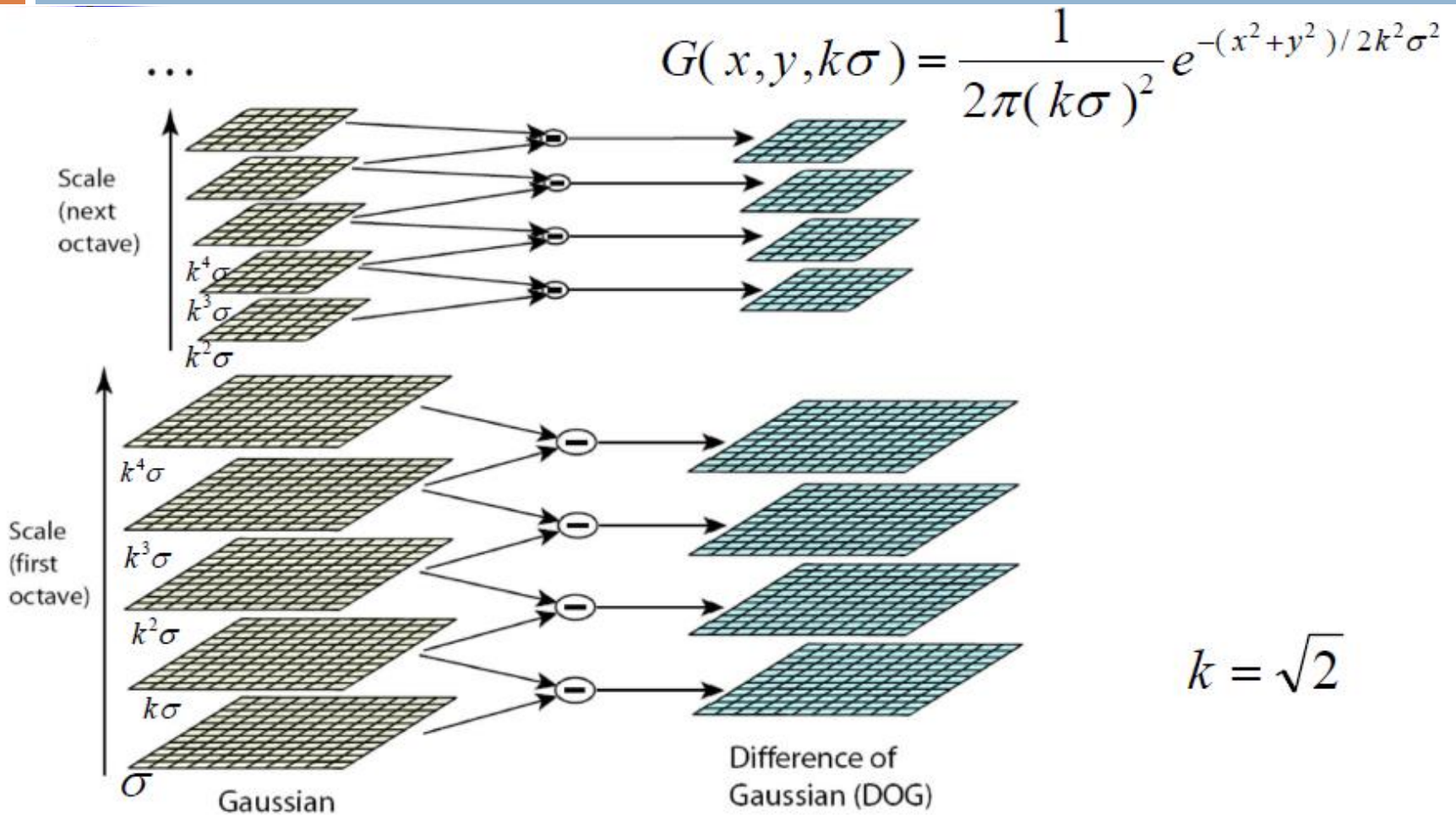
## Step 2: Laplacian of Gaussian approximation

137

- To find key points use Laplacian of Gaussian (LoG)
  - ▣ Take an image, and blur it a little.
  - ▣ Then calculate second order derivatives on it (or, the “laplacian”).
- The problem is, calculating all those second order derivatives is computationally intensive.
- Solution, use the Difference of Gaussians (DoG).
  - ▣ We use the scale space (from previous step).
  - ▣ We calculate the difference between two consecutive scales.
  - ▣ These DoG images are a great for finding out interesting key points in the image

# Step 2: Laplacian of Gaussian approximation

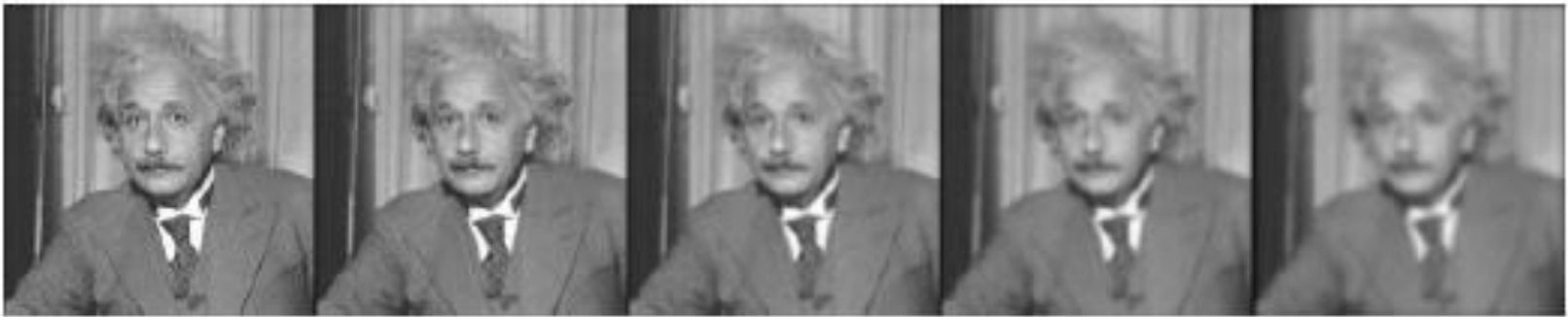
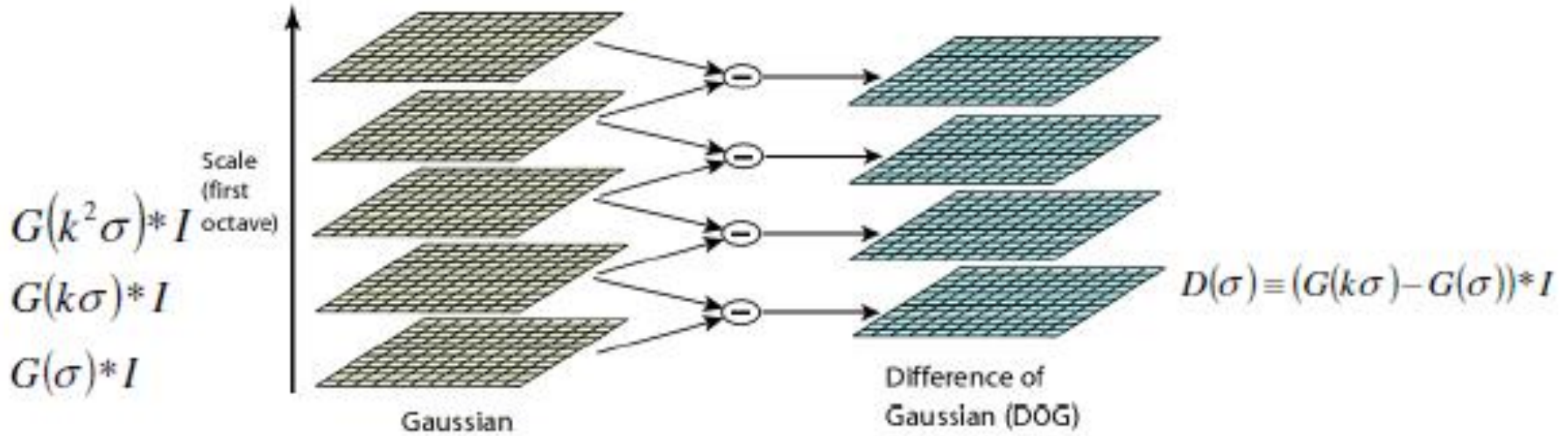
138



These Difference of Gaussian images are approximately equivalent to the Laplacian of Gaussian. And we've replaced a computationally intensive process with a simple subtraction (fast and efficient).

# Step 2: Laplacian of Gaussian approximation

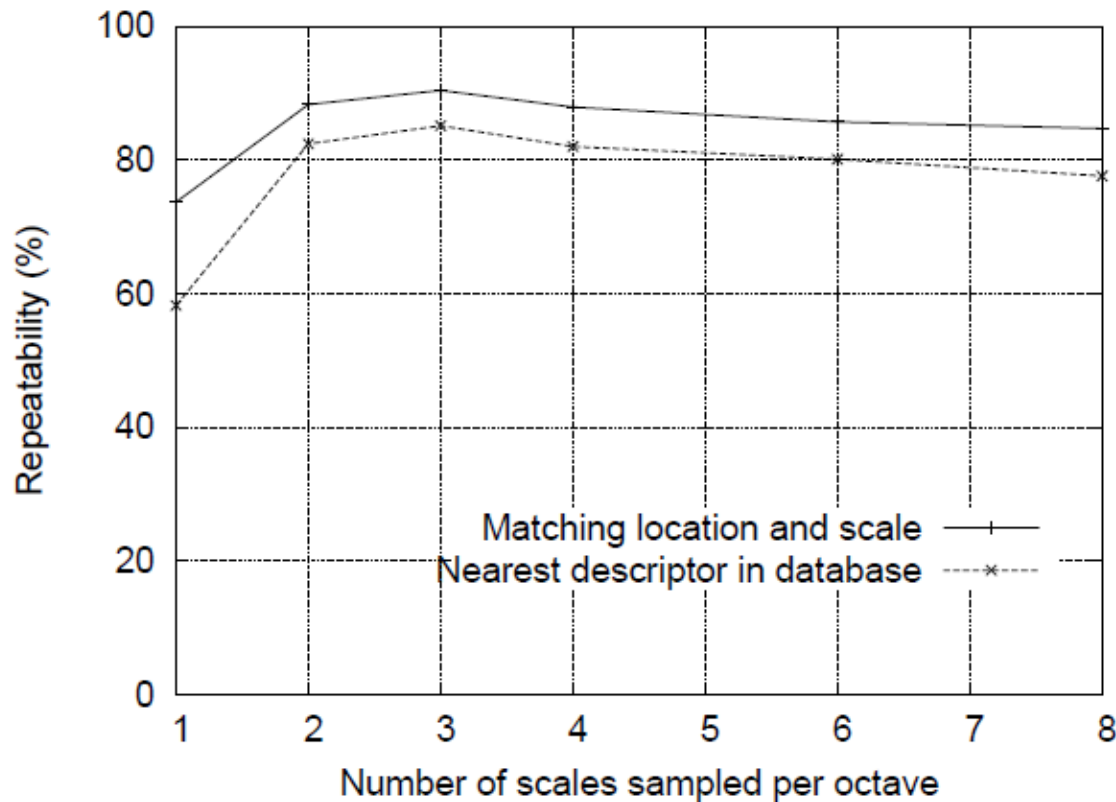
139



# How many scales per octave?

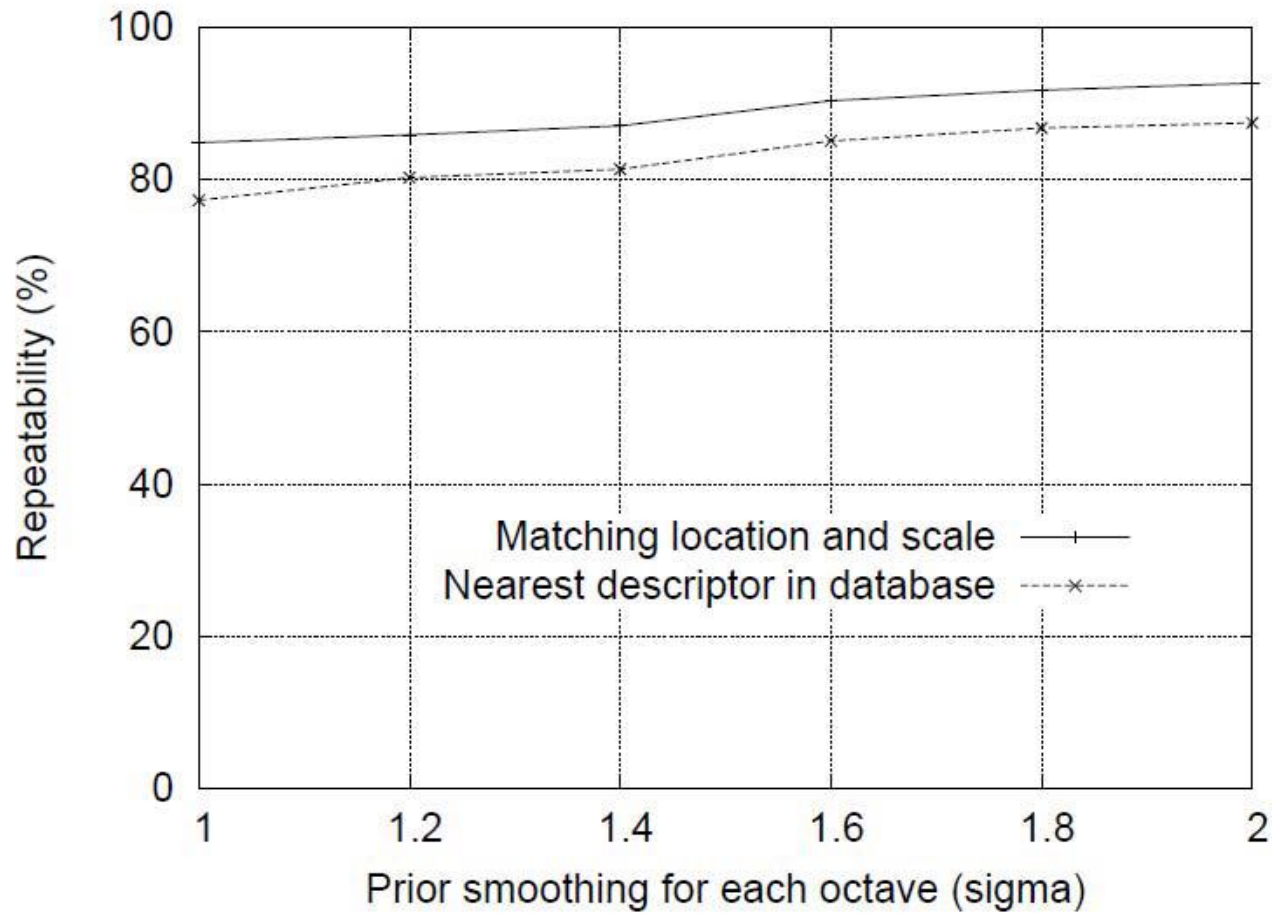
140

- A collection of 32 real images drawn from a diverse range, including:
  - Outdoor scenes, human faces, aerial photographs, and industrial
- Each image was then subject to a range of transformations:
  - Rotation, scaling, affine stretch, change in brightness and contrast, and addition of image noise.



# Initial value of sigma

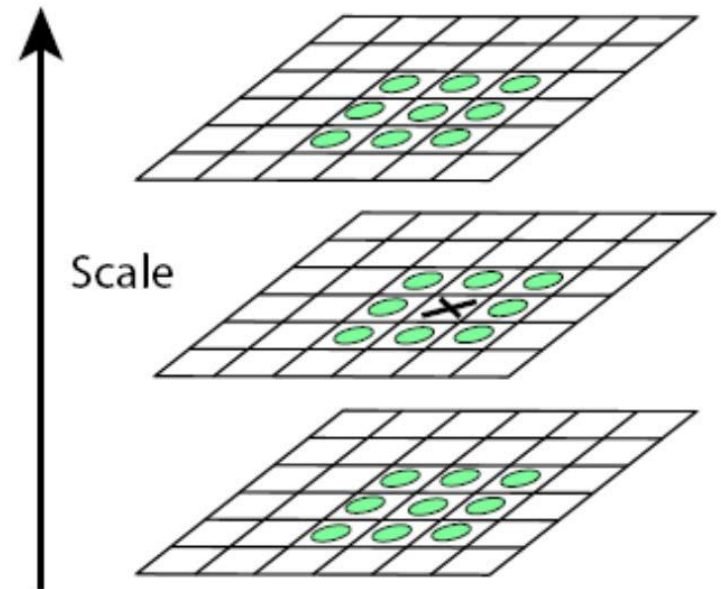
141



# Step 3: Finding Keypoints

142

- To detect the local maxima and minima of  $D(x, y, \sigma)$ 
  - Each point is compared with its 8 neighbors at the same scale, and its 9 neighbors up and down one scale.
  - X is marked as a “key point” if it is the greatest or least of all 26 Neighbours
- Large number of extrema,
  - computationally expensive
  - Detect the most stable subset with a coarse sampling of scales



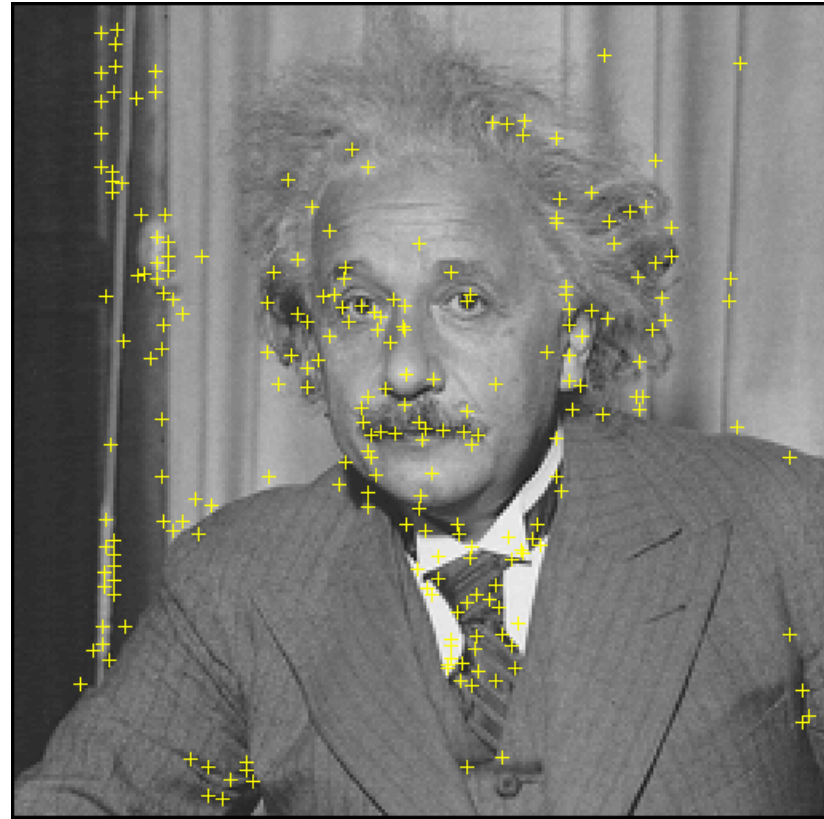
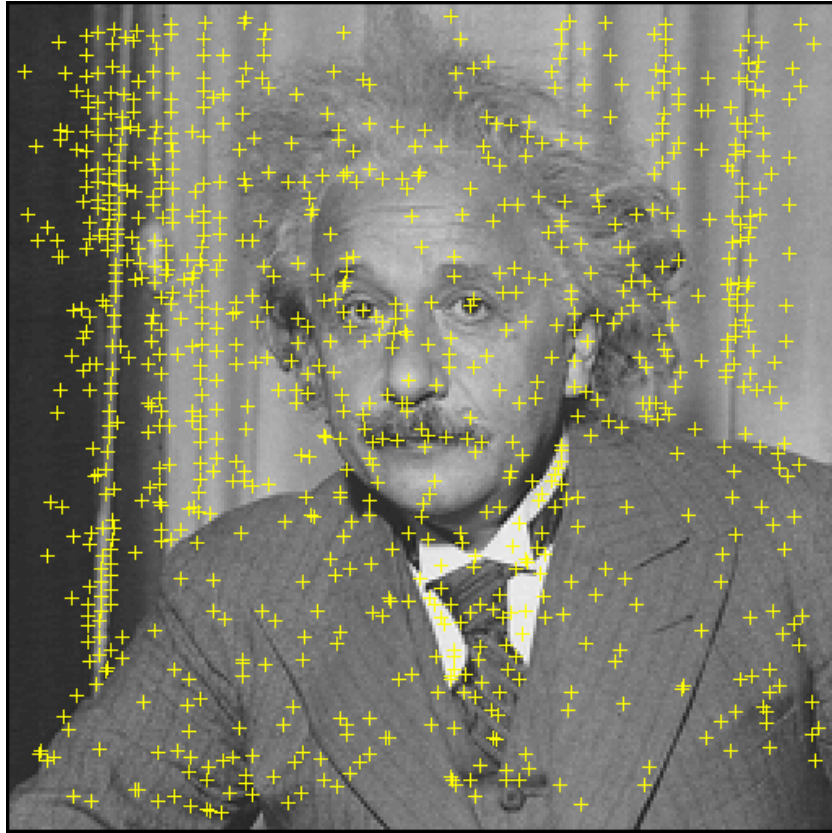
## Step 4: Eliminate edges and low contrast regions

143

- Key points generated in the previous step produce a lot of key points. Some of them lie along an edge, or they don't have enough contrast. In both cases, they are not useful as features, so we need to get rid of them.
- Reject points with bad contrast:
  - ▣ DoG smaller than 0.03 (image values in  $[0,1]$ )
- Reject edges
  - ▣ Use Harris detector and keep only corners

# Step 4: Eliminate edges and low contrast regions

144





# Step 5: Assign an orientation to the keypoints

145

- After step 4, we have legitimate key points.
  - ▣ We already know the scale at which the keypoint was detected (it's the same as the scale of the blurred image). So we have scale invariance.
- The next thing is to assign an orientation to each keypoint. This orientation provides rotation invariance
- This step aims to assign a consistent orientation to the keypoints based on local image properties. The keypoint descriptor, can then be represented relative to this orientation, achieving invariance to rotation.
- The idea is to collect gradient magnitude and orientation around each keypoint (window  $16 \times 16$ ). Then we figure out the most prominent orientation(s) in that region. And we assign this orientation(s) to the keypoint.
- This orientation provides rotation invariance

## Step 5: Assign an orientation to the keypoints

146

- For each pixel in the widow around Keypoint compute gradient magnitude and orientation using finite differences:

$$\textit{GradientVector} = \begin{bmatrix} L(x+1, y) - L(x-1, y) \\ L(x, y+1) - L(x, y-1) \end{bmatrix}$$

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)))$$

# Step 5: Assign an orientation to the keypoints

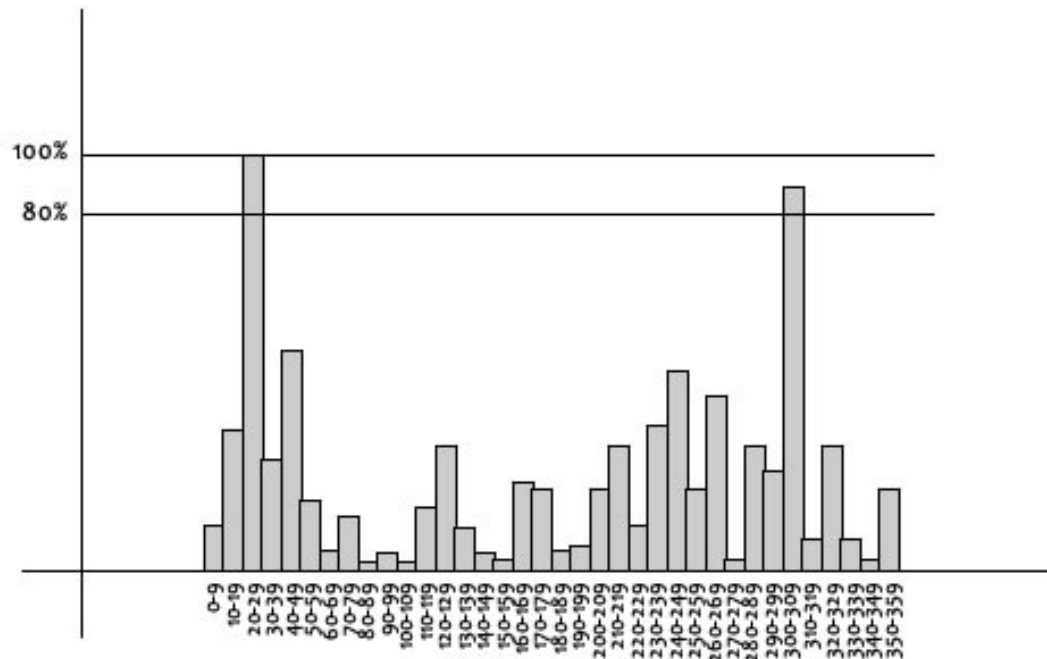
147

- The magnitude and orientation is calculated for all pixels around the keypoint as following:
  - ▣ Create a weighted direction histogram in a neighborhood of a key point
  - ▣ In this histogram, the 360 degrees of orientation are broken into 36 bins (each 10 degrees).
    - Lets say the gradient direction at a certain point (in the "orientation collection region") is 18.759 degrees, then it will go into the 10-19 degree bin. And the "amount/weight" that is added to the bin is proportional to the magnitude of gradient at that point.
      - In SIFT, you need to blur magnitude of gradient by an amount of  $1.5 * \sigma$ .
  - ▣ The size of the "orientation collection region" around the keypoint depends on it's scale. The bigger the scale, the bigger the collection region
    - The window size, or the "orientation collection region", is equal to the size of the kernel for Gaussian Blur of amount  $1.5 * \sigma$ .

# Step 5: Assign an orientation to the keypoints

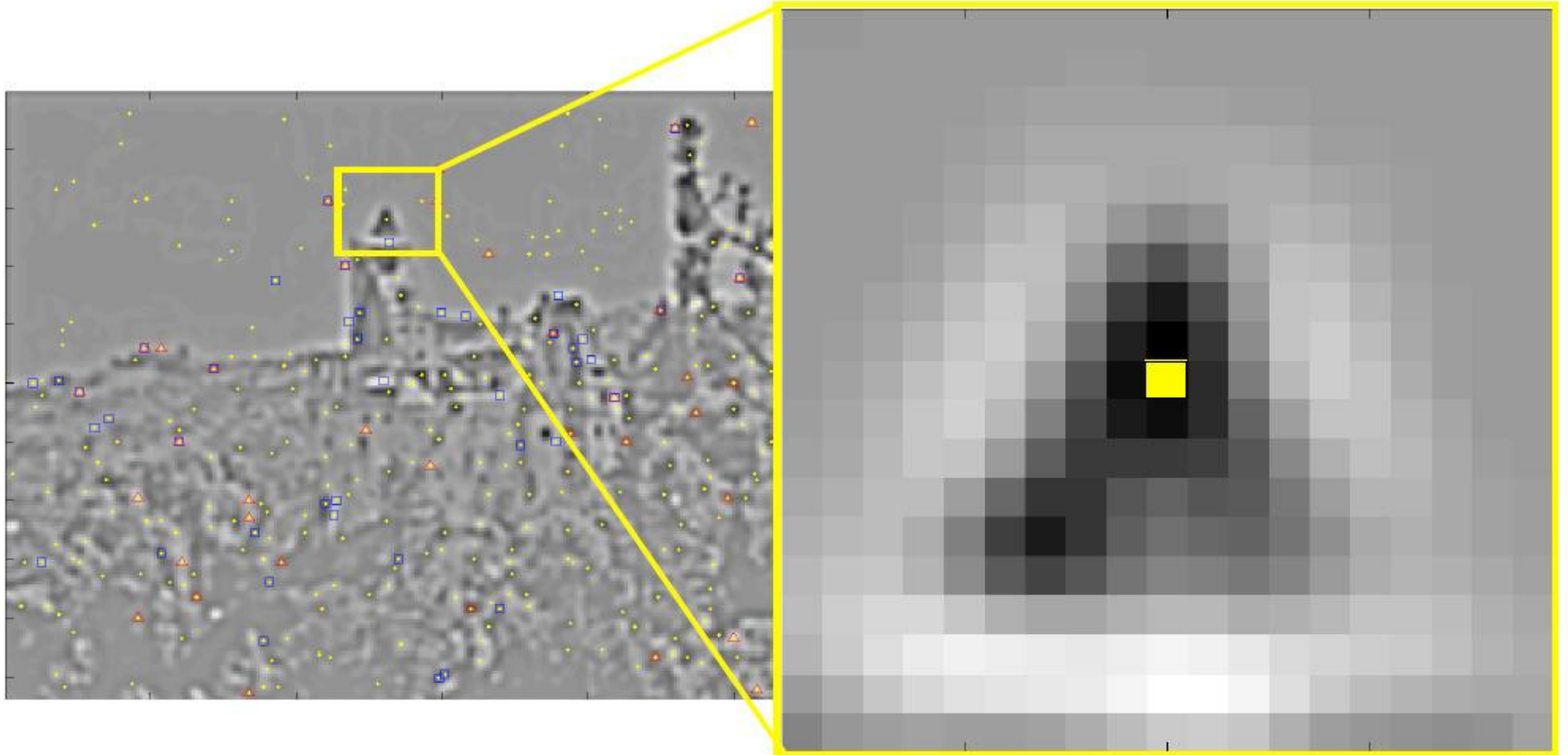
148

- The histogram peaks at 20-29 degrees. So, the keypoint is assigned orientation 3 (the third bin). And the “amount” that is added to the bin is proportional to the magnitude of gradient at that point
- Also, any peaks above 80% of the highest peak are converted into a new keypoint. This new keypoint has the same location and scale as the original. But it’s orientation is equal to the other peak. So, orientation can split up one keypoint into multiple keypoints.



# Orientation assignment

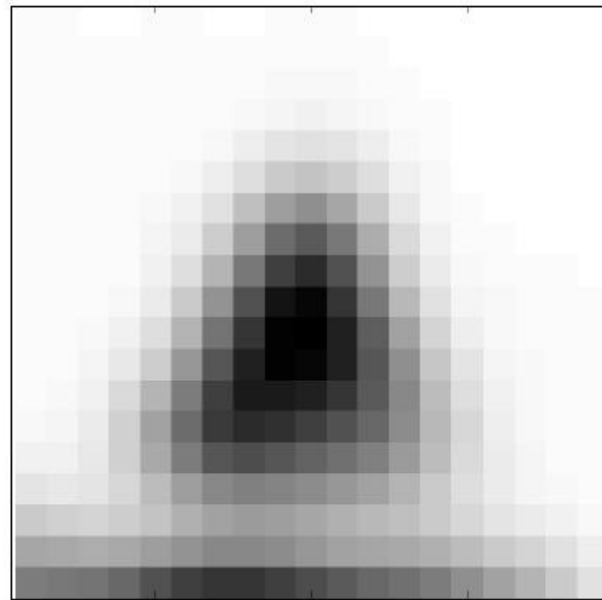
149



- **Keypoint location = extrema location**
- **Keypoint scale is scale of the DOG image**

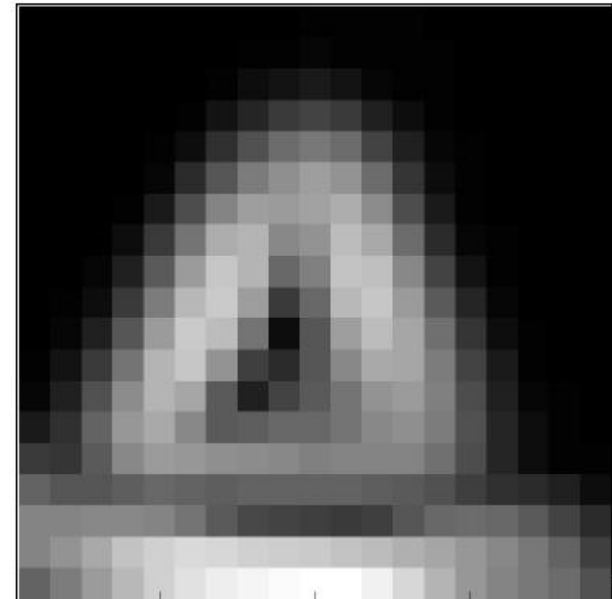
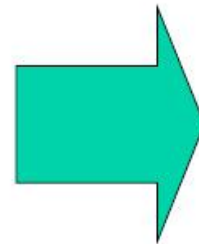
# Orientation assignment

150

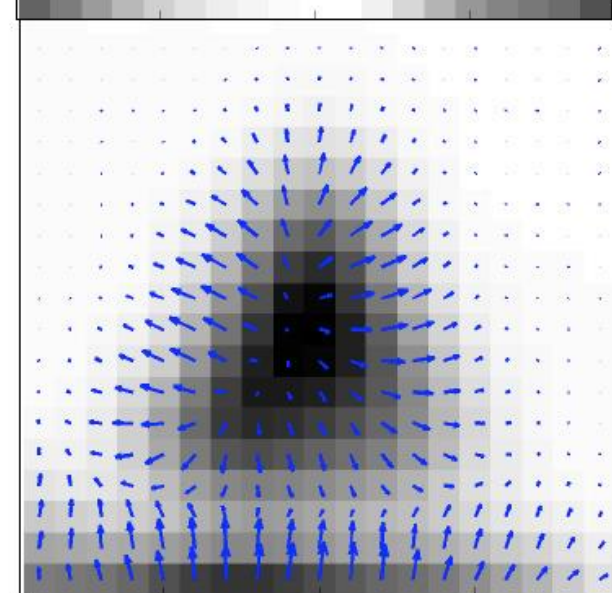


**gaussian image  
(at closest scale,  
from pyramid)**

**gradient  
magnitude**



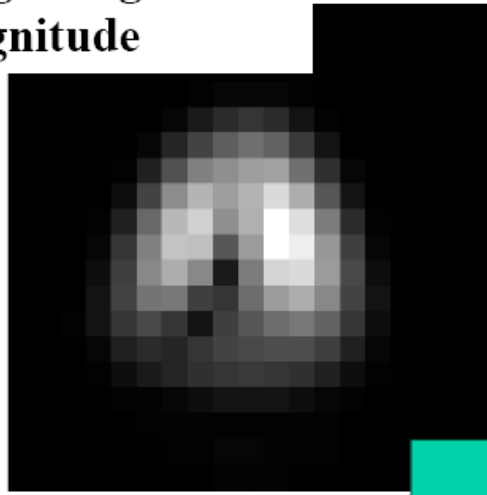
**gradient  
orientation**



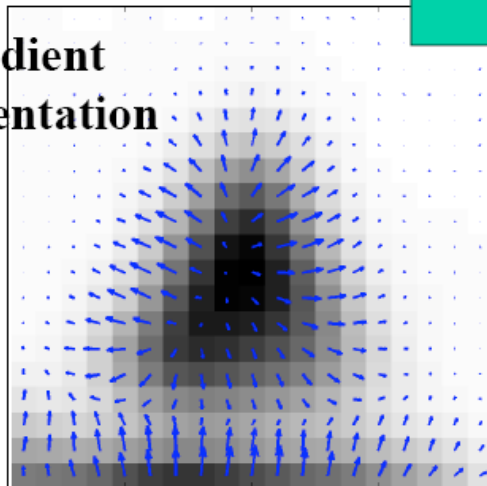
# Orientation assignment

151

**weighted gradient  
magnitude**

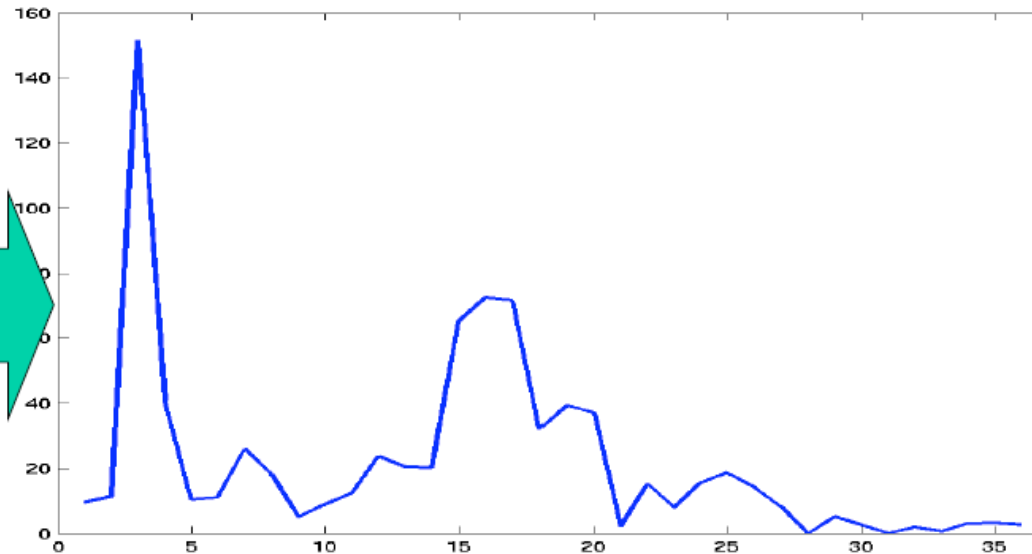


**gradient  
orientation**



**weighted orientation histogram.**

Each bucket contains sum of weighted gradient magnitudes corresponding to angles that fall within that bucket.



**36 buckets**

**10 degree range of angles in each bucket, i.e.**

**$0 \leq \text{ang} < 10$  : bucket 1**

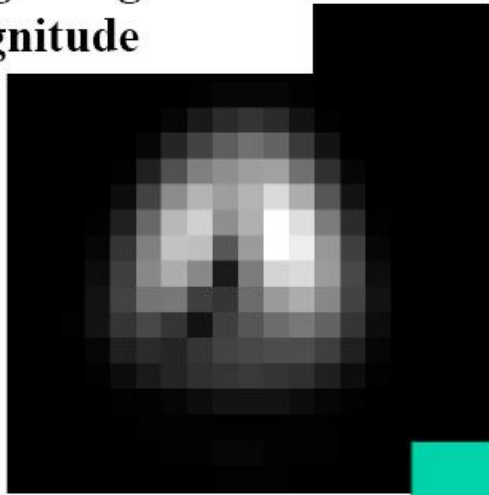
**$10 \leq \text{ang} < 20$  : bucket 2**

**$20 \leq \text{ang} < 30$  : bucket 3 ...**

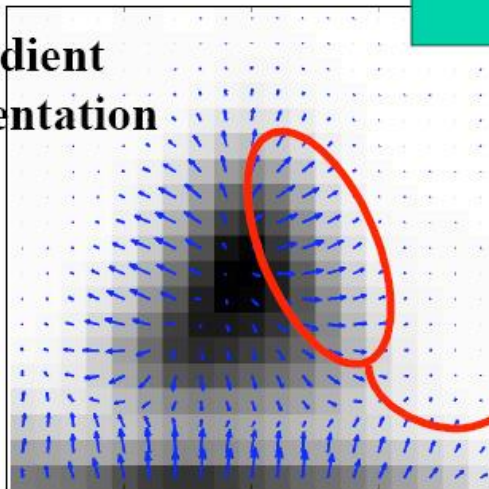
# Orientation assignment

152

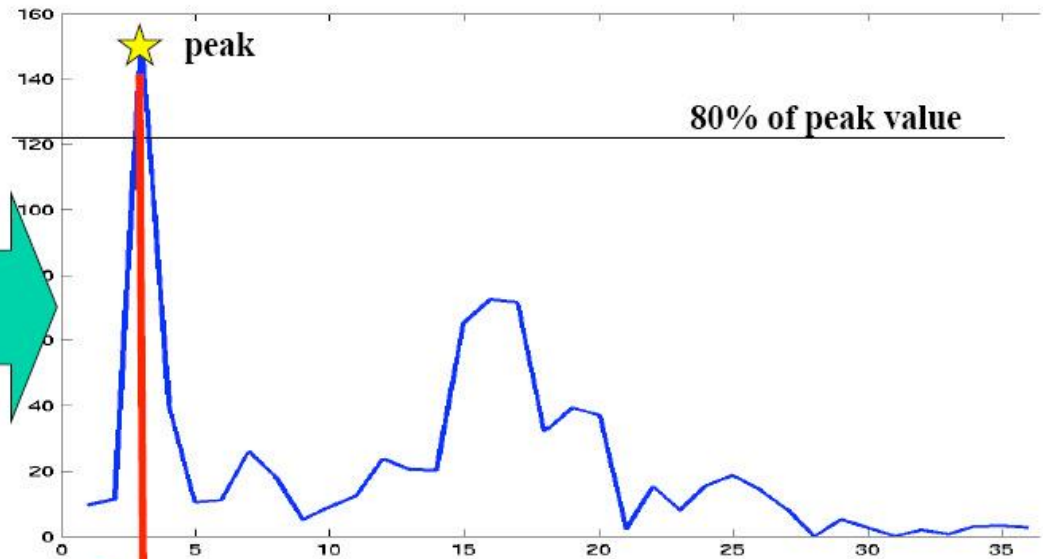
weighted gradient  
magnitude



gradient  
orientation



weighted orientation histogram.



20-30 degrees

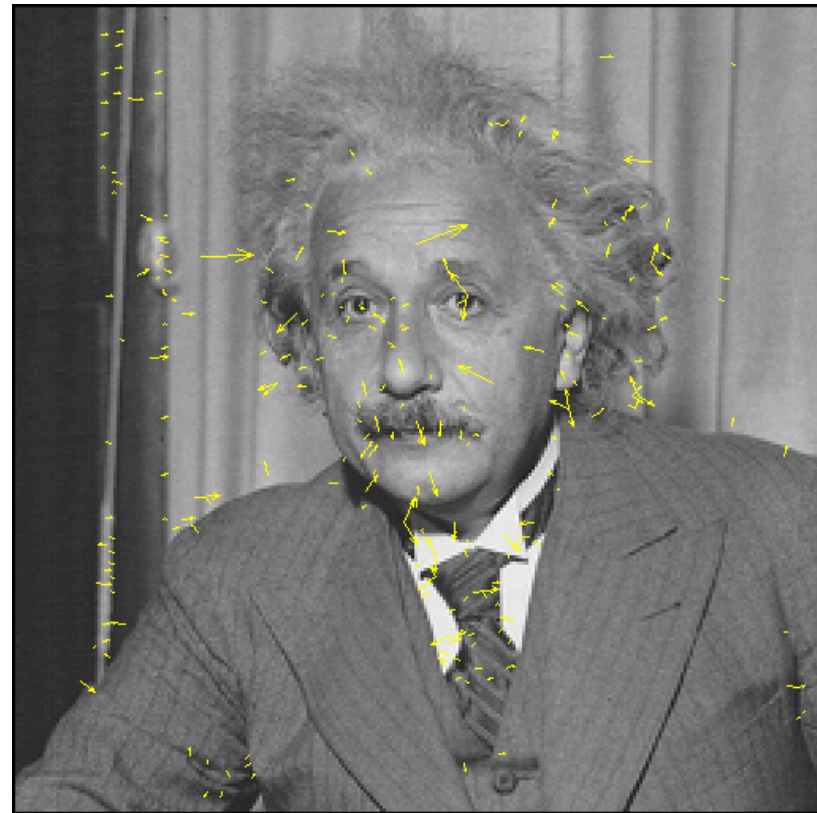
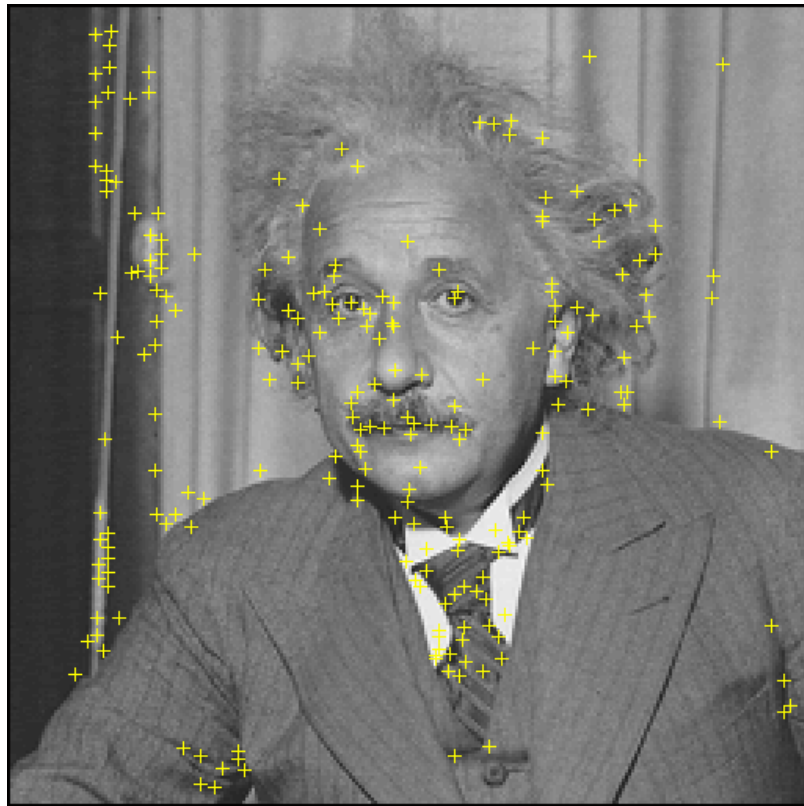
**Orientation of keypoint  
is approximately 25 degrees**



# Orientation assignment

153

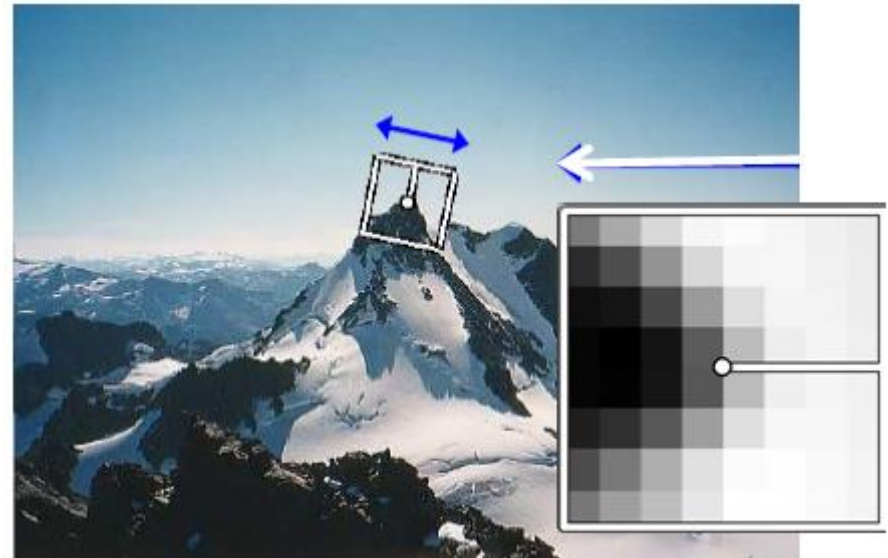
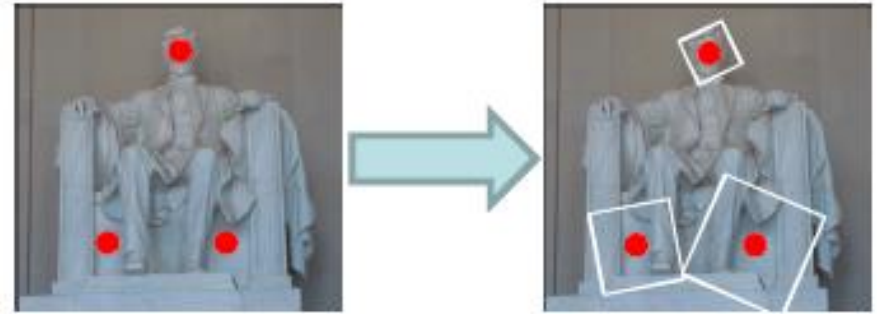
## Orientation Visualization



# Making descriptor rotation invariant

154

- Rotate patch (window around keypoint) according to its dominant gradient orientation to the horizontal orientation
  - ▣ The dominant orientation will be horizontal orientation
  - ▣ This puts the patches into a canonical orientation.
- Make scaling according to the arrow length
  - ▣ Eliminate scaling problem



# Step 6: Generate SIFT features

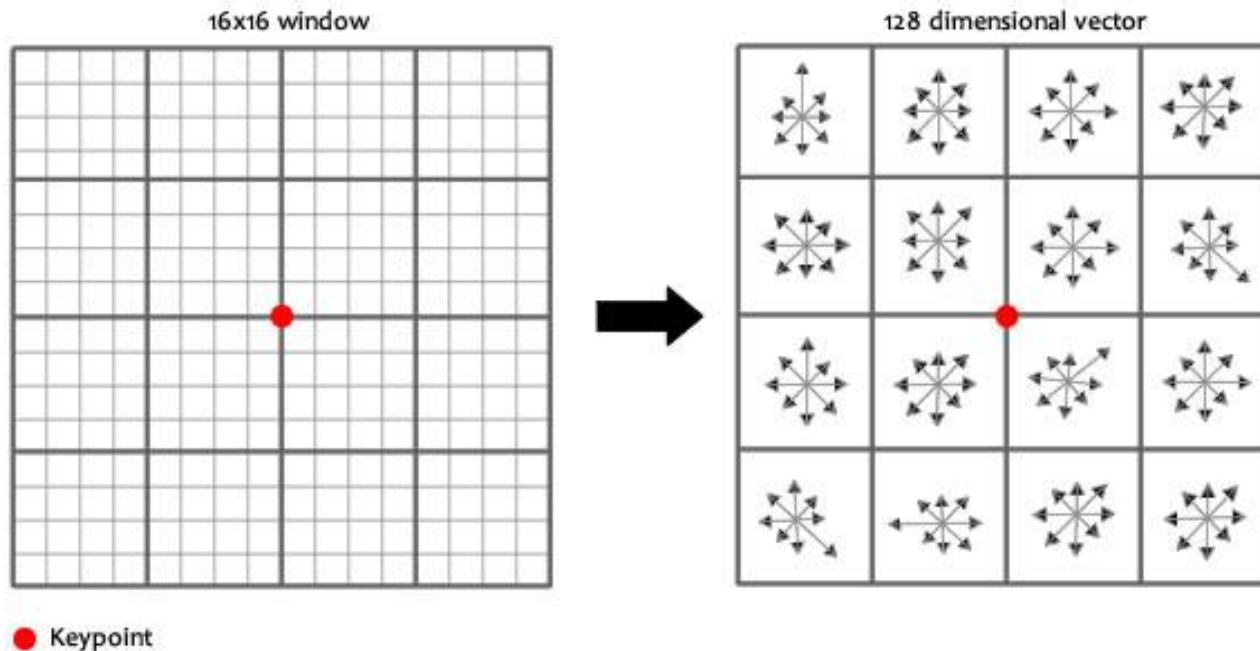
155

- Now we create a fingerprint for each keypoint. This is to identify a keypoint.
- Each point so far has  $x, y, \sigma, m, \theta$ 
  - ▣ Location  $x, y$
  - ▣ Scale:  $\sigma$
  - ▣ Gradient magnitude and orientation:  $m, \theta$
- Now we need a descriptor for the region
  - ▣ Could sample intensities around point, but...
    - Sensitive to lighting changes
    - Sensitive to slight errors in  $x, y, \theta$

# Step 6: Generate SIFT features

156

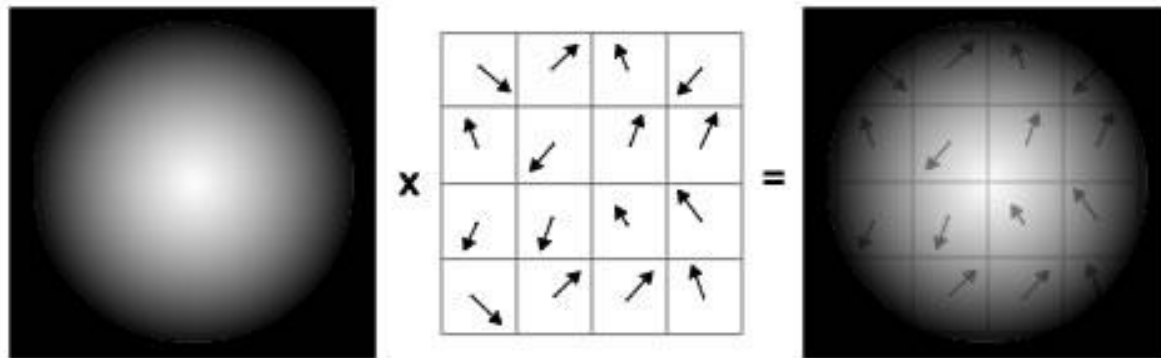
- The idea:
  - ▣ We want to generate a very unique fingerprint for the keypoint.
  - ▣ It should be easy to calculate.
  - ▣ We also want it to be relatively lenient when it is being compared against other keypoints.
- To do this, a 16x16 window around the keypoint. This 16x16 window is broken into sixteen 4x4 windows.



# Step 6: Generate SIFT features

157

- Within each 4x4 window, gradient magnitudes and orientations are calculated. These orientations are put into an 8 bin histogram.
- Any gradient orientation in the range 0-44 degrees add to the first bin. 45-89 add to the next bin. And so on.
- The amount added to the bin depends on the magnitude of the gradient.
- Unlike the past, the amount added also depends on the distance from the keypoint. So gradients that are far away from the keypoint will add smaller values to the histogram.
- This is done using a "gaussian weighting function". This function simply generates a gradient (it's like a 2D bell curve). You multiple it with the magnitude of orientations, and you get a weighted thingy. The farther away, the lesser the magnutide.

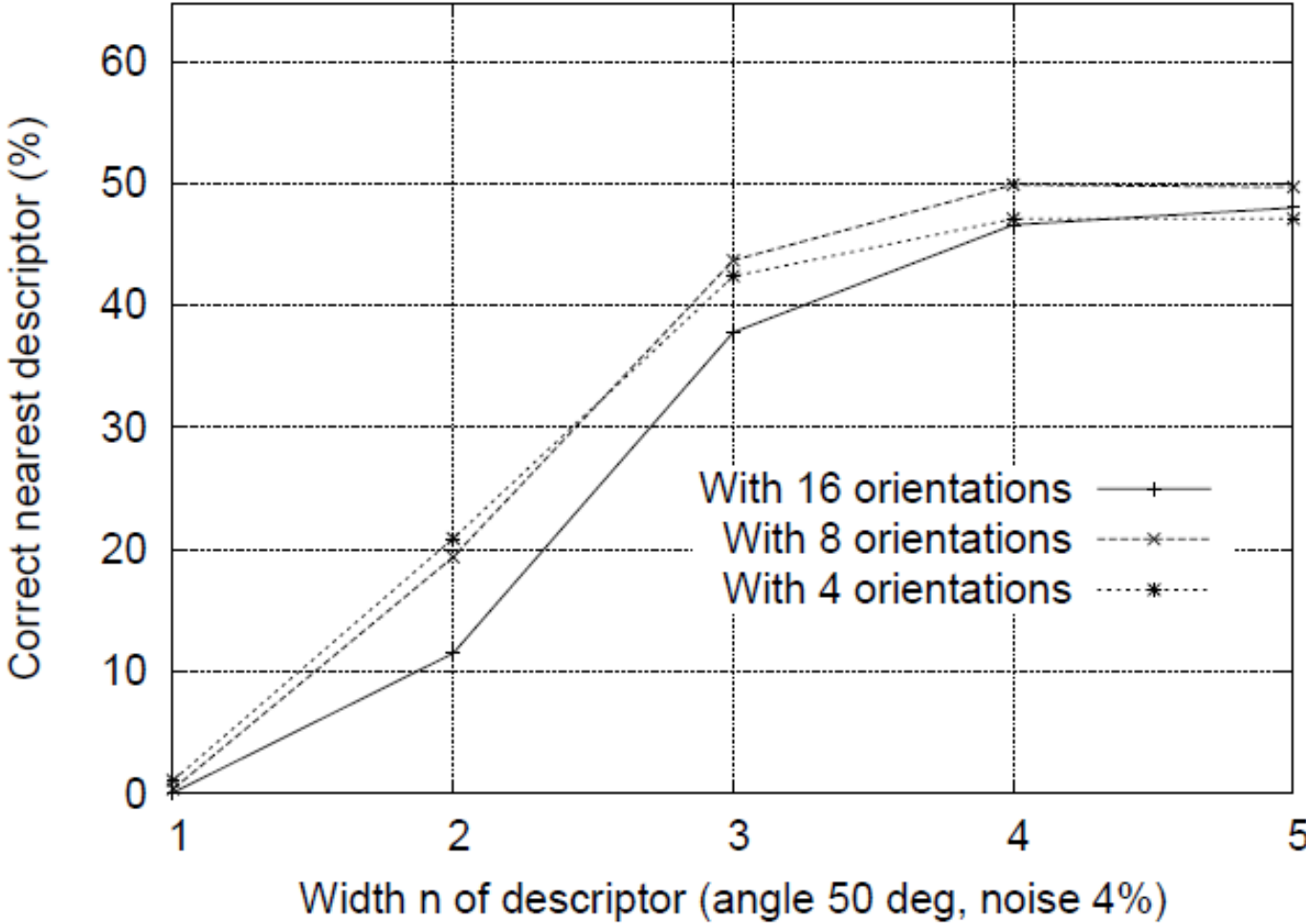


## Step 6: Generate SIFT features

158

- Doing this for all 16 pixels, you would've "compiled" 16 totally random orientations into 8 predetermined bins. You do this for all sixteen 4x4 regions.
- So you end up with  $4 \times 4 \times 8 = 128$  numbers. Once you have all 128 numbers, you normalize them (just like you would normalize a vector in school, divide by root of sum of squares). These 128 numbers form the "feature vector".
- This keypoint is uniquely identified by this feature vector.

# Descriptor Regions (n by n)



# SIFT Keypoint Descriptor Summary

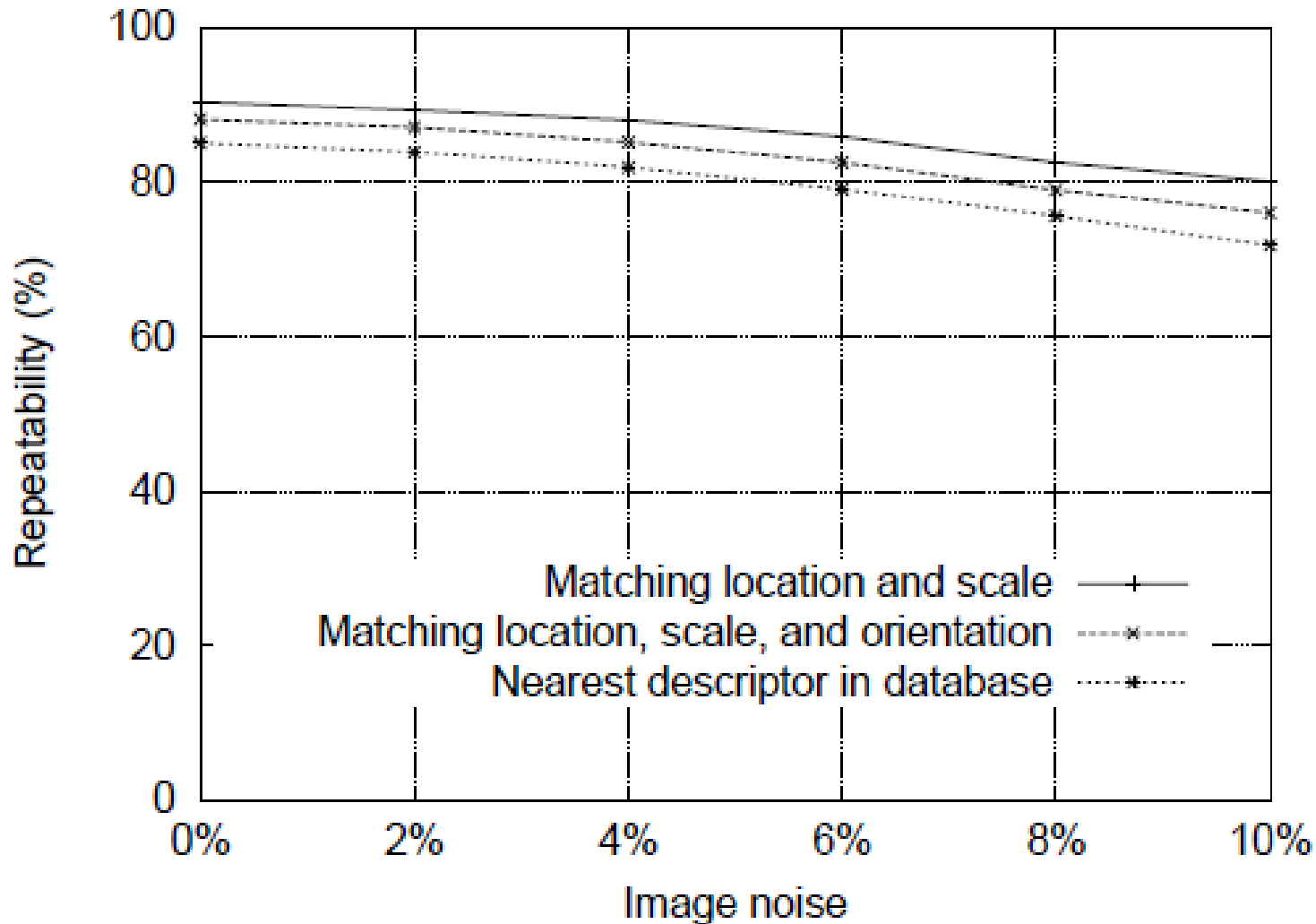
160

- Descriptor: 128-D
  - ▣ 4 by 4 patches, each with 8-D gradient angle histogram:  
 $4 \times 4 \times 8 = 128$
  - ▣ Normalized to reduce the effects of illumination change.
- Position:  $(x, y)$ 
  - ▣ Where the feature is located at.
- Scale
  - ▣ Control the region size for descriptor extraction.
- Orientation
  - ▣ To achieve rotation-invariant descriptor.



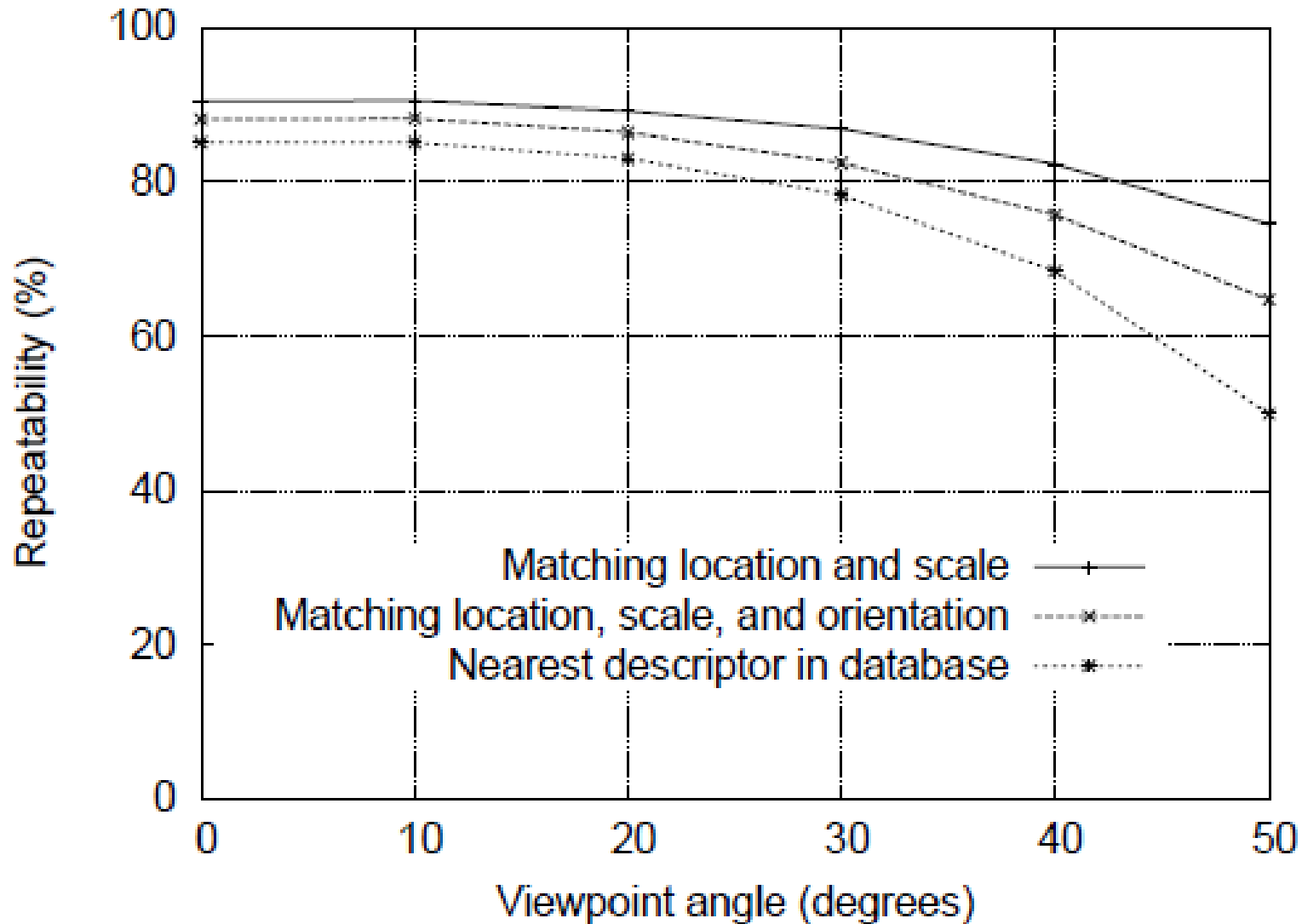
# Effect of Noise on SIFT

161



# Effect of Orientation on SIFT

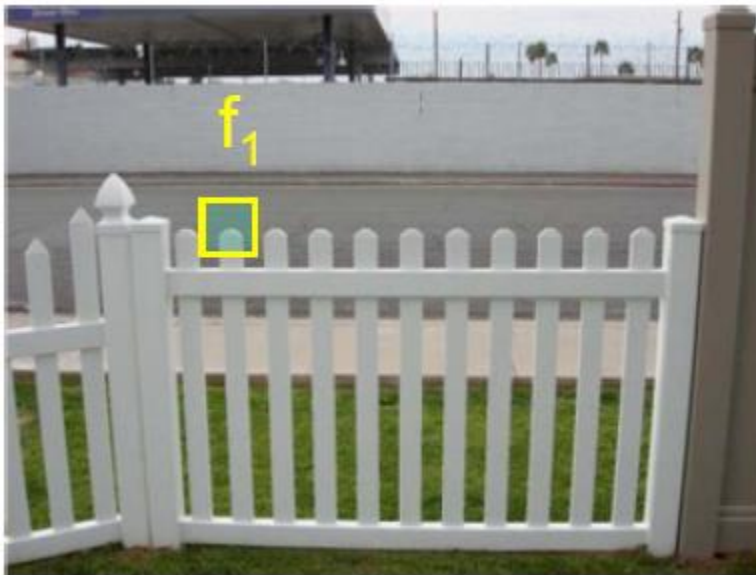
162



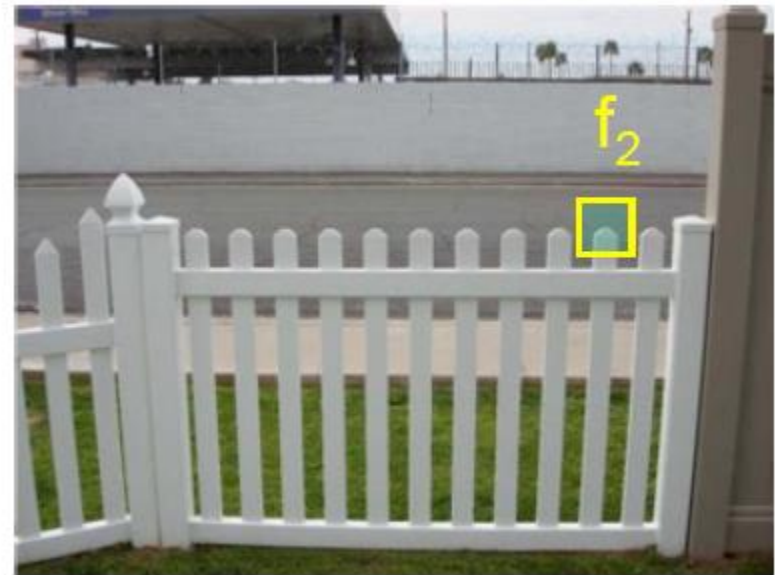
# Keypoints Matching

163

- Given a feature in  $I_1$ , how to find the best match in  $I_2$ ?
  1. Define distance function that compares two descriptors
  2. Test all the features in  $I_2$ , find the one with min distance
- How to define the difference between two features  $f_1, f_2$ ?
  - ▣ Simple approach is  $SSD(f_1, f_2)$ 
    - Sum of square differences between entries of the two descriptors
    - Can give good scores to very ambiguous (bad) matches



$I_1$

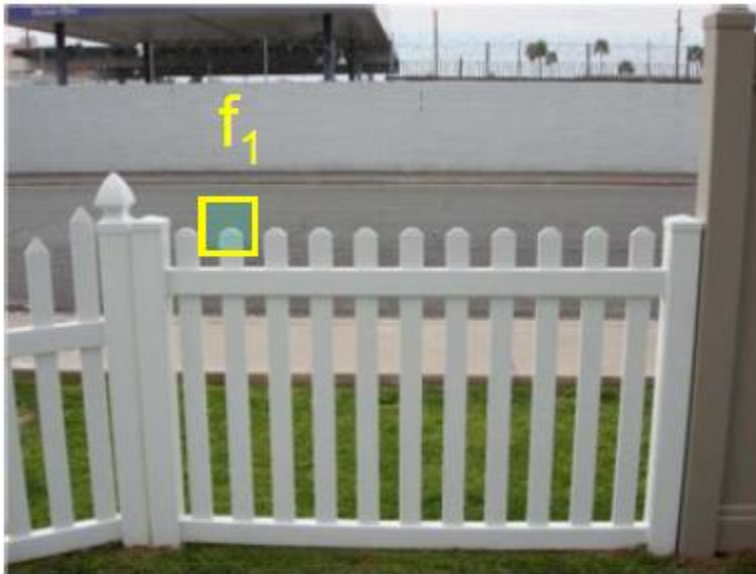


$I_2$

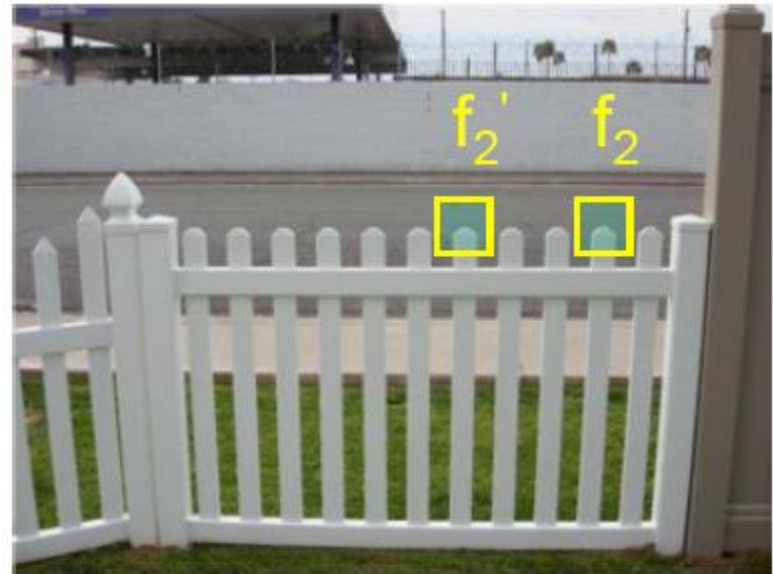
# Keypoints Matching

164

- How to define the difference between two features  $f_1$ ,  $f_2$ ?
  - Better approach: ratio distance =  $\text{SSD}(f_1, f_2) / \text{SSD}(f_1, f_2')$ 
    - $f_2$  is best SSD match to  $f_1$  in  $I_2$
    - $f_2'$  is 2nd best SSD match to  $f_1$  in  $I_2$
    - gives small values for ambiguous matches



$I_1$

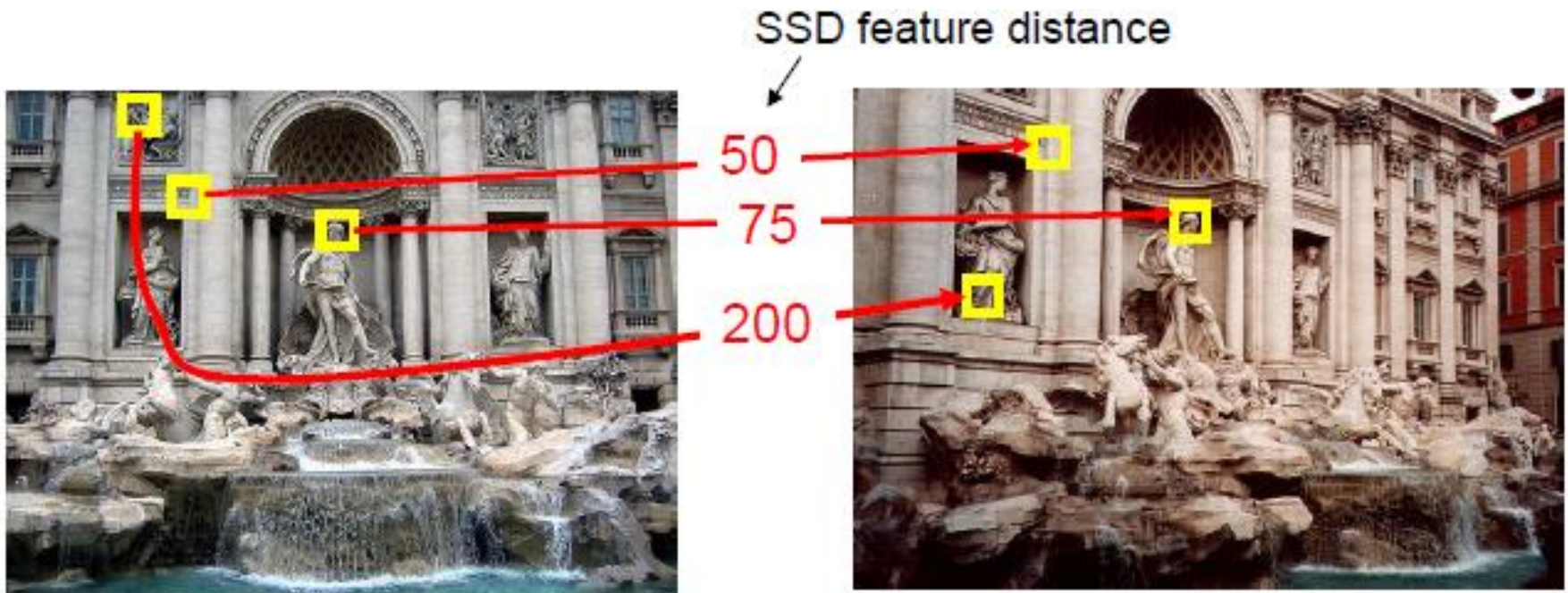


$I_2$

# Keypoints Matching

165

- Suppose we use SSD
- Small values are possible matches but how small?
- Decision rule: Accept match if  $SSD < T$ , where  $T$  is a threshold
- What is the effect of choosing a particular  $T$ ?



# Keypoints Matching

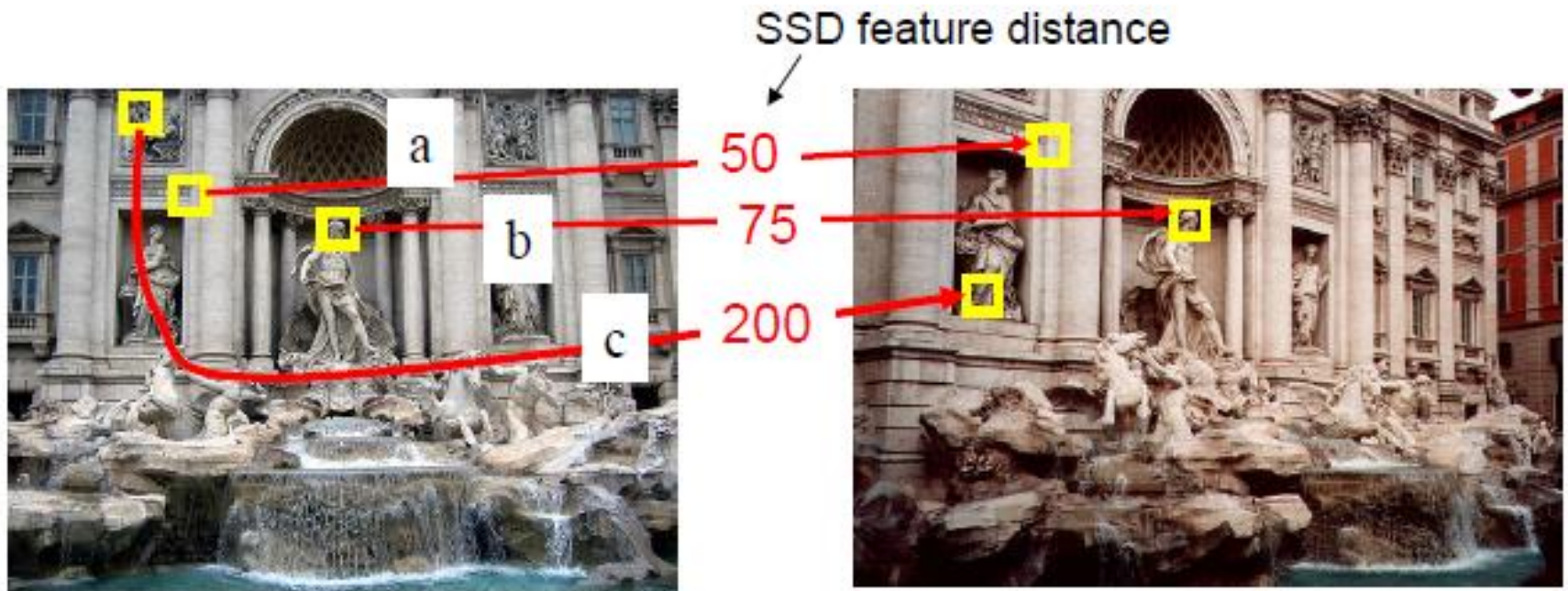
166

## Distance Decision rule:

- Accept match if  $SSD < T$

- Example: Large  $T$ ,  $T = 250 \Rightarrow$

- a, b, c are all accepted as matches
- a and b are true matches (“**true positives**”) –they are actually matches
- c is a false match (“**false positive**”) –actually not a match



# Keypoints Matching

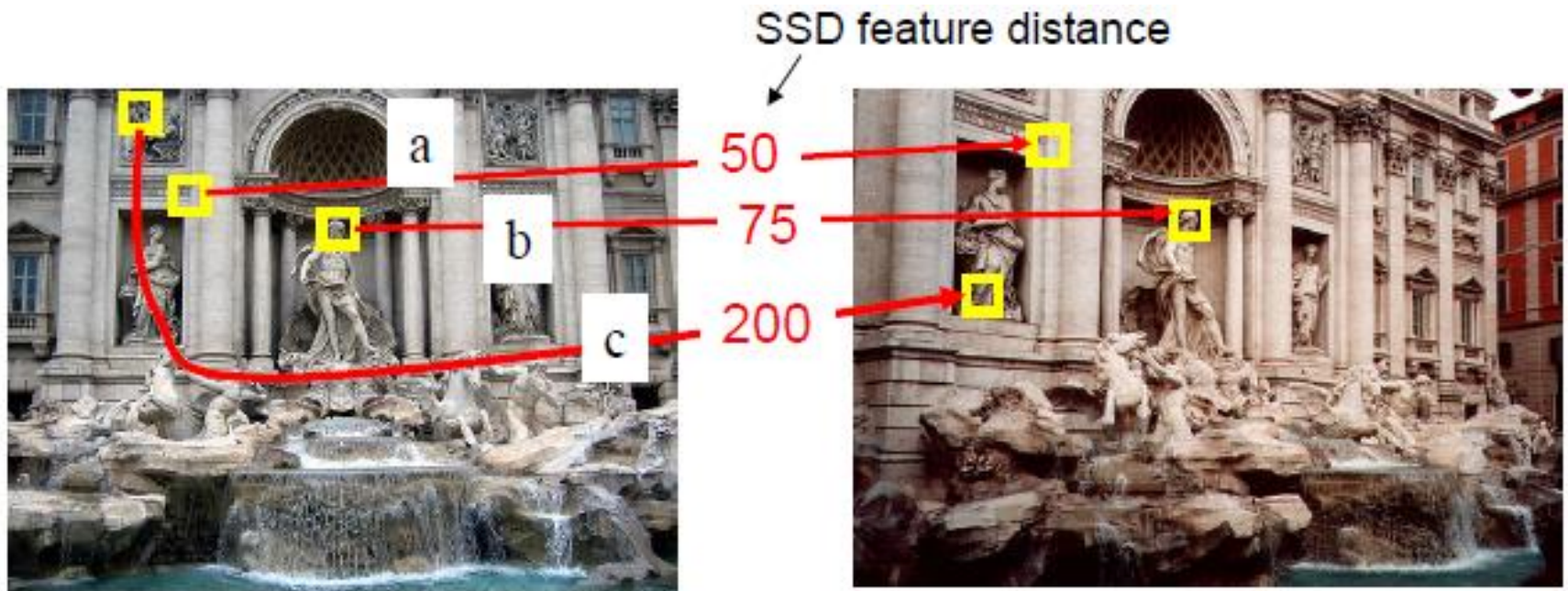
167

## Decision rule:

- Accept match if  $SSD < T$

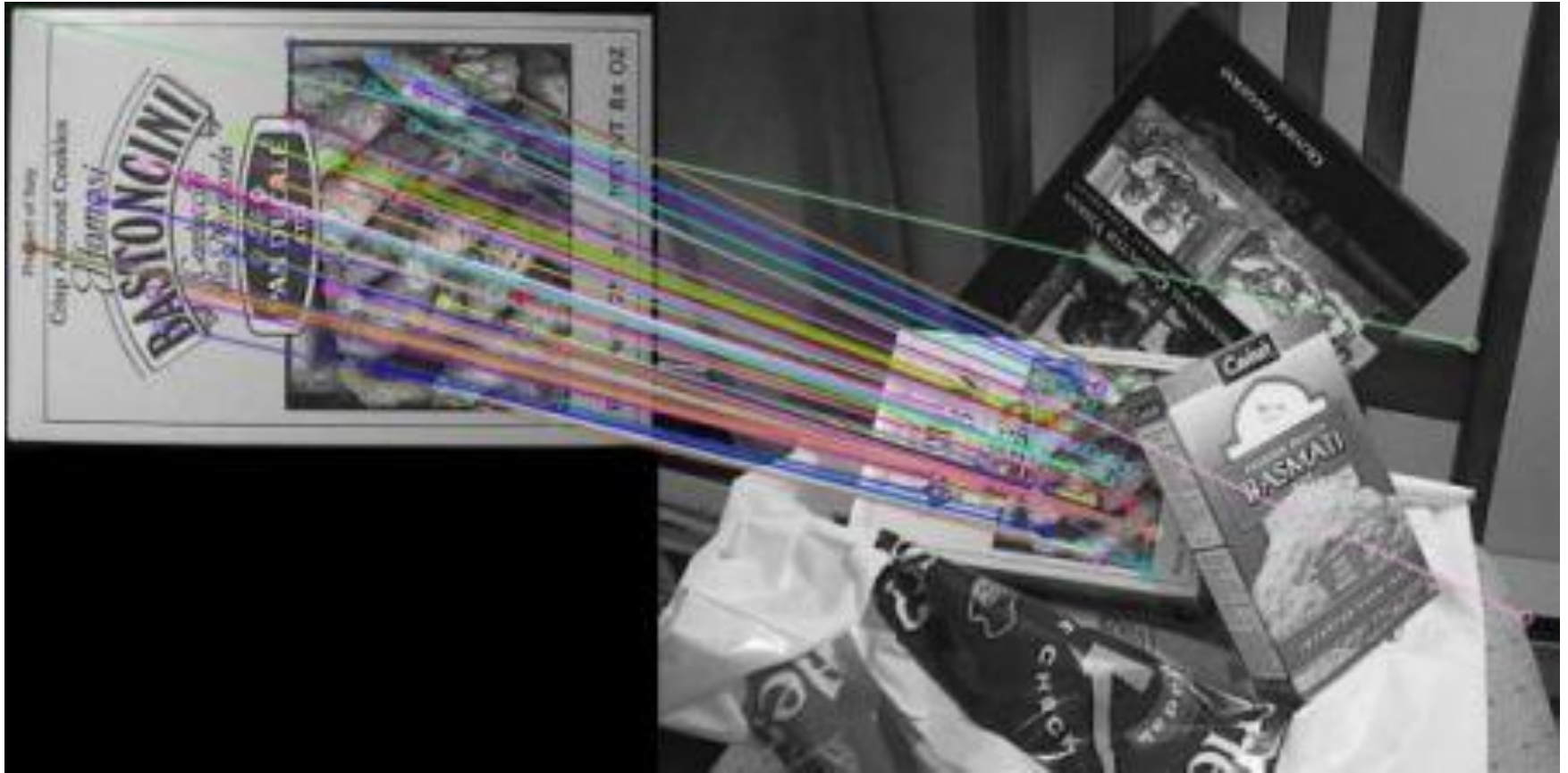
- Example: **Smaller T**,  $T = 100 \Rightarrow$

- only a and b are accepted as matches
- a and b are true matches (“true positives”)
- c is no longer a “false positive”(it is a “true negative”)



# Visualization of SIFT Key Matching using SSD

168





# Other Keypoints Detectors and Descriptors

169

## □ Detectors

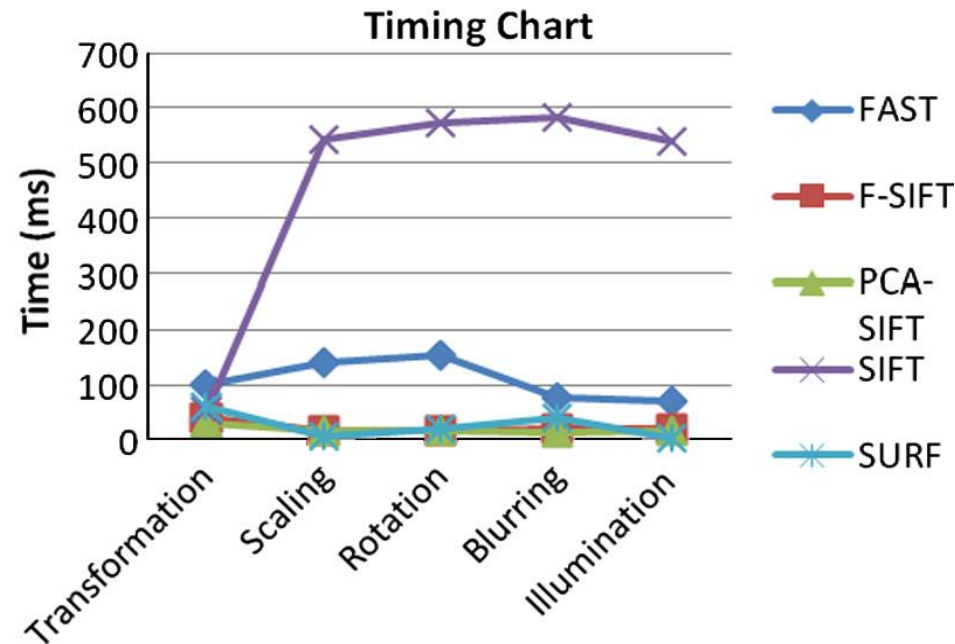
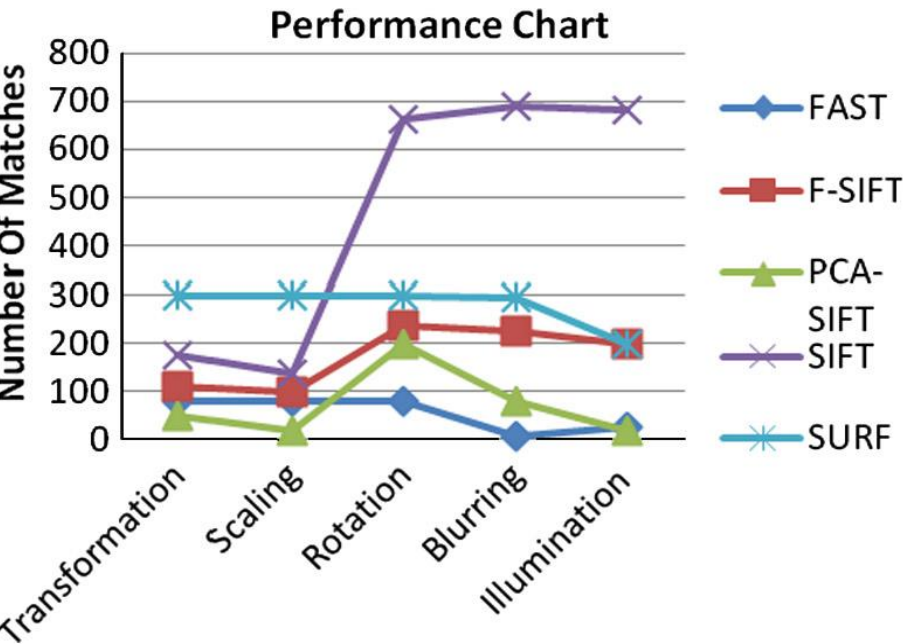
- PCA-SIFT
- SURF
- FAST
- ORB
- BRISK

## □ Descriptors

- SURF
- BREIF
- ORP
- BRISK

# SIFT compared to other algorithms

170



Methods	Timing	Transformation	Scaling	Rotation	Blurring	Illumination
FAST	Common	Common	Common	Bad	Bad	Bad
SIFT	Bad	Good	Good	Best	Best	Best
PCA-SIFT	Good	Bad	Bad	Common	Common	Bad
F-SIFT	Best	Good	Best	Good	Good	Good
SURF	Good	Best	Best	Good	Good	Good

# Some Comparisons

171

- ❑ SIFT was the most stable one except for time whereas SURF was the fastest with good results.
- ❑ The performance of the descriptor doesn't depend on the detector. Moreover, the SIFT descriptor was the best one for different image transformations except light changes.
- ❑ Binary descriptors are the best choice for time-constrained applications with good matching accuracy

# Color Features

- ▶ **Color histogram**
- ▶ **Color Moments**
- ▶ **Color coherence vector**

# Color Features

173

- The color feature is one of the most widely used visual features in image retrieval.
- Images characterized by color features have many advantages:
  - ▣ **Robustness.** The color histogram is invariant to rotation of the image on the view axis, and changes in small steps when rotated otherwise or scaled
  - ▣ **Effectiveness.** There is high percentage of relevance between the query image and the extracted matching images.
  - ▣ **Implementation simplicity.** The construction of the color histogram is a straightforward process
  - ▣ **Computational simplicity.** The histogram computation has  $O(X, Y)$  complexity for images of size  $X \times Y$ .

# Color Features

174

- Color features are defined subject to a particular color space or model.
- A number of color spaces have been used in literature, such as RGB, HSV, etc.
- Once the color space is specified, color feature can be extracted from images or regions.
- A number of important color features have been proposed in the literatures, including:
  - ▣ **Color histogram**
  - ▣ **Color moments(CM)**
  - ▣ **Color coherence vector (CCV)**
  - ▣ Color correlogram, *etc.*

# Color histogram

175

- A color histogram  $H$  for a given image is defined as a vector  $H = \{h[1], h[2], \dots, h[i], \dots, h[N]\}$ 
  - Where  $i$  represents a color in the color histogram,
  - $h[i]$  is the number of pixels in color  $i$  in that image,
  - and  $N$  is the number of bins in the color histogram, i.e., the number of colors in the adopted color model.
- In order to compare images of different sizes, color histograms should be normalized.
- Can be used to Measures the similarity of
  - images
  - speech
  - music
- Issue:
  - how to capture perceptual similarity of an image

# Color histogram

176

- The standard measure of similarity used for color histograms:
  - ▣ A color histogram  $H(i)$  is generated for each image  $h$  in the database (feature vector),
  - ▣ The histogram is *normalized so that its sum equals unity* (removes the size of the image),
  - ▣ The histogram is then stored in the database,
  - ▣ Now suppose we select a *model image* (the new image to match against all possible targets in the database).



# Color histogram

177

## □ Histogram distance measures

$L_1$  distance (Manhattan distance)

$$d_1(H, L) = \sum_{k=1}^K |h_k - l_k|$$

$L_2$  distance (euclidian distance)

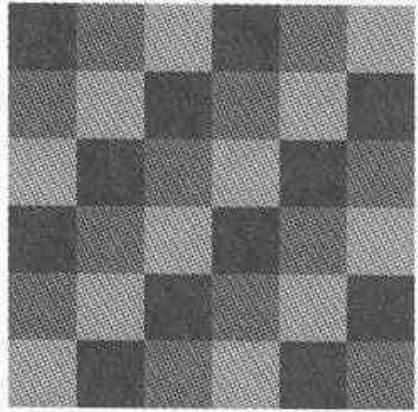
$$d_2(H, L) = \sqrt{\sum_{k=1}^K |h_k - l_k|^2}$$

$L_\infty$  distance (maximum distance)

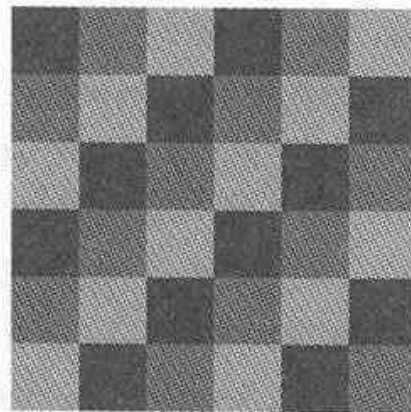
$$d_\infty(H, L) = \max_k (|h_k - l_k|)$$

# Example for potential problem with histogram distance

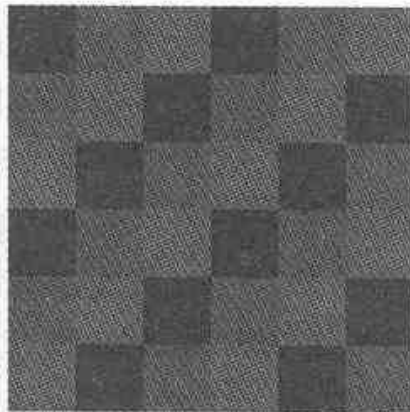
178



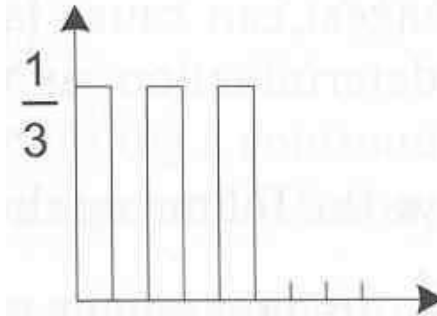
(a)



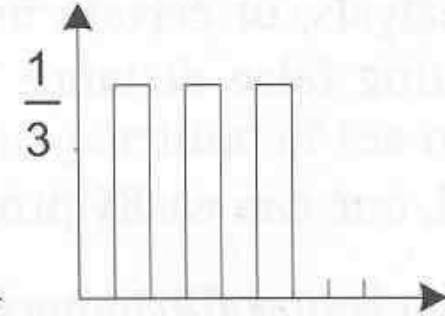
(b)



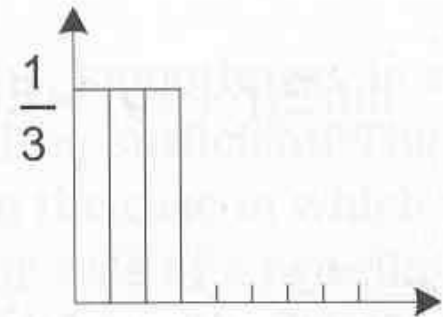
(c)



(a)



(b)



(c)

# Distances of the three checkerboard images

179

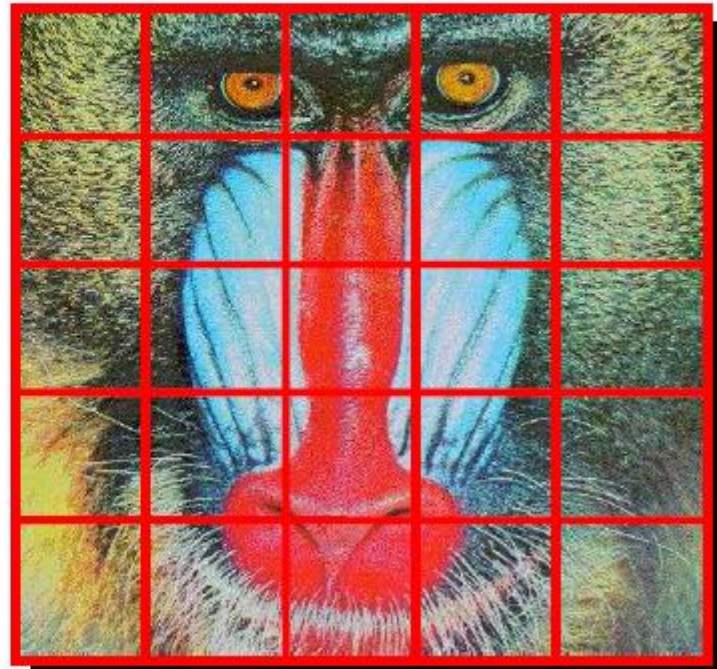
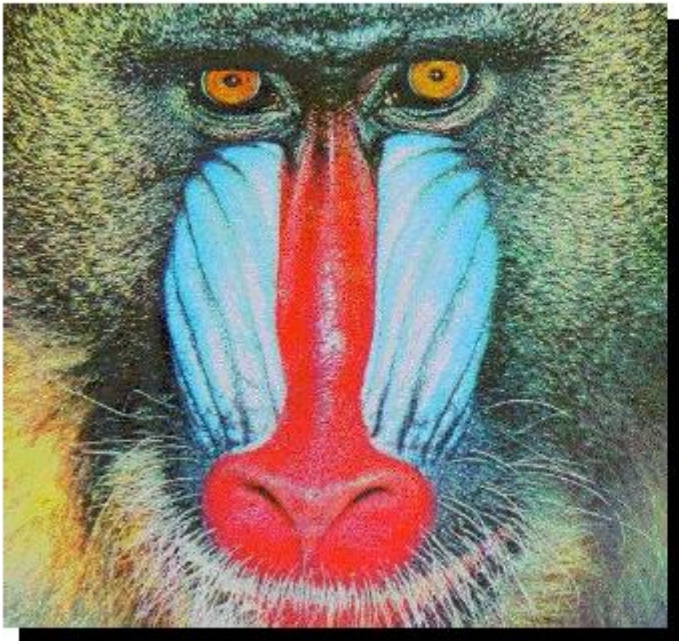
Distance type	$d(a,b)$	$d(a,c)$
$L_1$	2	$\sim 0.67$
$L_2$	$\sim 0.82$	$\sim 0.47$
$L_\infty$	$\sim 0.33$	$\sim 0.33$

None of the distances captures perceptual similarity

# Another Issue: loss of regional information

180

Partition the image  
One histogram per region



# Color Moments

181

- Provide measurement for color similarity between images. These values of similarity can then be compared to the values of images indexed in a database for tasks like image retrieval.
- The basis of color moments lays in the assumption that the distribution of color in an image can be interpreted as a probability distribution.
- Probability distributions are characterized by a number of unique moments (e.g. Normal distributions are differentiated by their mean and variance).
- It therefore follows that if the color in an image follows a certain probability distribution, the moments of that distribution can then be used as features to identify that image based on color.
- The first order (mean), the second (variance) and the third order (skewness) color moments have been proved to be efficient and effective in representing color distributions of images.

# Color Moments

182

## **MOMENT 1 – Mean :**

$$E_i = \sum_{j=1}^{j=1} \frac{1}{N} p_{ij}$$

Mean can be understood as the average color value in the image.

## **MOMENT 2 - Standard Deviation :**

$$\sigma_i = \sqrt{\left( \frac{1}{N} \sum_{j=1}^{j=1} (p_{ij} - E_i)^2 \right)}$$

The standard deviation is the square root of the variance of the distribution.

## **MOMENT 3 – Skewness :**

$$s_i = \sqrt[3]{\left( \frac{1}{N} \sum_{j=1}^{j=1} (p_{ij} - E_i)^3 \right)}$$

Skewness can be understood as a measure of the degree of asymmetry in the distribution.

# Color Moments

A function of the similarity between two image distributions is defined as the sum of the weighted differences between the moments of the two distributions. Formally this is:

$$d_{mom}(H, I) = \sum_{i=1}^r w_{i1} |E_i^1 - E_i^2| + w_{i2} |\sigma_i^1 - \sigma_i^2| + w_{i3} |s_i^1 - s_i^2|$$

Where:

- $(H, I)$  : are the two image distributions being compared
- $i$  : is the current channel index (e.g. 1 = H, 2 = S, 3 = V)
- $r$  : is the number of channels (e.g. 3)
- $E_i^1, E_i^2$  : are the first moments (mean) of the two image distributions
- $\sigma_i^1, \sigma_i^2$  : are the second moments (std) of the two image distributions
- $s_i^1, s_i^2$  : are the third moments (skewness) of the two image distributions
- $w_i$  : are the weights for each moment (described below)

Pairs of images can be ranked based on their  $d_{mom}$  values. Those with greater values are ranked lower and considered less similar than those with a higher rank and lower  $d_{mom}$  values.

# Color Moments example

184



Index Image



Test Image 1



Test Image 2

$$d_{mom}(Index, Test1) = 0.5878$$

$$d_{mom}(Index, Test2) = 1.5585$$



# Color Coherence Vector

185

- Color's coherence is the degree to which pixels of that color are members of large similarly-colored regions.
- The images Below have similar color histograms, despite their different appearances.
  - ▣ The color red appears in both images in approximately the same quantities.



# Color Coherence Vector

186

- ❑ Coherency measure classifies pixels as either coherent or incoherent.
- ❑ Coherent pixels are a part of some sizable contiguous region, while incoherent pixels are not.
- ❑ A color coherence vector represents this classification for each color in the image.
- ❑ CCV's prevent coherent pixels in one image from matching incoherent pixels in another.
- ❑ This allows fine distinctions that cannot be made with color histograms.

# Color Coherence Vector

187

## □ Computing CCV's

1. First blur the image. This eliminates small variations between neighboring pixels.
2. Then discretize the color space, such that there are only  $n$  distinct colors in the image.
3. Determine the pixel groups by computing connected components.
  - Each pixel will belong to exactly one connected component.
  - We classify pixels as either coherent or incoherent depending on the size in pixels of its connected component.
  - A pixel is coherent if the size of its connected component exceeds a fixed value  $T$ ; otherwise, the pixel is incoherent.

# Color Coherence Vector

188

## □ Computing CCV's

- For a given discretized color, some of the pixels with that color will be coherent and some will be incoherent.
  - Let us call the number of coherent pixels of the  $j$ 'th discretized color  $\alpha_j$  and the number of incoherent pixels  $\beta_j$ .
  - Clearly, the total number of pixels with that color is  $\alpha_j + \beta_j$ , for each color we compute the pair  $(\alpha_j + \beta_j)$  which we will call the *coherence pair* for the  $j$ 'th color.
  - The *color coherence vector* for the image consists of   
 **$((\alpha_1 + \beta_1), \dots, (\alpha_n + \beta_n))$**
  - This is a vector of coherence pairs, one for each discretized color

# Color Coherence Vector

189

## □ Comparing CCV's

- Consider two images  $I$  and  $I'$ , together with their CCV's  $G_I$  and  $G_{I'}$ :

$$G_I = \langle (\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n) \rangle$$

and

$$G_{I'} = \langle (\alpha'_1, \beta'_1), \dots, (\alpha'_n, \beta'_n) \rangle$$

$$\Delta_G = \sum_{j=1}^n |(\alpha_j - \alpha'_j)| + |(\beta_j - \beta'_j)|.$$

# Color Features Technique Summary

190

Color method	Pros.	Cons.
Histogram	Simple to compute, intuitive	High dimension, no spatial info, sensitive to noise
CM	Compact, robust	Not enough to describe all colors, no spatial info
CCV	Spatial info	High dimension, high computation cost
Correlogram	Spatial info	Very high computation cost, sensitive to noise, rotation and scale
DCD	Compact, robust, perceptual meaning	Need post-processing for spatial info
CSD	Spatial info	Sensitive to noise, rotation and scale
SCD	Compact on need, scalability	No spatial info, less accurate if compact

CCV: color coherence vector

DCD: dominant color descriptor

CSD: color structure descriptor

SCD: scalable color descriptor respectively

# Shape Features

- ▶ **Shape Descriptor**
  - ▶ Contour based
  - ▶ Region Based
- ▶ **Shape Matching**

# Shape features

192

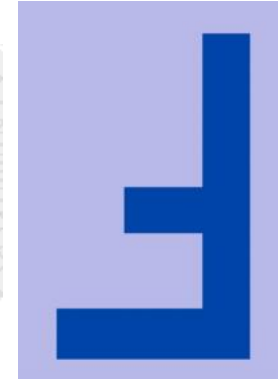
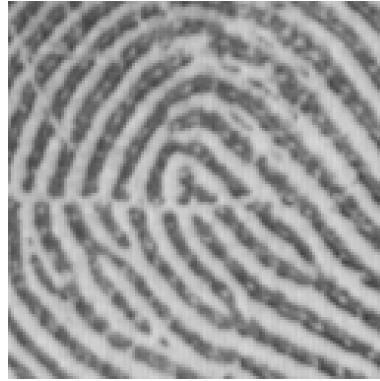
- Shape is probably the most important property that is perceived about objects. It allows to predict more facts about an object than other features, e.g. color (Palmer 1999)
- Thus, recognizing shape is crucial for object recognition. In some applications it may be the only feature present, e.g. logo recognition
- Shape content description is difficult to define because measuring the similarity between shapes is difficult.
- In order to extract object features, we need an image that has undergone image segmentation and any necessary morphological filtering.
- This will provide us with a clearly defined object which can be labeled and processed independently.



# Why Shape ?

193







These objects are recognized by...



# Why Shape ?

194

These objects are recognized by...

	Texture	Color	Shape
	X	X	
	X		X
			X
			X
			X
			X

# Shape feature categories of applications:

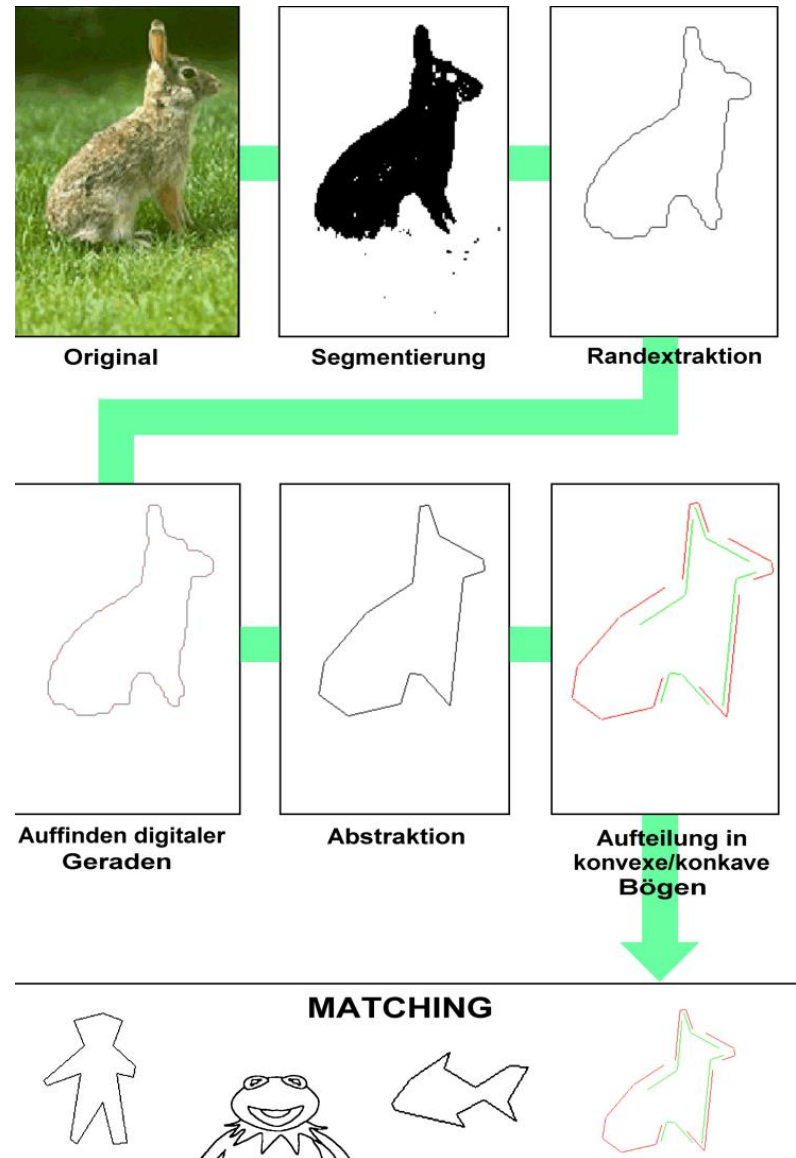
195

- **Shape retrieval:** searching for all shapes in a typically large database of shapes that are similar to a query shape. Usually all shapes within a given distance from the query are determined or the first few shapes that have the smallest distance.
- **Shape recognition and classification:** determining whether a given shape matches a model sufficiently, or which of representative class is the most similar.
- **Shape alignment and registration:** transforming or translating one shape so that it best matches another shape, in whole or in part.
- **Shape approximation and simplification:** constructing a shape with fewer elements (points, segments, triangles, etc.), so that it is still similar to the original.

# Object Recognition by Shape

196

- Source: 2D image of a 3D object
- Object Segmentation
- Contour Extraction
- Contour Cleaning
- Contour Segmentation
- Matching: Correspondence of Visual Parts



# Typical Issues

197

- ❑ Object segmentation and extraction
- ❑ How to describe object
- ❑ What is the matching techniques
- ❑ Occlusion
- ❑ Noise

# Shape descriptor

198

- Shape descriptor is a set of numbers that are produced to represent a given shape feature.
- A descriptor attempts to quantify the shape in ways that agree with human intuition.
- Good retrieval accuracy requires a shape descriptor to be able to effectively find perceptually similar shapes from a database.
- Usually, the descriptors are in the form of a vector.

# Shape descriptor

199

- Shape descriptors should meet the following requirements:
  - ▣ The descriptors should be as complete as possible to represent the content of the information items.
  - ▣ The descriptors should be represented and stored compactly. The size of a descriptor vector must not be too large.
  - ▣ The computation of the similarity or the distance between descriptors should be simple; otherwise the execution time would be too long.
  - ▣ Descriptors are invariant of variations of scale, rotation and translation whenever possible

# Shape features categories

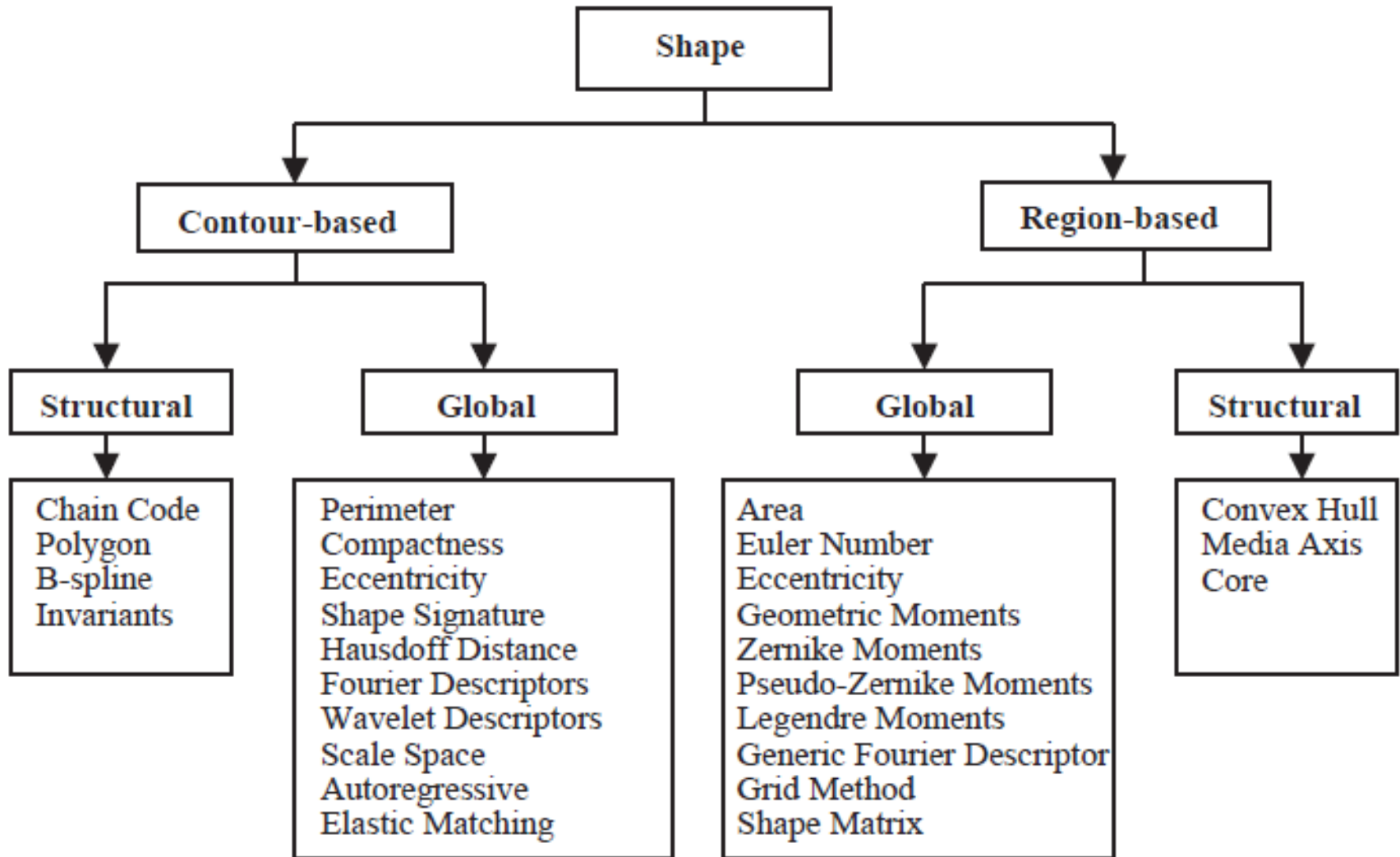
200

- Shape descriptors can be divided into two main categories:
  - ▣ **Region-based methods** use the whole area of an object for shape description
  - ▣ **Contour-based methods** use only the information present in the contour of an object.
- Under each class, the different methods are further divided into **structural approaches and global approaches**. This subclass is based on whether the shape is represented as a whole or represented by segments/sections (primitives).
- These approaches can be further distinguished into **space domain and transform domain**, based on whether the shape features are derived from the spatial domain or from the transformed domain.



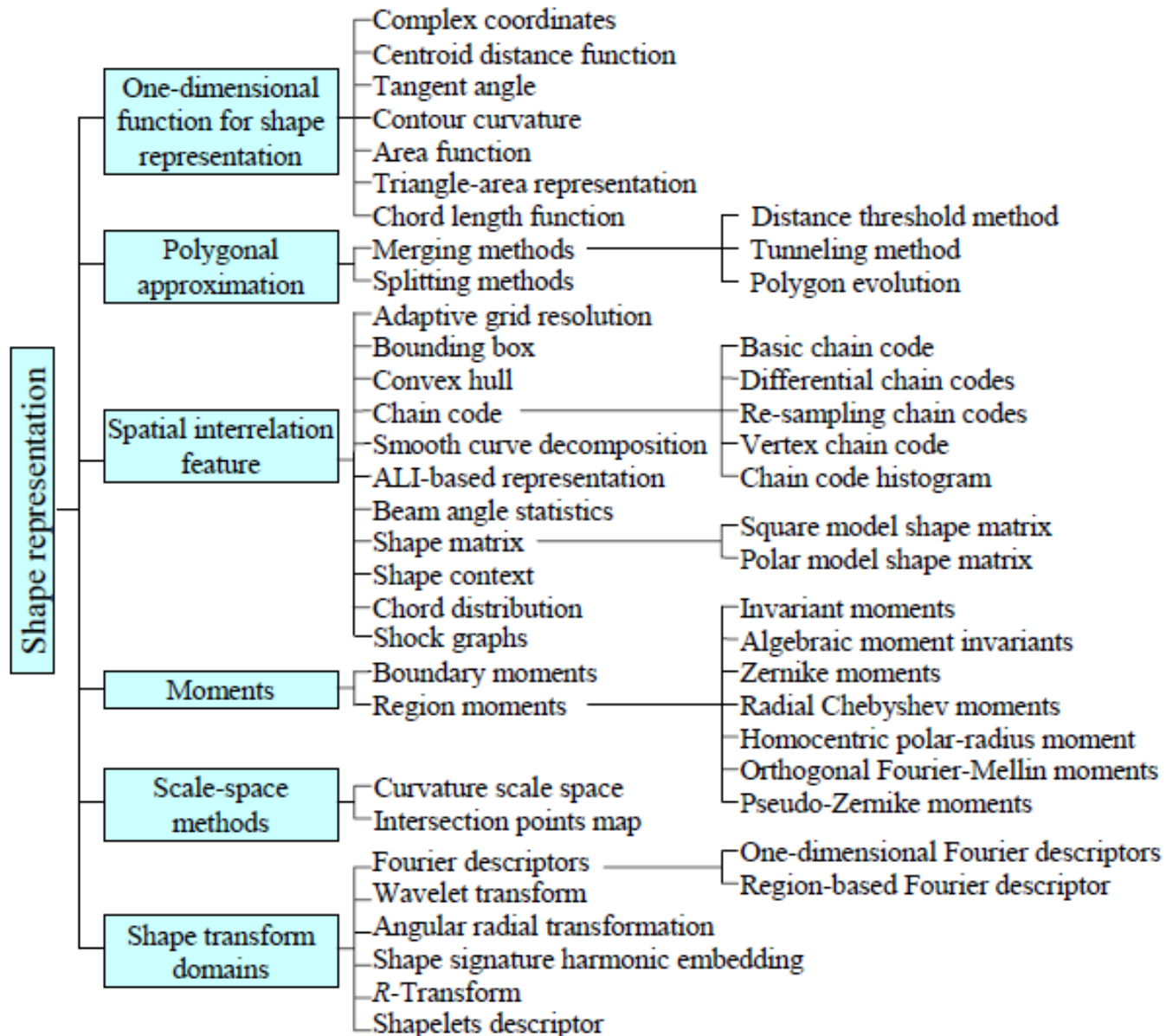
# Shape features categories

201



# Shape features categories

202



# Boundary Descriptors

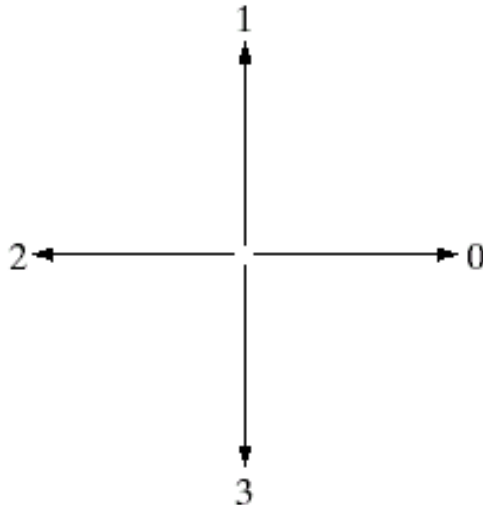
- **Chain Code**
- **Polygon Approximation**
- **Shape number**
- **Fourier descriptor**
- **Statistical Moments**

# Chain Codes

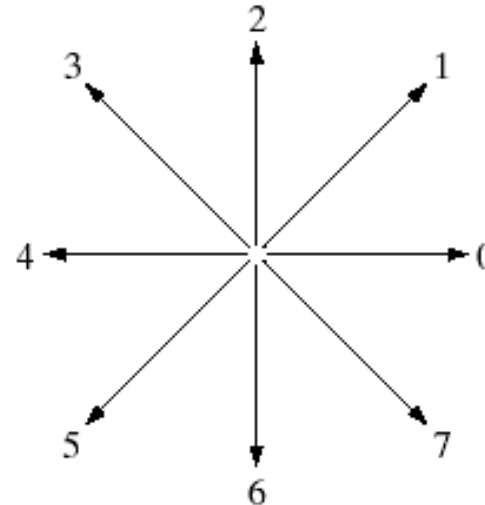
## ❑ Why we focus on a boundary?

The boundary is a good representation of an object shape and also requires a few memory.

❑ **Chain codes:** represent an object boundary by a connected sequence of straight line segments of specified length and direction.



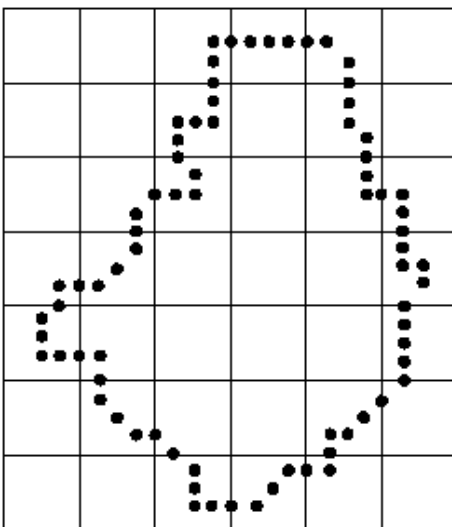
4-directional chain code



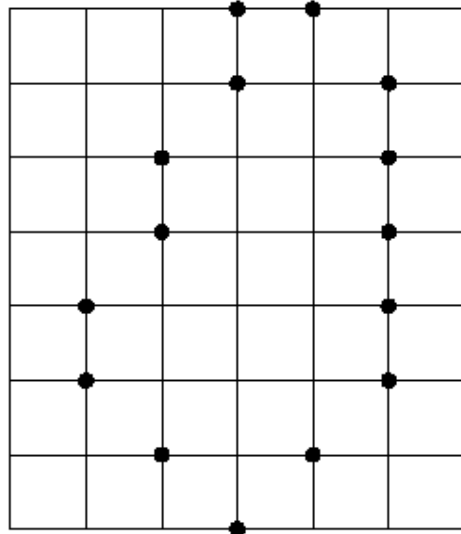
8-directional chain code

# Examples of Chain Codes

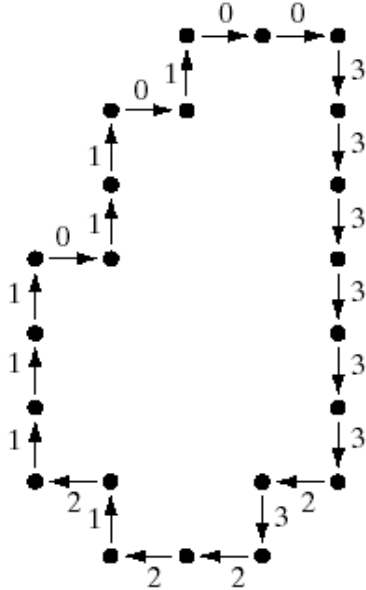
Object boundary (resampling)



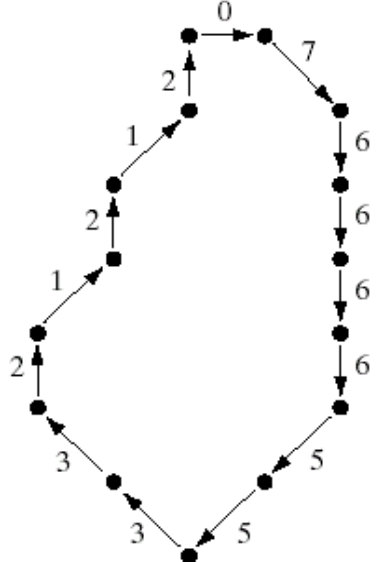
Boundary vertices



4-directional chain code



8-directional chain code



# Chain Codes

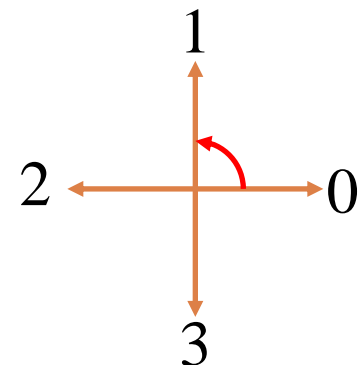
206

- Not scale invariant
  - ▣ You can provide several chain codes of the same object at different “resolutions”
- Translation invariant
  - ▣ Note that this is different than a chain of (x,y) coordinates
  - ▣ We are encoding the boundary itself
- Codes are sensitive to noise
  - ▣ If your boundary has some noise, this will show up in the chain code
  - ▣ One solution
    - Resample using a larger grid spacing
    - Also provides a more compact representation
- Chain Code depends on the starting point
  - ▣ We can normalize the chain code to address this problem
    - Assume the chain is a circular sequence
    - (given a chain of 1 to N codes ;  $N+1 = 1$ )
    - Redefine the starting point such that we generate an integer of smallest magnitude

# The First Difference of a Chain Codes

- Chain code depends on orientation
  - a rotation results in a different chain code
  - One solution
    - Use the “first difference” of the chain code instead of the code itself
    - The difference is obtained by simply counting (counter-clockwise) the number of directions that separate two adjacent elements

**Example:** Chain code : The first difference



0 → 1	1
0 → 2	2
0 → 3	3
2 → 3	1
2 → 0	2
2 → 1	3

**Example:**

- a chain code: 10103322
- The first difference = 3133030
- Treating a chain code as a circular sequence, we get the first difference = 33133030

The first difference is rotational invariant.

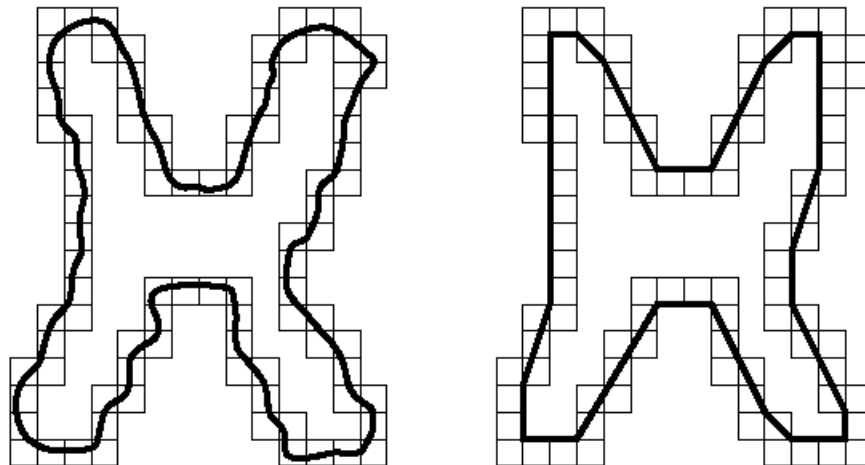
# Representation Polygonal Approximations

- **Polygonal approximations**: to represent a boundary by straight line segments, and a closed path becomes a polygon.
- The number of straight line segments used determines the accuracy of the approximation.
- Only the minimum required number of sides necessary to preserve the needed shape information should be used (**Minimum perimeter polygons**).
- A larger number of sides will only add noise to the model.

a b

**FIGURE 11.3**

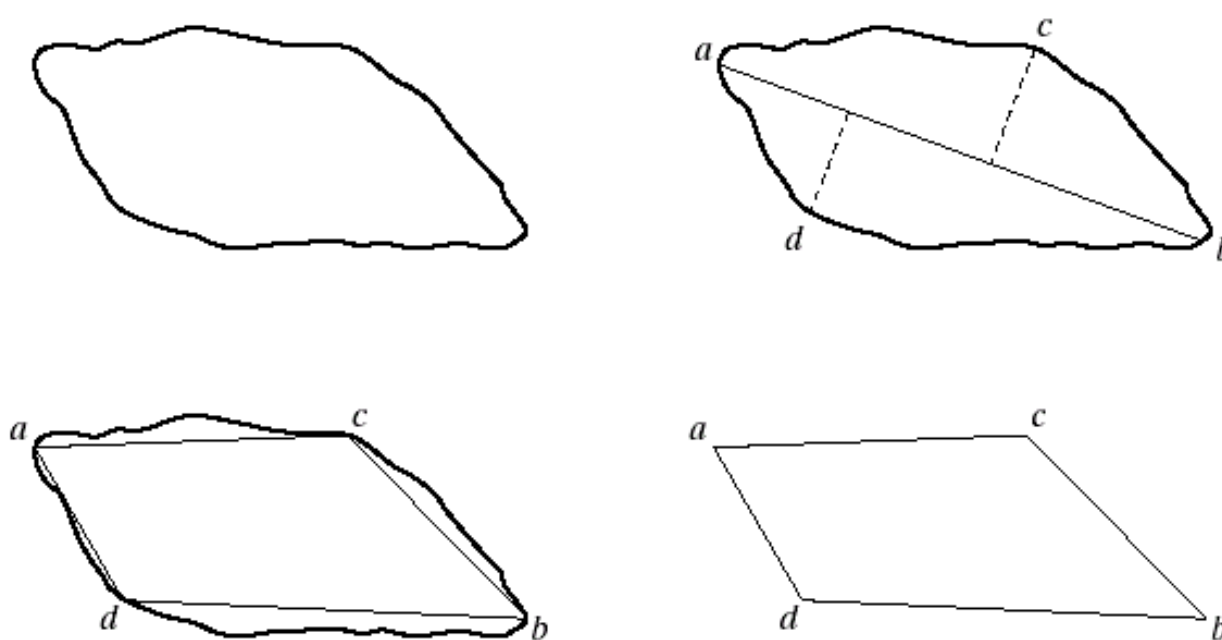
(a) Object boundary enclosed by cells.  
(b) Minimum perimeter polygon.





# Representation Polygonal Approximations

- Minimum perimeter polygons: (Merging and splitting)
  - ▣ Merging and splitting are often used together to ensure that vertices appear where they would naturally in the boundary.
  - ▣ A least squares criterion to a straight line is used to stop the processing.



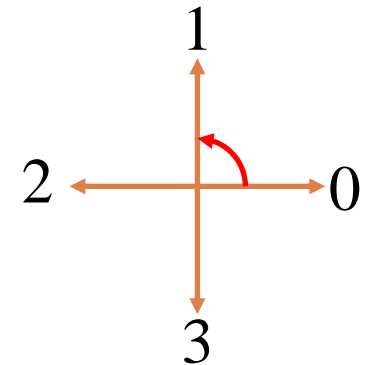
**FIGURE 11.4**

(a) Original boundary.  
(b) Boundary divided into segments based on extreme points. (c) Joining of vertices.  
(d) Resulting polygon.

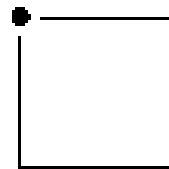
# Shape Number

**Shape number of the boundary definition:**  
the first difference of smallest magnitude

**The order n of the shape number:** the  
number of digits in the sequence



Order 4

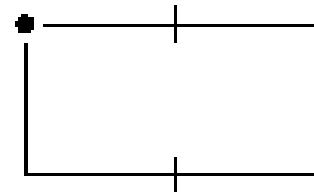


Chain code: 0 3 2 1

Difference: 3 3 3 3

Shape no.: 3 3 3 3

Order 6



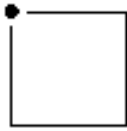
Chain code: 0 0 3 2 2 1

Difference: 3 0 3 3 0 3

Shape no.: 0 3 3 0 3 3

# Shape Number (cont.)

Order 4

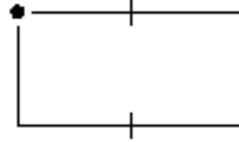


Chain code: 0 3 2 1

Difference: 3 3 3 3

Shape no.: 3 3 3 3

Order 6



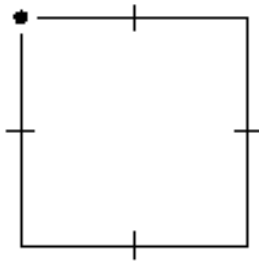
Chain code: 0 0 3 2 2 1

Difference: 3 0 3 3 0 3

Shape no.: 0 3 3 0 3 3

Shape numbers of order  
4, 6 and 8

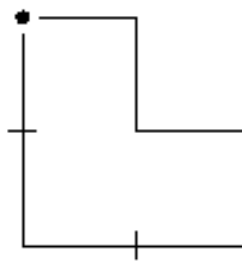
Order 8



Chain code: 0 0 3 3 2 2 1 1

Difference: 3 0 3 0 3 0 3 0

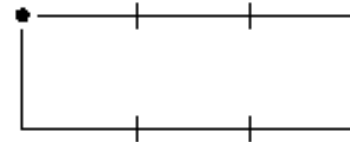
Shape no.: 0 3 0 3 0 3 0 3



Chain code: 0 3 0 3 2 2 1 1

Difference: 3 3 1 3 3 0 3 0

Shape no.: 0 3 0 3 3 1 3 3

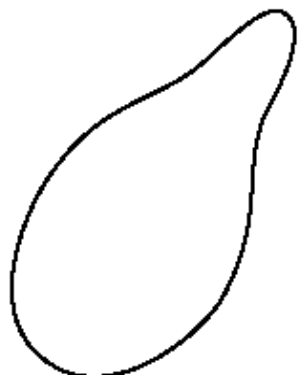


Chain code: 0 0 0 3 2 2 2 1

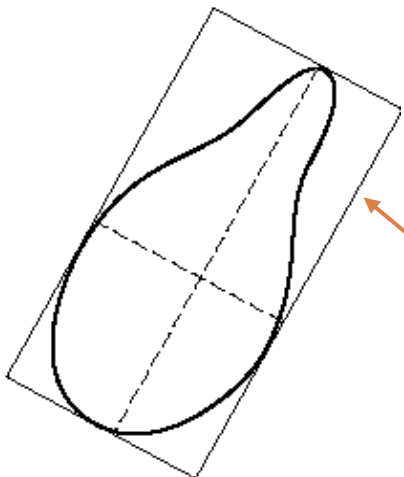
Difference: 3 0 0 3 3 0 0 3

Shape no.: 0 0 3 3 0 0 3 3

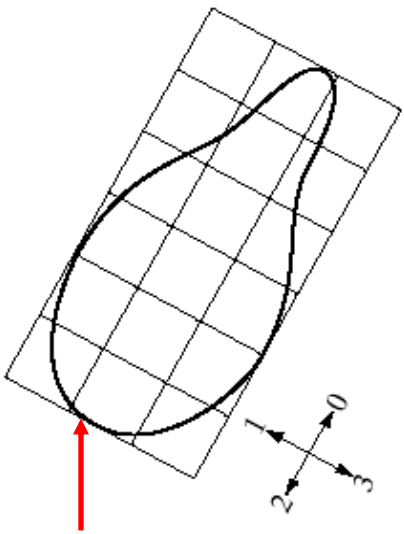
# Example: Shape Number



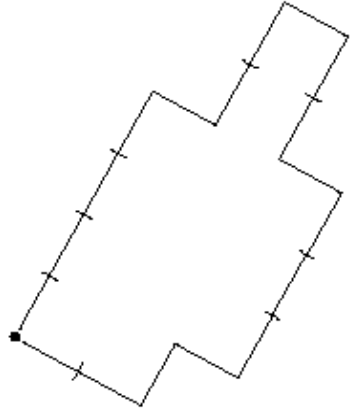
1. Original boundary



2. Find the smallest rectangle that fits the shape



3. Create grid



4. Find the nearest Grid.

Chain code:  
0 0 0 3 0 0 3 2 2 3 2 2 2 1 2 1 1

First difference:  
3 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0

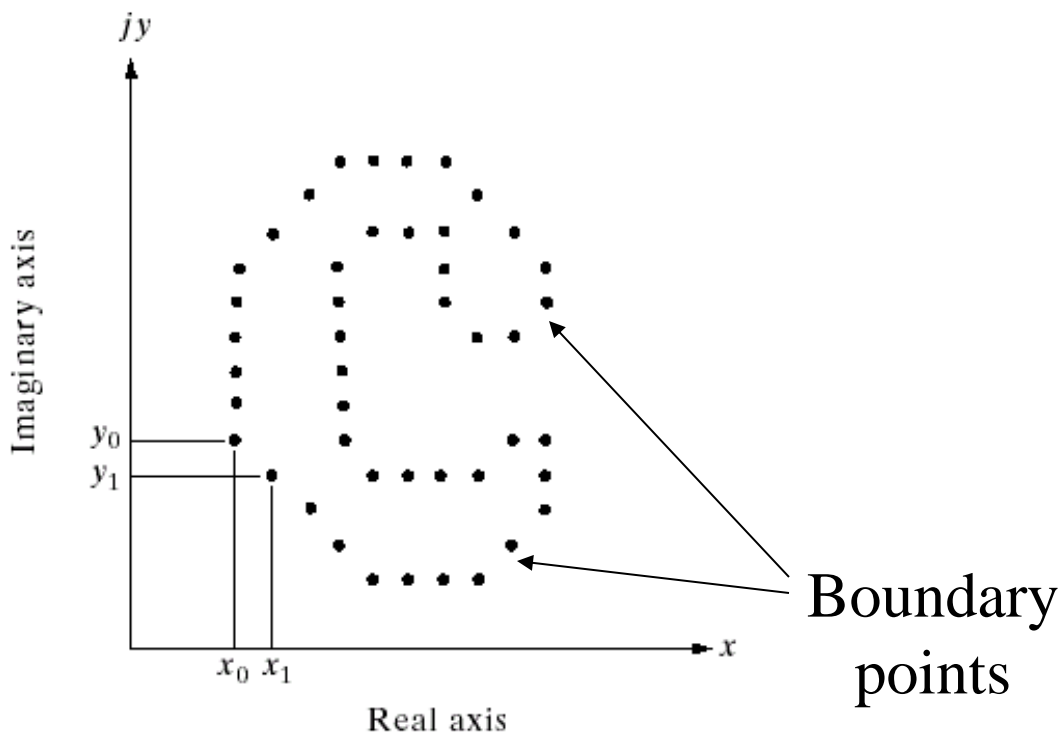
Shape No.  
0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0 3

# Fourier Descriptor

**Fourier descriptor:** view a coordinate  $(x,y)$  as a complex number ( $x = \text{real part}$  and  $y = \text{imaginary part}$ ) then apply the Fourier transform to a sequence of boundary points.

Let  $s(k)$  be a coordinate of a boundary point  $k$  :

$$s(k) = x(k) + jy(k)$$



Fourier descriptor :

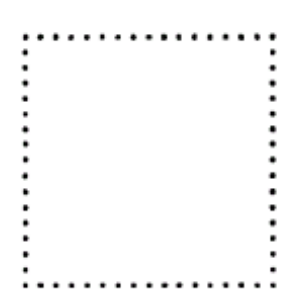
$$a(u) = \frac{1}{K} \sum_{k=0}^{K-1} s(k) e^{-2\pi i k / K}$$

Reconstruction formula

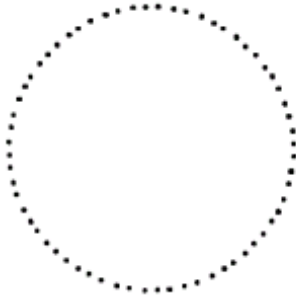
$$s(k) = \frac{1}{K} \sum_{u=0}^{K-1} a(u) e^{2\pi i k u / K}$$

# Example: Fourier Descriptor

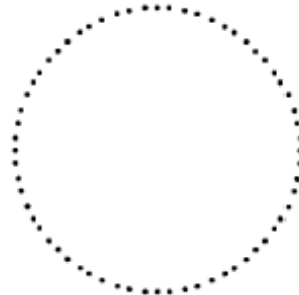
Examples of reconstruction from Fourier descriptors



Original ( $K = 64$ )



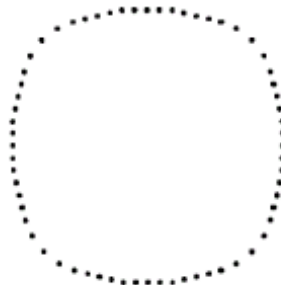
$P = 2$



$P = 4$



$P = 8$



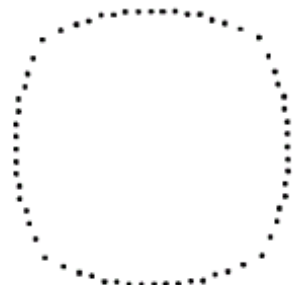
$P = 16$



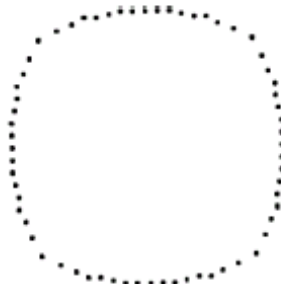
$P = 24$



$P = 32$



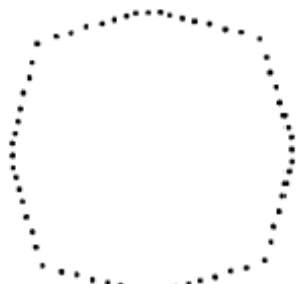
$P = 40$



$P = 48$



$P = 56$



$P = 61$



$P = 62$

$$\hat{s}(k) = \frac{1}{K} \sum_{u=0}^{P-1} a(u) e^{2\pi i k u / K}$$

$P$  is the number of Fourier coefficients used to reconstruct the boundary

# Statistical Moments

Definition: the  $n^{\text{th}}$  moment

$$\mu_n(r) = \sum_{i=0}^{K-1} (r_i - m)^n g(r_i)$$

where

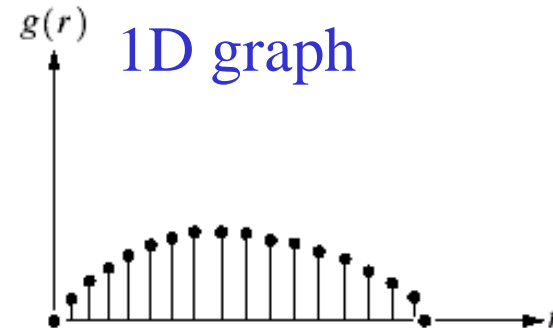
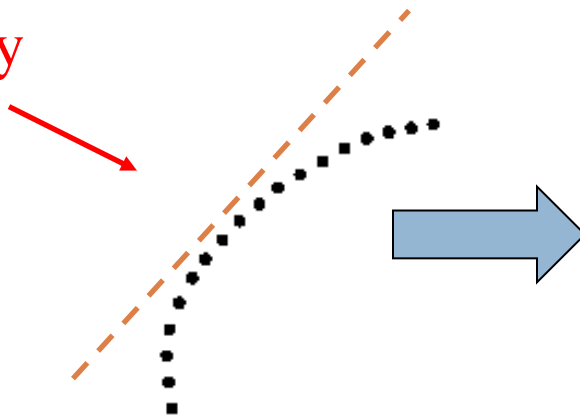
$$m = \sum_{i=0}^{K-1} r_i g(r_i)$$

Example of moment:

The first moment = mean

The second moment = variance

Boundary  
segment



1. Convert a boundary segment into 1D graph
2. View a 1D graph as a PDF function
3. Compute the  $n^{\text{th}}$  order moment of the graph

# Regional Descriptors

216

## □ **Some simple regional descriptors**

### □ **Area** of the region

- Number of pixels in the region

### □ **Perimeter**

- Length of its boundary

### □ **Compactness**

- $(\text{perimeter}^2)/\text{area}$
- Compactness is invariant to translation, rotation, and scale
- It is minimal for a disk-shaped region

## □ The previously mentioned regional descriptors are often used with “blob” detection algorithms

### □ Especially “area” and compactness

- For example, consider that you are looking for circles with radius of 10 pixels



# Reginal Descriptors

217

- Topological
- Texture
  - ▣ Statistical
  - ▣ Structural
  - ▣ Spectral

# Topological Descriptors

- Topology = The study of the properties of a figure that are unaffected by any deformation
- Use to describe holes and connected components of the region

Euler number ( $E$ ):

$$E = C - H$$

$C$  = the number of connected components

$H$  = the number of holes

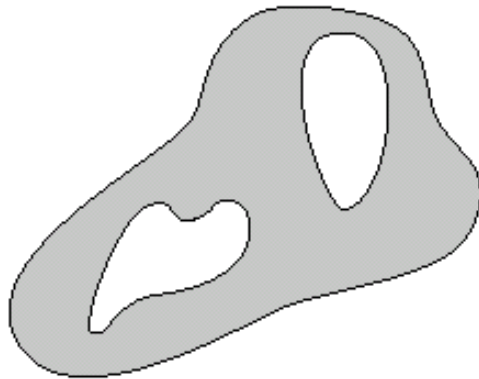


FIGURE 11.17 A region with two holes.

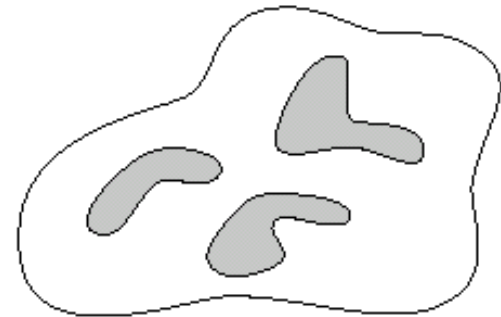
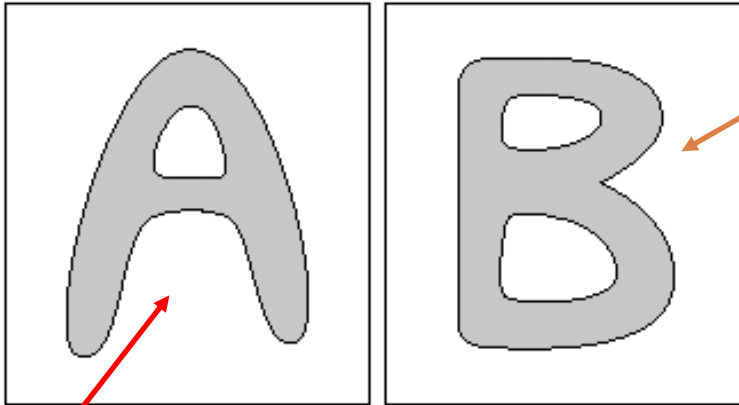


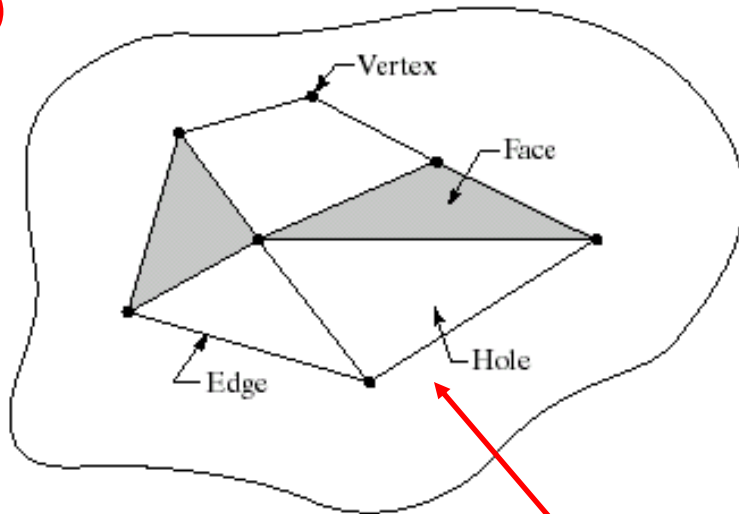
FIGURE 11.18 A region with three connected components.

# Topological Descriptors (cont.)



$E = 0$

$E = -1$



$E = -2$

Euler Formula

$$V - Q + F = C - H = E$$

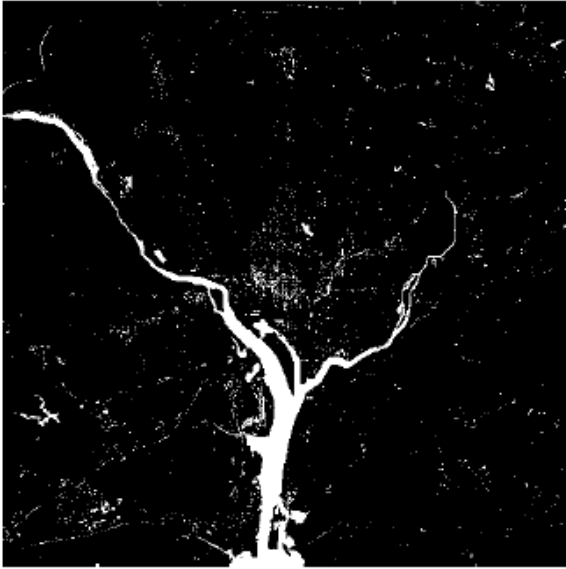
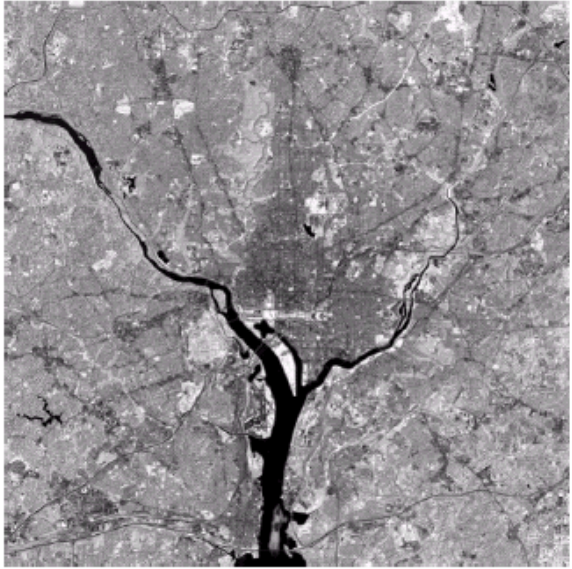
$V$  = the number of vertices

$Q$  = the number of edges

$F$  = the number of faces

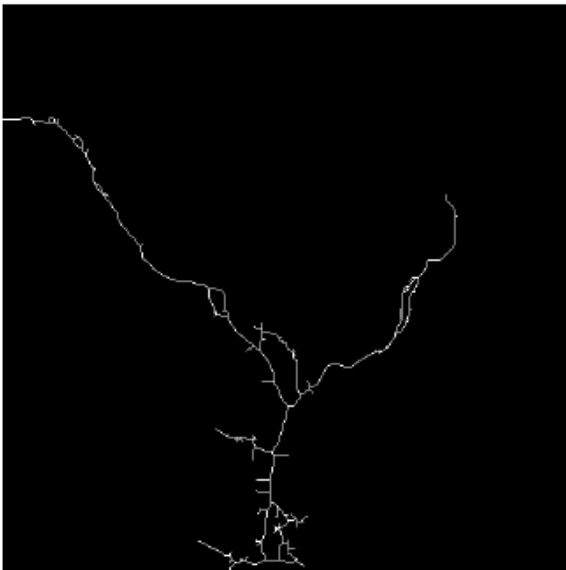
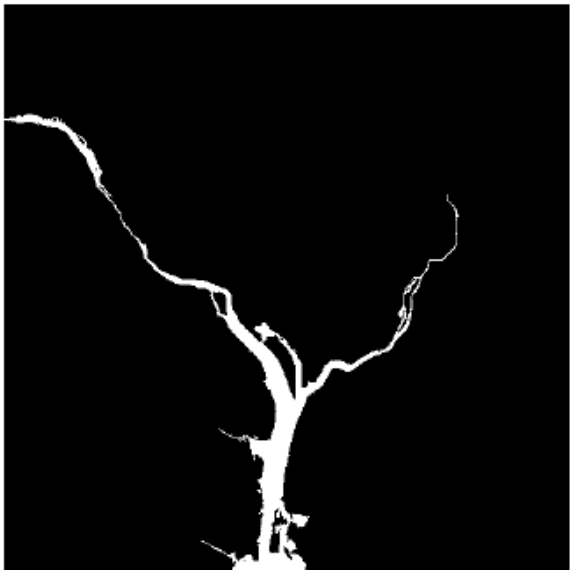
# Example: Topological Descriptors

**Original image:**  
Infrared image  
Of Washington  
D.C. area



After intensity  
Thresholding  
(1591 connected  
components  
with 39 holes)  
Euler no. = 1552

The largest  
connected  
area  
(8479 Pixels)  
(Hudson river)

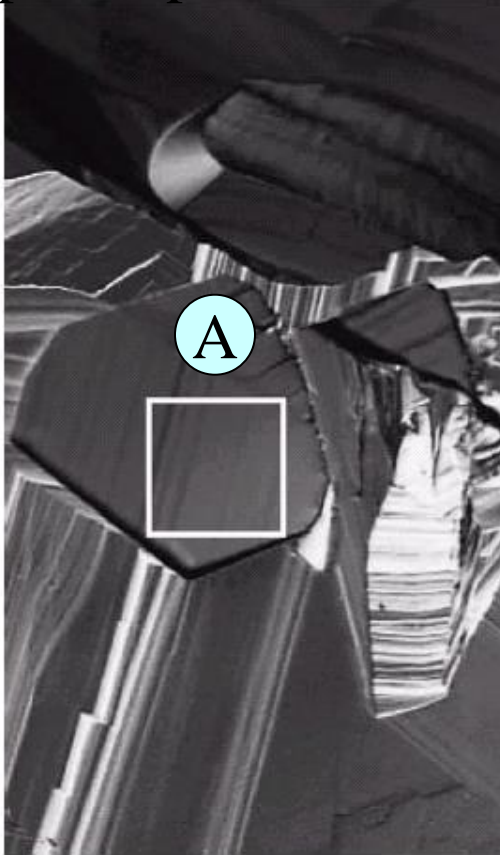


After thinning

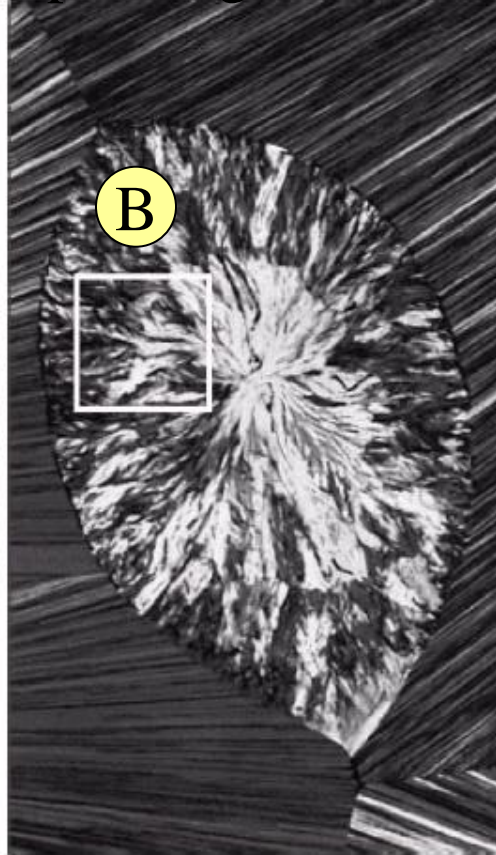
# Texture Descriptors

Purpose: to describe “texture” of the region.

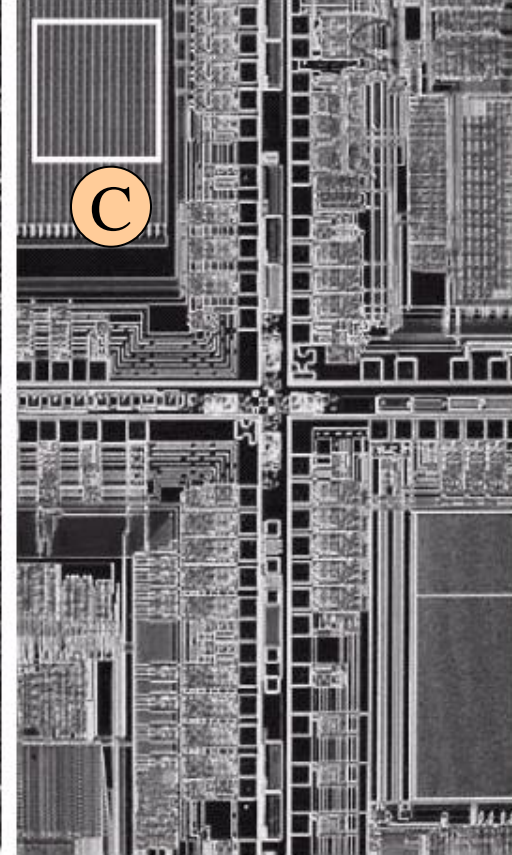
Examples: optical microscope images:



Superconductor  
(smooth texture)



Cholesterol  
(coarse texture)



Microprocessor  
(regular texture)

# Regional Descriptors Texture

- Texture is usually defined as the smoothness or roughness of a surface.
- In computer vision, it is the visual appearance of the uniformity or lack of uniformity of brightness and color.
- There are two types of texture: random and regular.
  - ▣ **Random texture** cannot be exactly described by words or equations; it must be described statistically. The surface of a pile of dirt or rocks of many sizes would be random.
  - ▣ **Regular texture** can be described by words or equations or repeating pattern primitives. Clothes are frequently made with regularly repeating patterns.
  - ▣ Random texture is analyzed by **statistical methods**.
  - ▣ Regular texture is analyzed by **structural or spectral (Fourier) methods**.

# Texture descriptions

223

- Three main approaches:
  1. **Statistical**: moments, co-occurrence matrix
  2. **Structural**, viewing a texture as an arrangement of texture primitives
  3. **Spectral**, using the Fourier transform to detect global periodicities

# Statistical Descriptors - Moments

- Let  $z$  be a random variable denoting gray levels and let  $p(z_i)$ ,  $i=0,1,\dots,L-1$ , be the corresponding histogram, where  $L$  is the number of distinct gray levels.

- The  $n$ th moment of  $z$ :

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_i - m)^n p(z_i) \quad \text{where } m = \sum_{i=0}^{L-1} z_i p(z_i)$$

**Example:** The 2<sup>nd</sup> moment = variance  $\rightarrow$  measure “smoothness”

The 3<sup>rd</sup> moment  $\rightarrow$  measure “skewness”

The 4<sup>th</sup> moment  $\rightarrow$  measure “uniformity” (flatness)

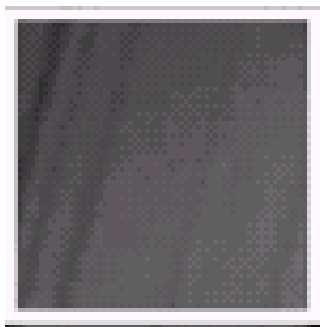
- The measure  $R$ :  $R = 1 - \frac{1}{1 + \sigma^2(z)}$

- The uniformity:  $U = \sum_{i=0}^{L-1} p^2(z_i)$

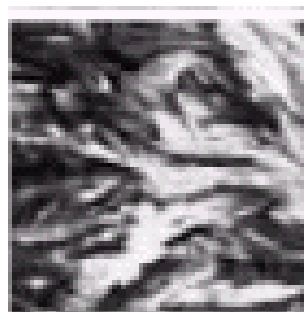
- The average entropy:  $e = -\sum_{i=0}^{L-1} p(z_i) \log_2 p(z_i)$



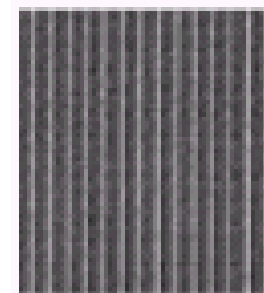
# Statistical Descriptors - Moments



Smooth



Coarse



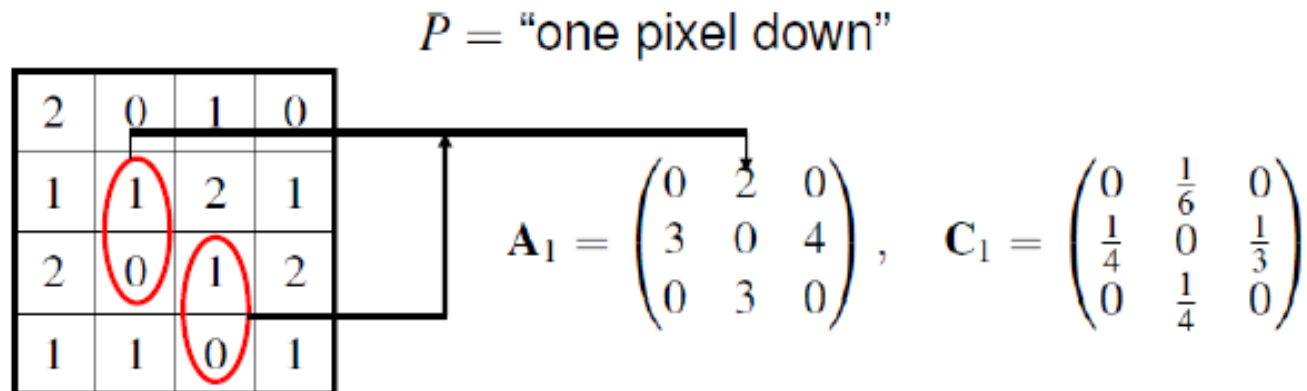
Regular

<b>Texture</b>	<b>Mean</b>	<b>Standard deviation</b>	<b><i>R</i> (normalized)</b>	<b>Third moment</b>	<b>Uniformity</b>	<b>Entropy</b>
Smooth	82.64	11.79	0.002	-0.105	0.026	5.434
Coarse	143.56	74.63	0.079	-0.151	0.005	7.783
Regular	99.72	33.73	0.017	0.750	0.013	6.674

# Statistical Descriptors - co-occurrence matrix

226

- Statistically sampling the way certain grey-levels occur in relation to other grey-levels.
- For a position operator  $p$ , we can define a matrix  $P_{ij}$  that counts the number of times a pixel with grey-level  $i$  occurs at position  $p$  from a pixel with grey-level  $j$ .
- For example, if we have three distinct grey-levels 0, 1, and 2, and the position operator  $p$  is “lower right”, the counts matrix  $P$  of the image
- If we normalize the matrix  $P$  by the total number of pixels so that each element is between 0 and 1, we get a *grey-level co-occurrence matrix*  $C$ .



# Co-Occurrence Matrices

- For a given co-occurrence matrix  $P(a, b)$ , we can compute the following six important characteristics:
  - ▣ You should compute these six characteristics for **multiple displacement vectors**, including different directions.

$$\text{Energy} = \sum_{a,b} P^2(a, b)$$

$$\text{Entropy} = \sum_{a,b} P(a, b) \log_2 P(a, b)$$

$$\text{Maximum probability} = \max_{a,b} P(a, b)$$

$$\text{Contrast} = \sum_{a,b} |a - b|^\kappa P^\lambda(a, b), \text{ usually } \kappa = 2, \lambda = 1$$

# Co-Occurrence Matrices

$$\text{Inverse difference moment} = \sum_{a,b;a \neq b} \frac{P^\lambda(a,b)}{|a-b|^\kappa}$$

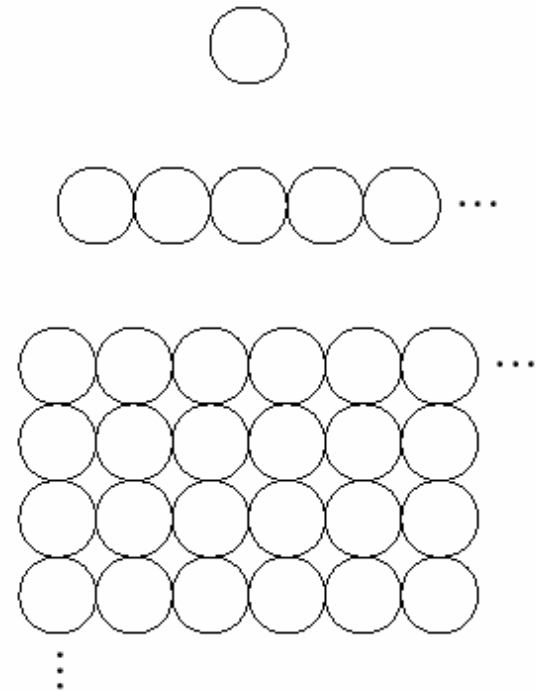
$$\text{Correlation} = \frac{\sum_{a,b} [(ab)P(a,b)] - \mu_x \mu_y}{\sigma_x \sigma_y}$$

$$\mu_x = \sum_a a \sum_b P(a,b) \quad \sigma_x = \sum_a (a - \mu_x)^2 \sum_b P(a,b)$$

$$\mu_y = \sum_b b \sum_a P(a,b) \quad \sigma_y = \sum_b (b - \mu_y)^2 \sum_a P(a,b)$$

# Structural Approaches

- Structural concepts:
  - ▣ Define a grammar for the way that the pattern of the texture produces structure.
  - ▣ Suppose that we have a rule of the form  $S \rightarrow aS$ , which indicates that the symbol  $S$  may be rewritten as  $aS$ .
  - ▣ If  $a$  represents a circle and the meaning of “circle to the right” is assigned to a string of the form  $aaaaa\dots$

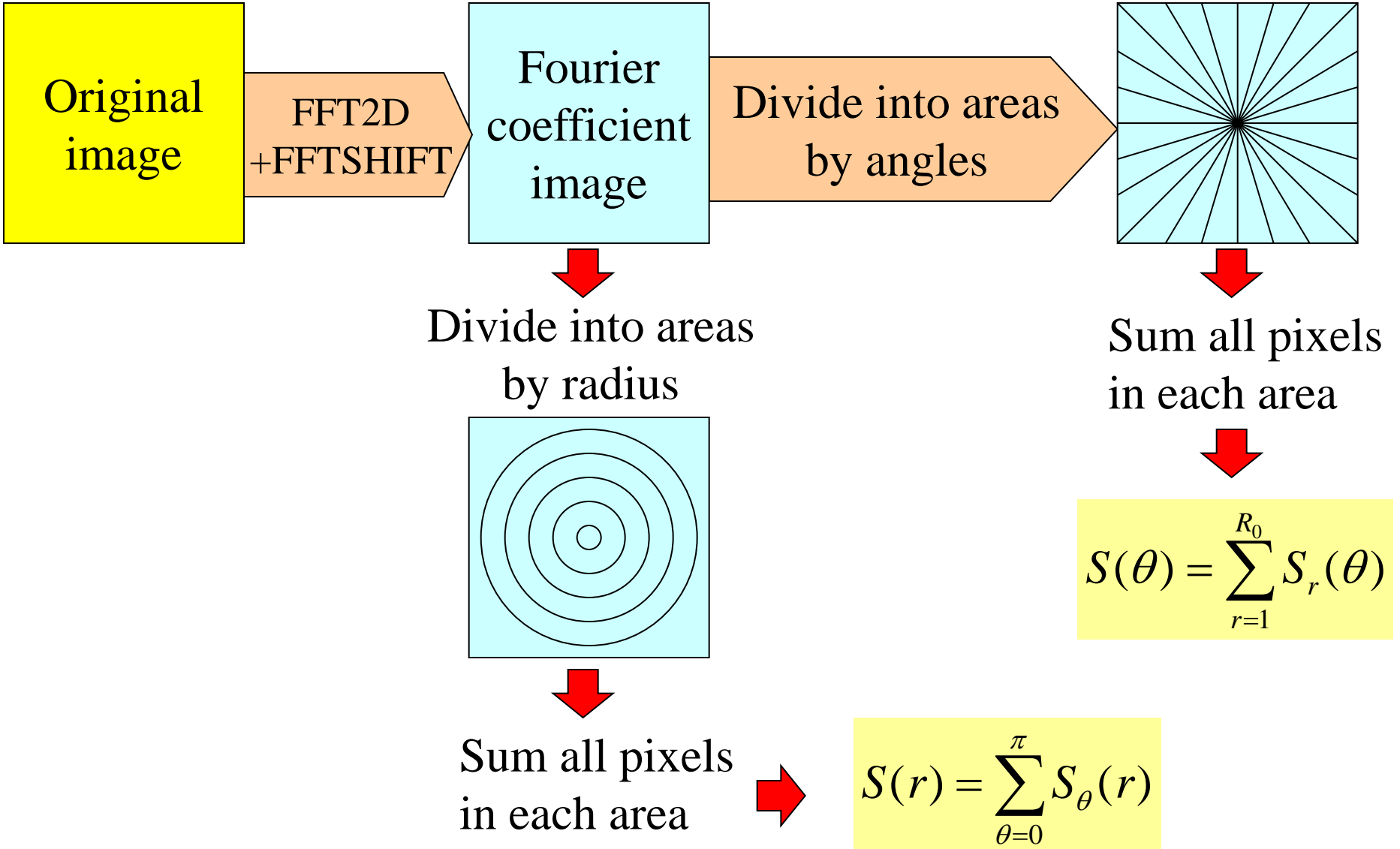


# Spectral Approaches

- For non-random primitive spatial patterns, the 2-dimensional **Fourier transform** allows the patterns to be analyzed in terms of spatial frequency components and direction.
- It may be more useful to express the spectrum in terms of **polar coordinates**, which directly give direction as well as frequency.
- Let  $S(r, \theta)$  is the spectrum function, and  $r$  and  $\theta$  are the variables in this coordinate system.
  - ▣ For each direction  $\theta$ ,  $S(r, \theta)$  may be considered a 1-D function  $S_\theta(r)$ .
  - ▣ For each frequency  $r$ ,  $S_r(\theta)$  is a 1-D function.
  - ▣ A global description:  $S(r) = \sum_{\theta=0}^{\pi} S_\theta(r)$      $S(\theta) = \sum_{r=1}^{R_0} S_r(\theta)$

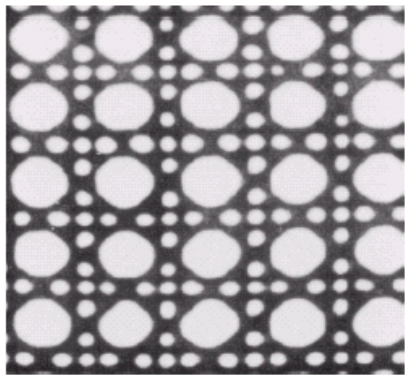
# Fourier Approach for Texture Descriptor

Concept: convert 2D spectrum into 1D graphs

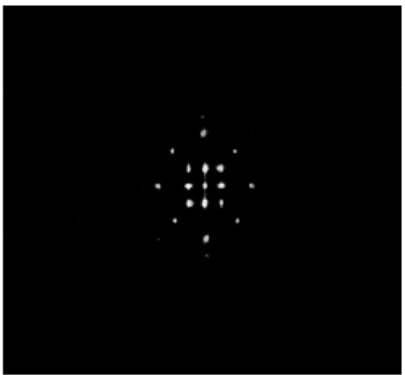


# Fourier Approach for Texture Descriptor

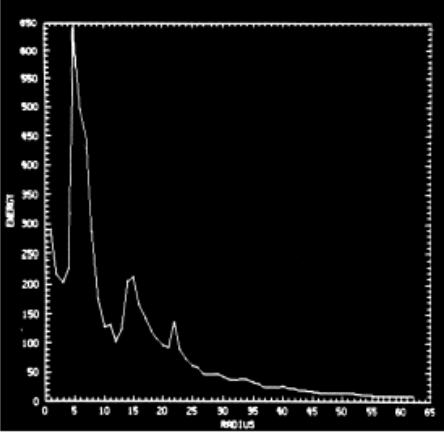
Original image



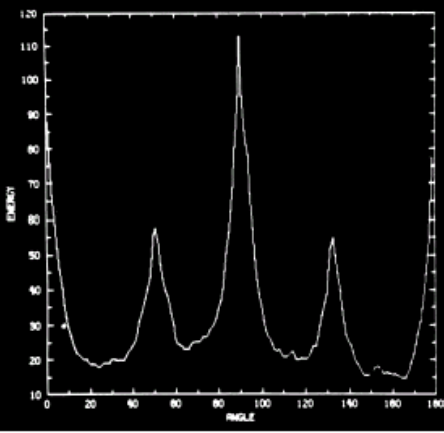
2D Spectrum (Fourier Tr.)



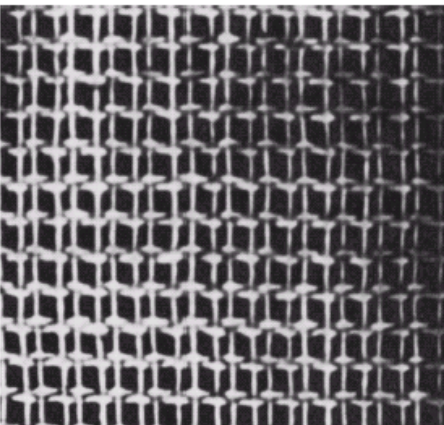
$S(r)$



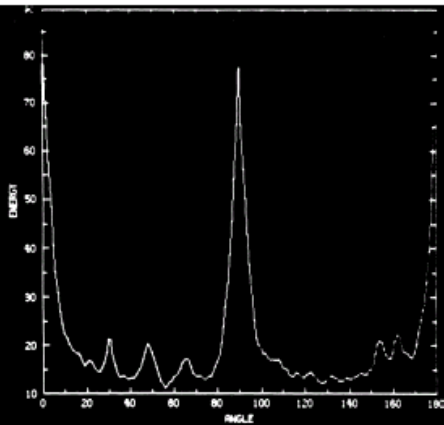
$S(\theta)$



Another image



Another  $S(\theta)$





# Moments of Two-D Functions

**The moment of order  $p + q$**

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y)$$

$$\bar{x} = \frac{m_{10}}{m_{00}}$$

$$\bar{y} = \frac{m_{01}}{m_{00}}$$

**The central moments of order  $p + q$**

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y)$$

$$\mu_{00} = m_{00}$$

$$\mu_{01} = \mu_{10} = 0$$

$$\mu_{11} = m_{11} - \bar{x}m_{01} = m_{11} - \bar{y}m_{10}$$

$$\mu_{20} = m_{20} - \bar{x}m_{10} \quad \mu_{02} = m_{02} - \bar{y}m_{01}$$

$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2m_{01}$$

$$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2m_{10}$$

$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2m_{10}$$

$$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2m_{01}$$

# Invariant Moments of Two-D Functions

The normalized central moments of order  $p + q$

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad \text{where} \quad \gamma = \frac{p+q}{2} + 1$$

**Invariant moments: independent of rotation, translation, scaling, and reflection**

$$\phi_1 = \eta_{20} + \eta_{02} \quad \phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

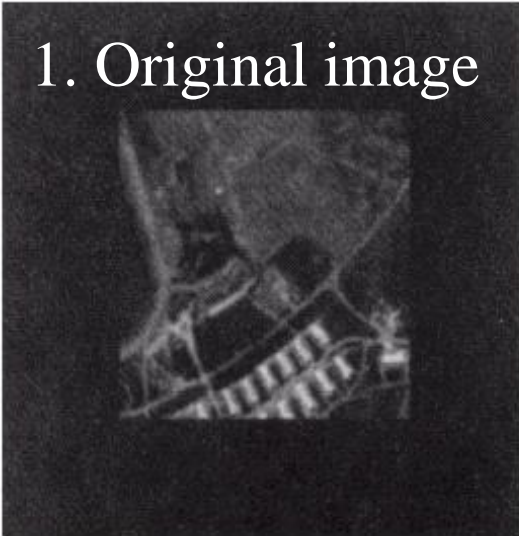
$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \quad \phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

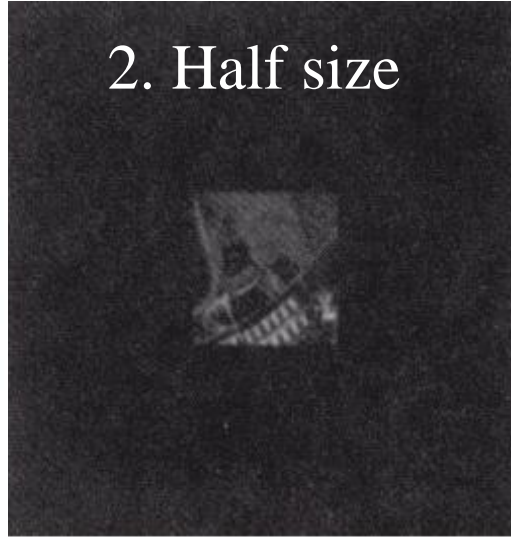
$$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + \\ 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

# Example: Invariant Moments of Two-D Functions

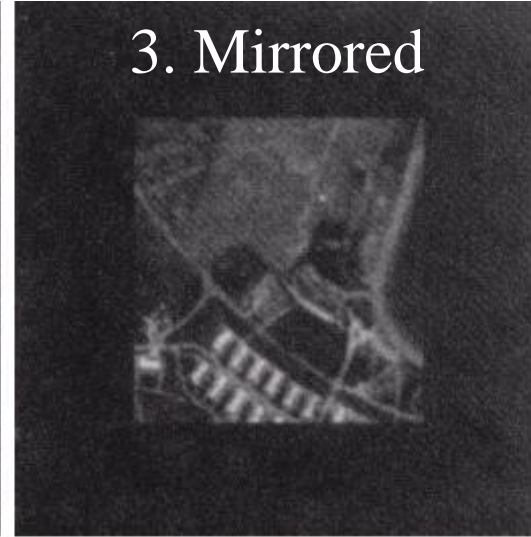
1. Original image



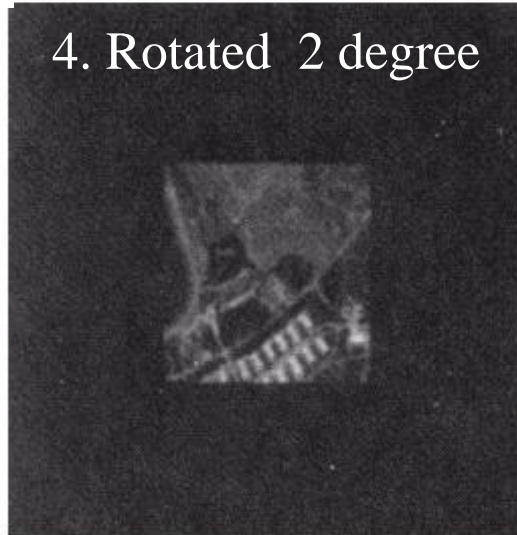
2. Half size



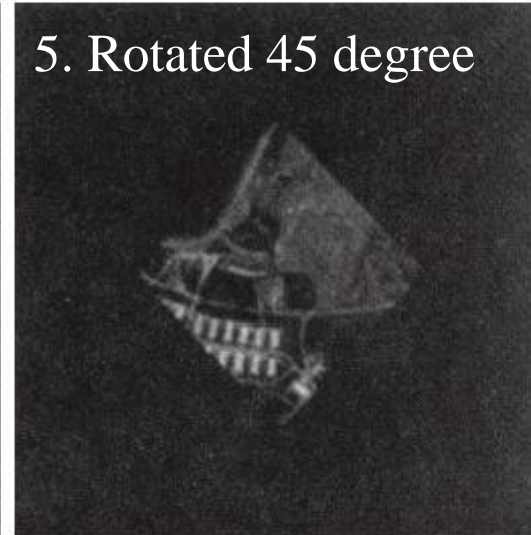
3. Mirrored



4. Rotated 2 degree



5. Rotated 45 degree



# Example: Invariant Moments of Two-D Functions

Invariant moments of images in the previous slide

Invariant (Log)	Original	Half Size	Mirrored	Rotated 2°	Rotated 45°
$\phi_1$	6.249	6.226	6.919	6.253	6.318
$\phi_2$	17.180	16.954	19.955	17.270	16.803
$\phi_3$	22.655	23.531	26.689	22.836	19.724
$\phi_4$	22.919	24.236	26.901	23.130	20.437
$\phi_5$	45.749	48.349	53.724	46.136	40.525
$\phi_6$	31.830	32.916	37.134	32.068	29.315
$\phi_7$	45.589	48.343	53.590	46.017	40.470

**Invariant moments are independent of rotation, translation, scaling, and reflection**

# Shape features categories Comparison

237

- Contour-based approaches are more popular than region-based approaches in literature.
  - ▣ Human beings are thought to discriminate shapes mainly by their contour features.
- However, there are several limitations with contour-based methods.
  - ▣ contour shape descriptors are generally sensitive to noise and variations because they only use a small part of shape information
  - ▣ In many cases, the shape contour is not available.
  - ▣ In some applications, shape content is more important than the contour features.

# Shape features categories Comparison

238

- Region-base methods are more robust as they use all the shape information available; they can be applied to general applications; and
  - ▣ they generally provide more accurate retrieval. In addition,
  - ▣ region-based methods can cope well with shape defection, which is a common problem for contour-based shape representation techniques.
- Although region-based methods make use of all the shape information, it is not necessarily more complex than contour-based methods

# Shape features categories Performance

239

Shape representation		Recon- struc- ture	Invariance				Resistance			Computational complexity	
			Translation	Rotation	Scale	Affine transform	Noise	Occultation	Non-rigid deformation		
Shape signatures	Complex coordinates	Yes	Bad	Bad	Bad	Bad	Average	Good	Bad	Low	
	Central distance	No	Good	Good	Good	Bad	Average	Good	Bad	Low	
	Tangent angle	No	Good	Good	Good	Bad	Bad	Good	Average	Low	
	Curvature function	No	Good	Good	Good	Bad	Bad	Good	Average	Low	
	Area function	No	Good	Good	Good	Good	Good	Good	Bad	Low	
	Triangle-area representation	No	Good	Good	Good	Good	Good	Average	Average	Low	
	Chord length function	No	Good	Good	Good	Bad	Bad	Bad	Bad	Low	
Polygonal approximation	Merging	Distance threshold	No	Good	Good	Good	Bad	Good	Bad	Bad	Average
		Tunneling	No	Good	Good	Good	Bad	Good	Bad	Bad	Average
		Polygon evolution	No	Good	Good	Good	Bad	Good	Bad	Bad	Average
	Splitting	No	Good	Good	Good	Bad	Good	Bad	Bad	Average	

# Shape features categories Performance

240

Shape representation		Recon- struc- ture	Invariance				Resistance			Computational complexity
			Translation	Rotation	Scale	Affine transform	Noise	Occultation	Non-rigid deformation	
Adaptive grid resolution		Yes	Good	Good	Good	Bad	Good	Good	Bad	Low
Bounding box		Yes	Good	Good	Good	Average	Good	Good	Average	Average
Convex hull		No	Good	Good	Good	Good	Average	Bad	Bad	High
Chain code	Basic chain code	Yes	Good	Bad	Bad	Bad	Bad	Good	Bad	Low
	Vertex chain code	Yes	Good	Bad	Bad	Bad	Bad	Good	Bad	Low
	Statistic chain code	No	Good	Bad	Bad	Bad	Bad	Bad	Bad	Low
Smooth curve decomposition		No	Good	Good	Good	Bad	Good	Good	Average	Average
ALI-based representation		No	Good	Good	Good	Average	Good	Average	Bad	Average
Beam angle statistics		No	Good	Good	Good	Bad	Good	Bad	Bad	Low
Shape matrix	Square model	Yes	Good	Good	Good	Bad	Bad	Good	Bad	Average
	Polar model	Yes	Good	Good	Good	Bad	Bad	Good	Bad	Low
Shape context		No	Good	Good	Good	Bad	Bad	Average	Average	Average
Chord distribution		No	Good	Good	Good	Bad	Good	Bad	Bad	Low
Shock graphs		Yes	Good	Good	Good	Good	Good	Good	Good	High

Space interrelation Feature



# Shape features categories Performance

Shape representation		Recon- struc- ture	Invariance				Resistance			Computational complexity	
			Translation	Rotation	Scale	Affine transform	Noise	Occultation	Non-rigid deformation		
Moments	Boundary moments		No	Good	Good	Good	Bad	Average	Bad	Bad	Low
	Region moments	Invariant moments	No	Good	Good	Good	Bad	Bad	Bad	Bad	Average
		Algebraic Moment	No	Good	Good	Good	Good	Average	Bad	Bad	Average
		Zernike Moments	No	Good	Good	Good	Bad	Good	Average	Average	High
		Radial Chebyshev Moments	No	Good	Good	Good	Bad	Good	Average	Average	High
Scale-space methods	Curvature scale space		No	Good	Good	Good	Average	Good	Good	Average	Average
	Intersection points map		No	Good	Good	Good	Average	Good	Good	Bad	Average
Shape transform domains	Fourier descriptors	1-D Fourier descriptor	No	Good	Good	Good	Bad	Bad	Bad	Bad	Average
		Region-based Fourier descriptor	No	Good	Good	Good	Good	Good	Average	Average	High
	Wavelet transform		No	Good	Good	Good	Good	Average	Average	Bad	Average
	Angular radial transformation		No	Good	Good	Good	Bad	Good	Bad	Bad	High
	Signature harmonic embedding		No	Good	Good	Good	Average	Good	Average	Bad	High
	$\mathcal{R}$ -Transform		No	Good	Good	Good	Bad	Good	Average	Average	High
	Shapelets descriptor		No	Good	Good	Good	Bad	Good	Bad	Bad	High

# Shape Features

- ▶ **Shape Descriptor**
  - ▶ Contour based
  - ▶ Region Based
- ▶ **Shape Matching**

# Shape-Based Recognition

243

- High-level feature extraction concerns in finding shapes in computer images.
- Humans can recognize many objects based on shape alone
- Fundamental cue for many object categories
- Invariant to photometric variation.



Similar to a human in terms of shape, but very different in terms of pixel values.

# kinds of problems addressed by shape matching

244

- **Computation problems:** compute the similarity between two patterns.
- **Decision problems:** given a threshold, decide if the similarity/dissimilarity is larger/smaller than the threshold.
- **Decision problem:** given a threshold, decide if there is a transformation after which the dissimilarity between the transformed shape and the other shape is less than the threshold.
- **Optimization problem:** find the transformation that minimizes the dissimilarity between the transformed shape and the other shape.
- **Approximate optimization problem:** Often, the complexities of solving the above problems are extremely high. For such a case, an approximation algorithm finds a transformation that permits a dissimilarity between the two shapes that is within a constant factor from the minimum dissimilarity.

# Applications

245

- **Shape retrieval:** search for shapes in a large database that are similar to a query shape
- **Shape recognition and classification:** determine if a given shape is sufficiently similar to another shape, or find the most similar shape from a set of shapes.
- **Shape alignment and registration:** transform one shape to find the best matching to a second shape.
- **Shape approximation and simplification:** create a shape that is less complex (with fewer vertices, triangles, etc.), but still similar to the original shape.

# Types of matching

- Direct use of pixel
  - ▣ Correlation
- Use low-level features
  - ▣ Edges or corners
- High-level matchers
  - ▣ Use identified parts of objects
  - ▣ Relations between features.

# Shape Matching Approaches

247

- Distance based – Binary images
  - ▣ Hausdorff Distance
  - ▣ Shape Context
- Correlation Based – Gray level images
- Hierarchical Approach
  - ▣ Hierarchical Matching
- Machine Learning Approach
  - ▣ Boundary Fragment Model

# Hausdorff Distance

248

- Use Hausdorff distance to compare images to a model
  - Fast and simple approach
  - Tolerant of small position errors
  - Model is only allowed to translate with respect to the image
  - Can be extended to allow rotation and scale



# Hausdorff Distance

249

- A means of determining the resemblance of one point set to another
- Examines the fraction of points in one set that lie near points in the other set

$$H(A, B) = \max \{h(A, B), h(B, A)\}$$

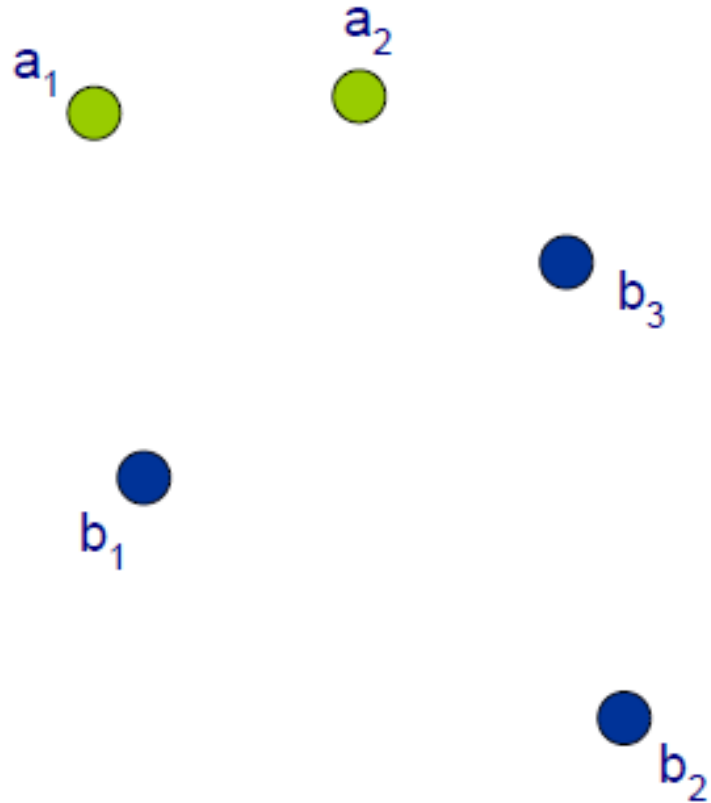
$$h(A, B) = \max_{a \in A} \left\{ \min_{b \in B} \{d(a, b)\} \right\}$$

- Intuitively, the function  $h(P, Q)$  finds the point  $p$  from  $P$  that is farthest from any point in  $Q$  and measures the distance from  $p$  to its nearest neighbor in  $Q$ .

# Hausdorff Distance - Example

250

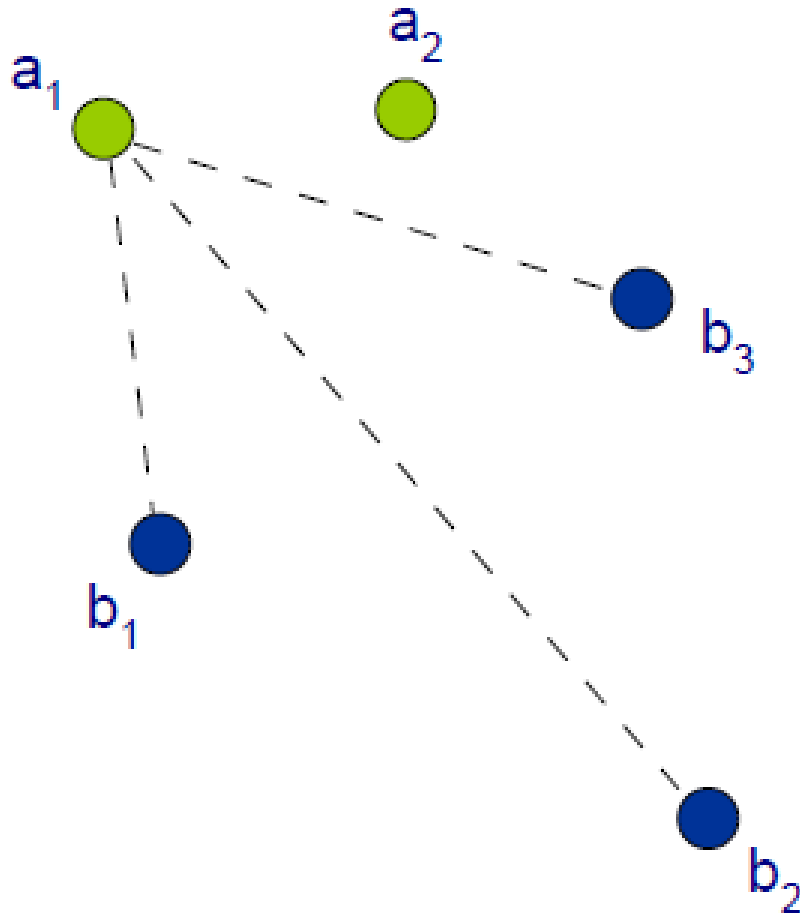
Given two sets of points  
A and B, find  $h(A,B)$



# Hausdorff Distance - Example

251

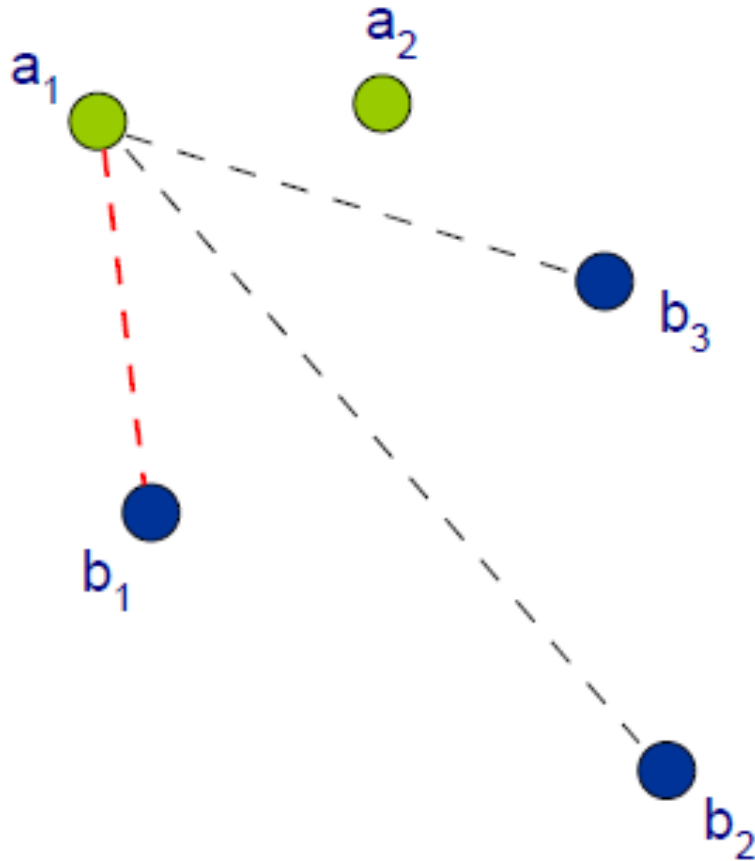
Compute the distance between  $a_1$  and each  $b_j$



# Hausdorff Distance - Example

252

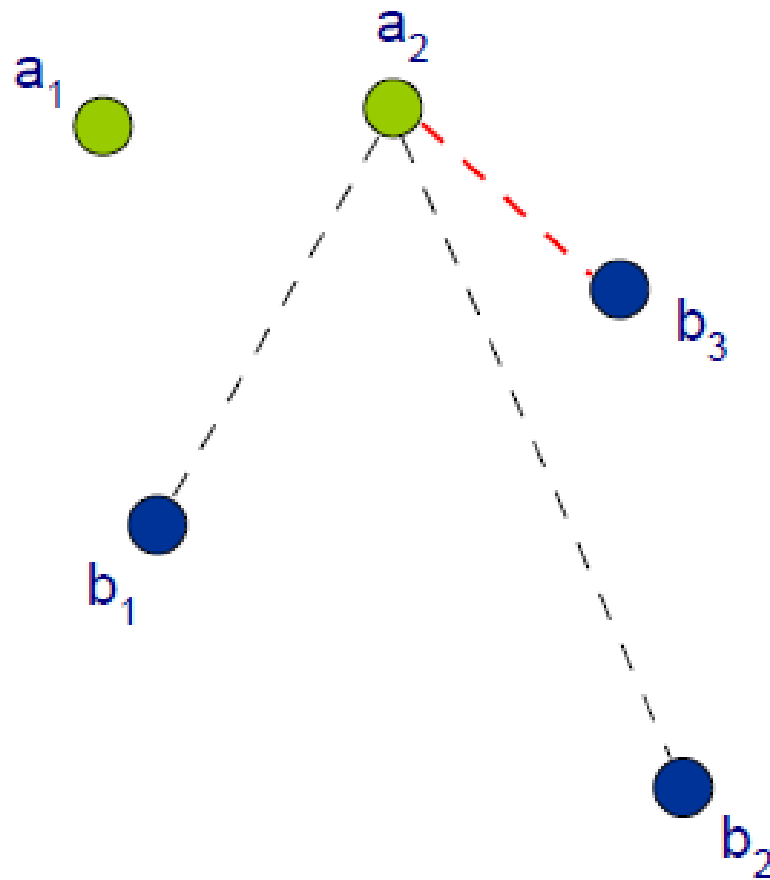
Keep the shortest



# Hausdorff Distance - Example

253

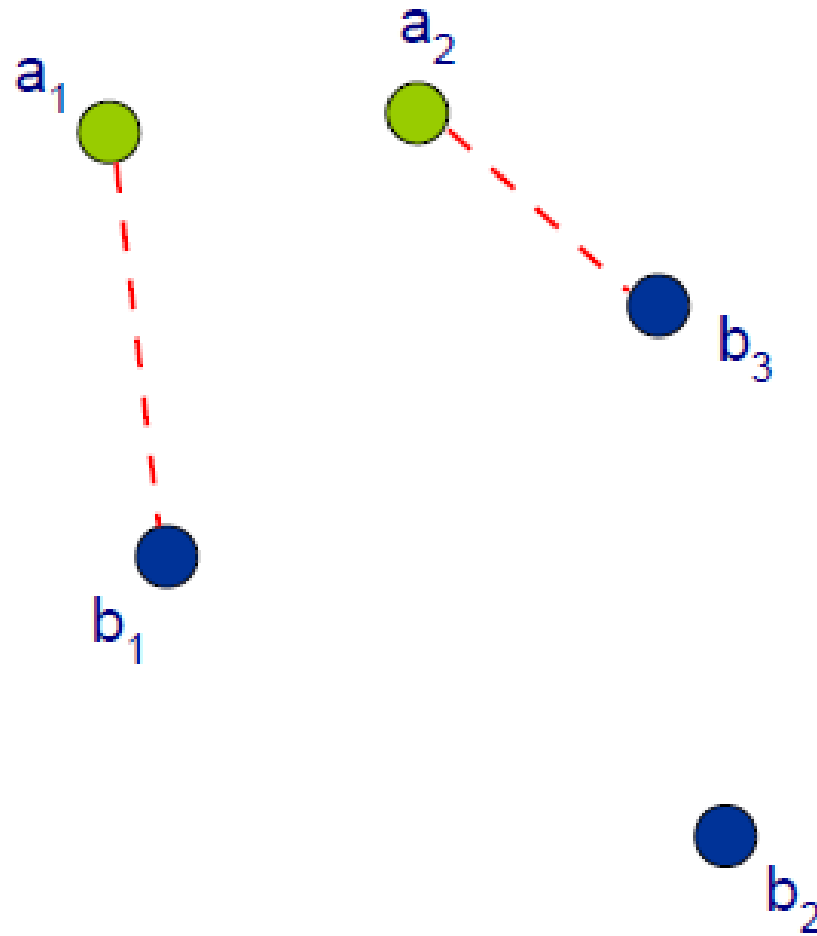
Do the same for  $a_2$



# Hausdorff Distance - Example

254

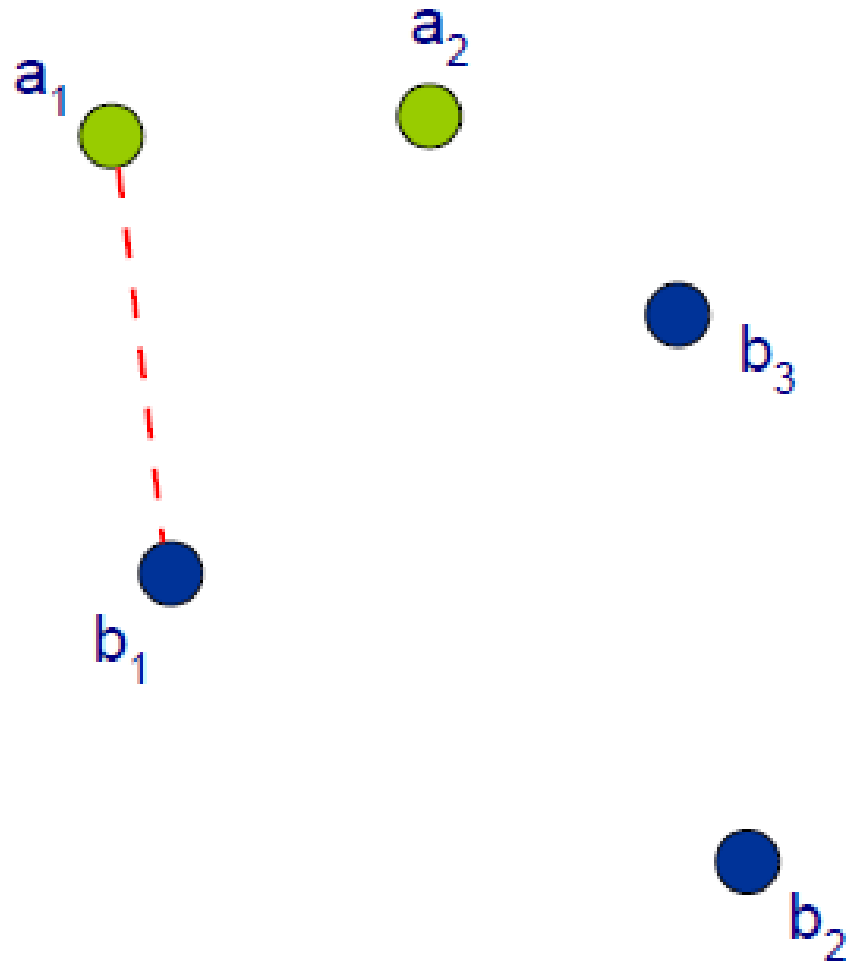
Find the largest of these two distances



# Hausdorff Distance - Example

255

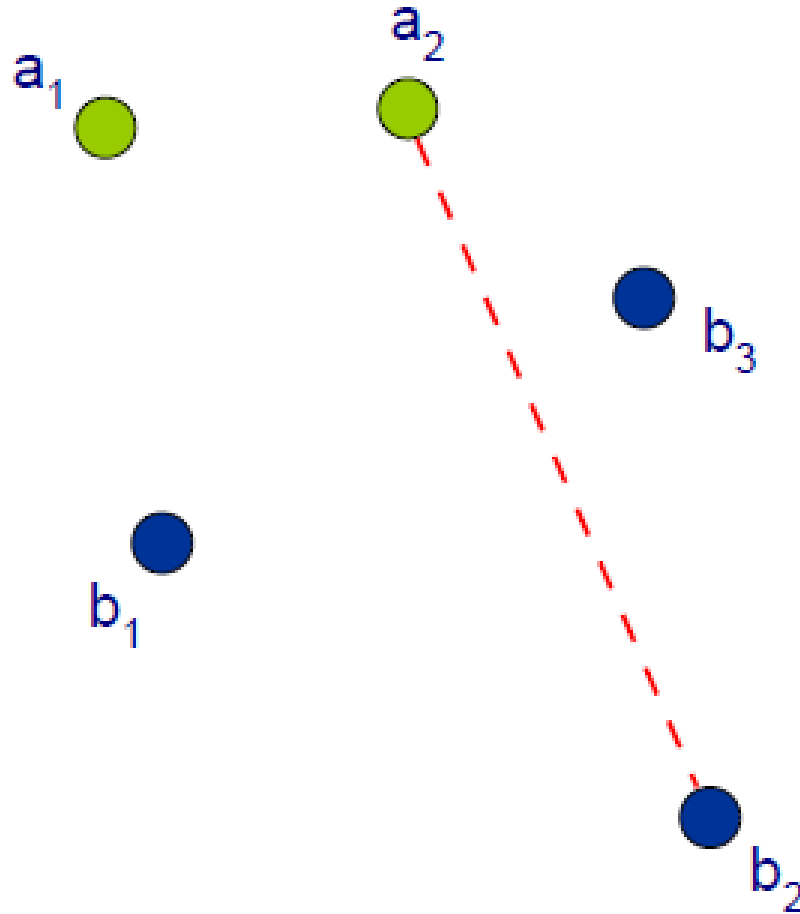
This is  $h(A,B)$



# Hausdorff Distance - Example

256

This is  $h(B,A)$

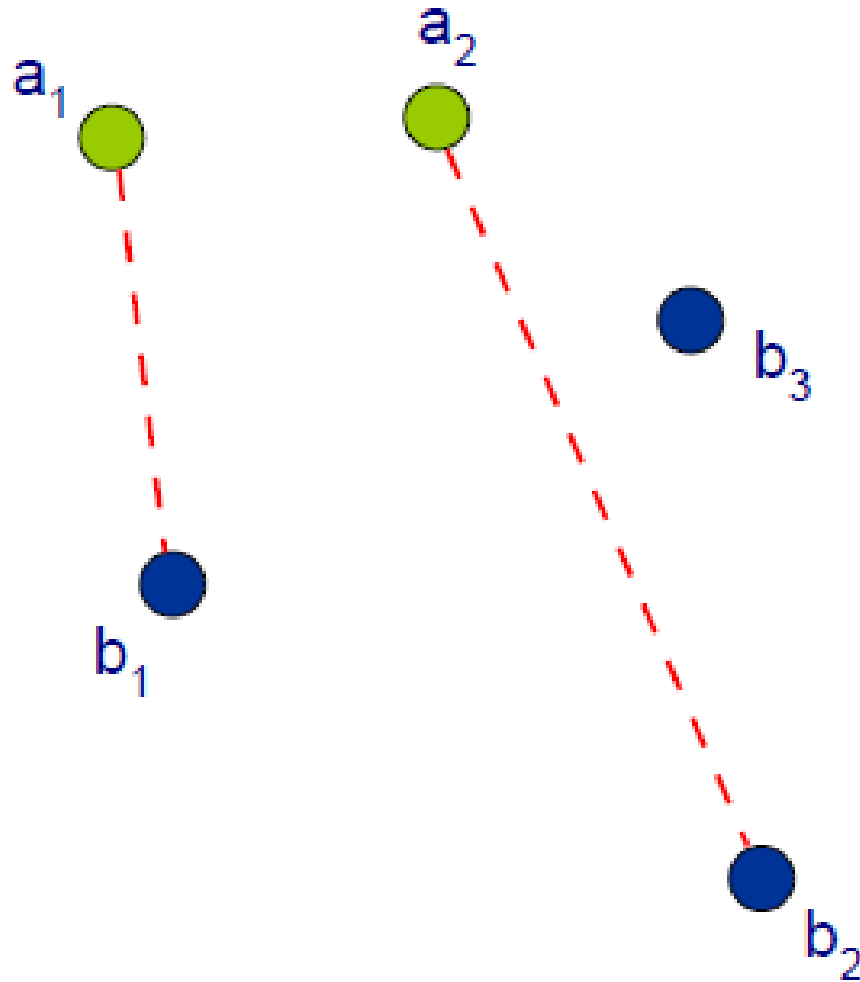




# Hausdorff Distance - Example

257

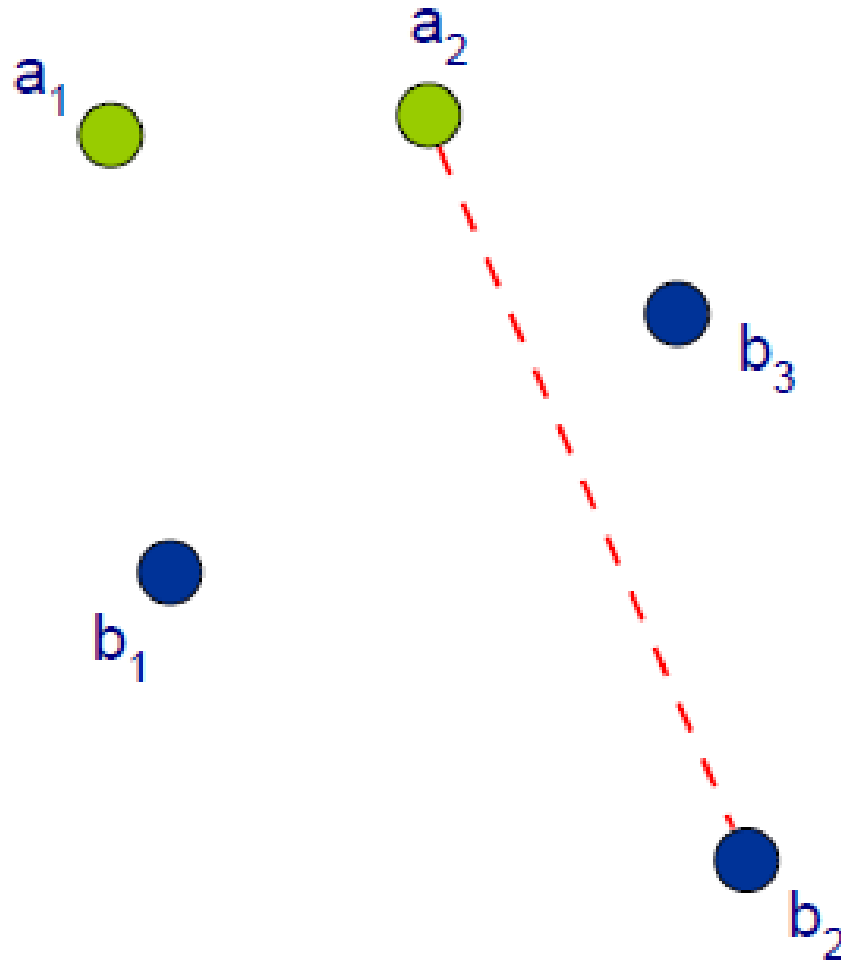
$$H(A,B) = \max(h(A,B), h(B,A))$$



# Hausdorff Distance - Example

258

This is  $H(A,B)$



# Hausdorff Distance

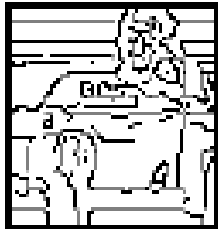
259

- Hausdorff distance is very sensitive to even one outlier in  $A$  or  $B$
- Use  $k^{\text{th}}$  ranked distance instead of the maximal distance
- Match if  $h_k(A, B) < \delta$ 
  - $k$  is how many points of the model need to be near points of the image
  - $\delta$  is how near these points need to be

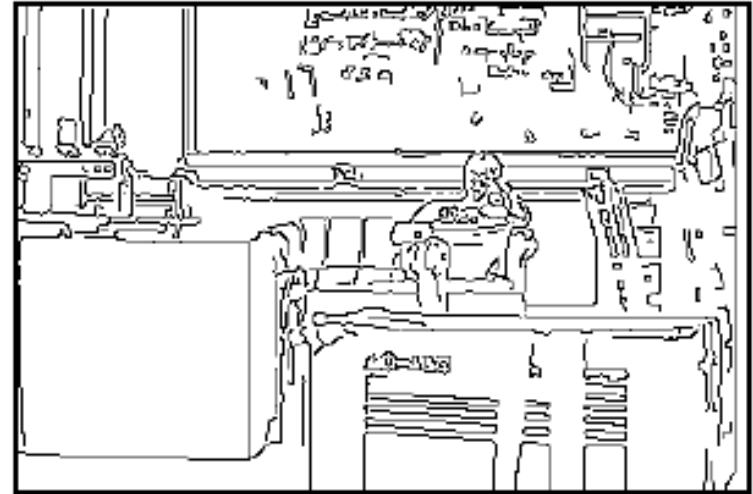
$$h_k(A, B) = \underset{a \in A}{k^{\text{th}}} \left\{ \min_{b \in B} \{d(a, b)\} \right\}$$

# Hausdorff Distance- Example Matching

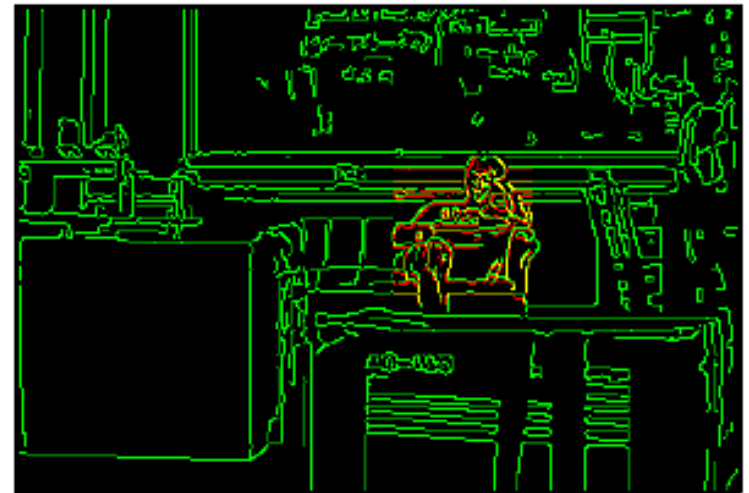
260



Model



Edges



Match

# Hausdorff Distance- Example Matching

261



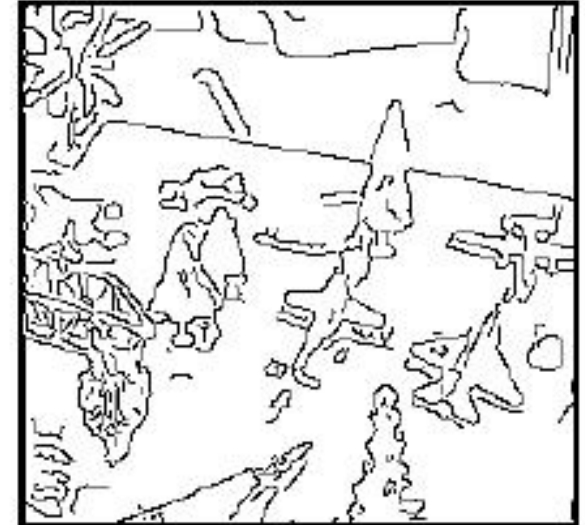
Model



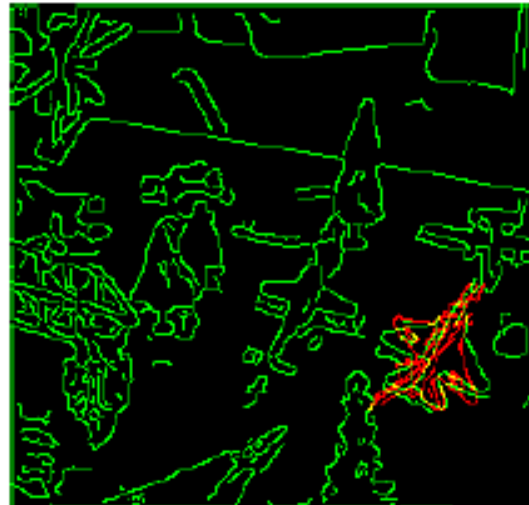
Image



Model



Edges



Match

# Comparing Binary Feature Maps

262

- Binary “image” specifying feature locations
  - ▣ In  $x,y$  or  $x,y,scale$
- Even small variations will cause maps not to align precisely
- Distance transforms a natural way to “blur” feature locations geometrically
- Natural generalization also applies not just to binary data but to any cost or height map

# Distance Transform

- The distance transform is an operator normally only applied to binary images.
- **Distance Transform** is a function that for each image pixel  $p$  assigns a non-negative number corresponding to distance from  $p$  to the nearest feature in the image  $I$
- The result of the transform is a graylevel image that looks similar to the input image, except that the graylevel intensities of points inside foreground regions are changed to show the distance to the closest boundary from each point.

*Image features (2D)*

*Distance Transform*

1	0	1	2	3	4	3	2
1	0	1	2	3	3	2	1
1	0	1	2	3	2	1	0
1	0	0	1	2	1	0	1
2	1	1	2	1	0	1	2
3	2	2	2	1	0	1	2
4	3	3	2	1	0	1	2
5	4	4	3	2	1	0	1

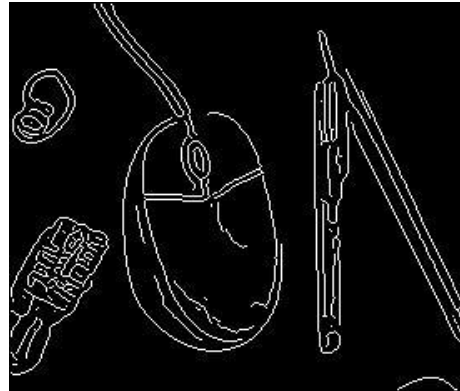
# Uses of Distance Transforms

264

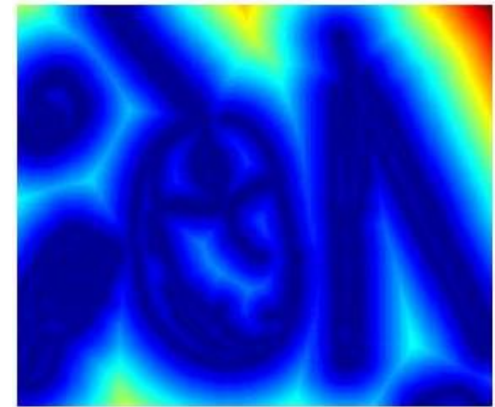
- Image matching and object recognition
  - – Hausdorff and Chamfer matching
  - – Skeletonization



original



edges



distance transform

↑  
Value at  $(x,y)$  tells how far that position is from the nearest edge point



# Shape Matching by Shape Contexts

265

- The shape context is intended to be a way of describing shapes that allows for measuring shape similarity and the recovering of point correspondences
- Shape contexts for shape matching steps:
  - ▣ Randomly select a set of points that lie on the edges of a known shape and another set of points on an unknown shape.
  - ▣ Compute the shape context of each point found in step 1.
  - ▣ Match each point from the known shape to a point on an unknown shape.
  - ▣ Calculate the "shape distance" between each pair of points on the two shapes.

# Shape Matching by Shape Contexts

266

- **Step 1: Finding a list of points on shape edges**
  - The approach assumes that the shape of an object is essentially captured by a finite subset of the points on the internal or external contours on the object.
  - These can be simply obtained using the Canny edge detector and picking a random set of points from the edges.
  - Note that these points need not and in general do not correspond to key-points such as maxima of curvature or inflection points.
  - It is preferable to sample the shape with roughly uniform spacing, though it is not critical

# Shape Matching by Shape Contexts

267

## □ Step 2: Computing the shape context

### ▣ The basic idea

- Pick  $n$  points on the contours of a shape.
- For each point  $p_i$  on the shape, consider the  $n - 1$  vectors obtained by connecting  $p_i$  to all other points. The set of all these vectors is a rich description of the shape localized at that point but is far too detailed. The key idea is that the distribution over relative positions is a robust, compact, and highly discriminative descriptor.
- The point  $p_i$ , the coarse histogram of the relative coordinates of the remaining  $n - 1$  points, is defined to be the shape context of  $p_i$ .
- The bins are normally taken to be uniform in log-polar space.

# Shape Matching by Shape Contexts

268

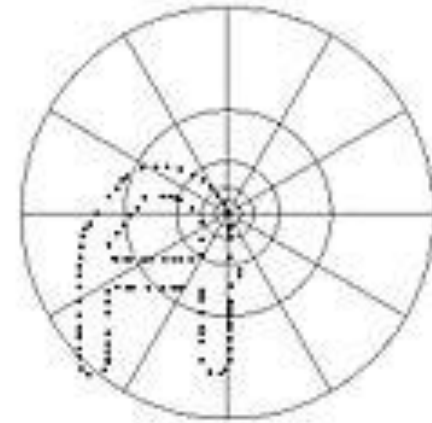
## Step 2: Computing the shape context



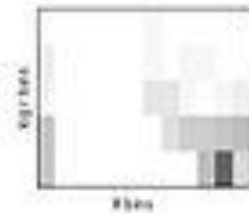
(a)



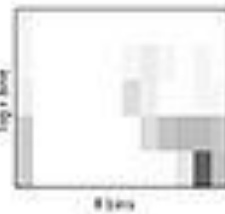
(b)



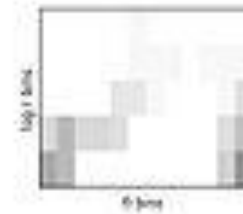
(c)



(d)



(e)



(f)

# Shape Matching by Shape Contexts

269

## □ Step 3: Computing the cost matrix

- ▣ Consider two points  $p$  and  $q$  that have normalized  $K$ -bin histograms (i.e. shape contexts)  $g(k)$  and  $h(k)$ . As shape contexts are distributions represented as histograms, it is natural to use the  $\chi^2$  test statistic as the "shape context cost" of matching the two points:

$$C_S = \frac{1}{2} \sum_{k=1}^K \frac{[g(k) - h(k)]^2}{g(k) + h(k)}$$

- The values of this range from 0 to 1

# Shape Matching by Shape Contexts

270

## □ Step 3: Computing the cost matrix

- ▣ In addition to the shape context cost, an extra cost based on the appearance can be added. For instance, it could be a measure of tangent angle dissimilarity (particularly useful in digit recognition):

$$C_A = \frac{1}{2} \left\| \begin{pmatrix} \cos(\theta_1) \\ \sin(\theta_1) \end{pmatrix} - \begin{pmatrix} \cos(\theta_2) \\ \sin(\theta_2) \end{pmatrix} \right\|$$

- Its values also range from 0 to 1

# Shape Matching by Shape Contexts

271

## □ Step 3: Computing the cost matrix

- Now the total cost of matching the two points could be a weighted-sum of the two costs:

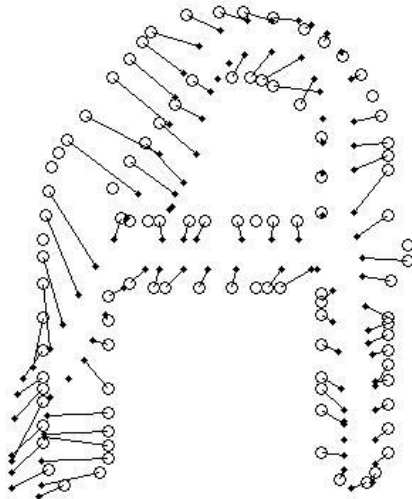
$$C = (1 - \beta)C_S + \beta C_A$$

- Now for each point  $p_i$  on the first shape and a point  $q_j$  on the second shape, calculate the cost as described and call it  $C_{i,j}$ . This is the cost matrix.

# Shape Matching by Shape Contexts

272

- **Step 4: Finding the matching that minimizes total cost**
  - ▣ Now, a one-to-one matching  $p_i$  that matches each point  $p_i$  on shape 1 and  $q_j$  on shape 2 that minimizes the total cost of matching is needed
  - ▣ This can be done in  $O(N^3)$  time using the [Hungarian method](#), although there are more efficient algorithms

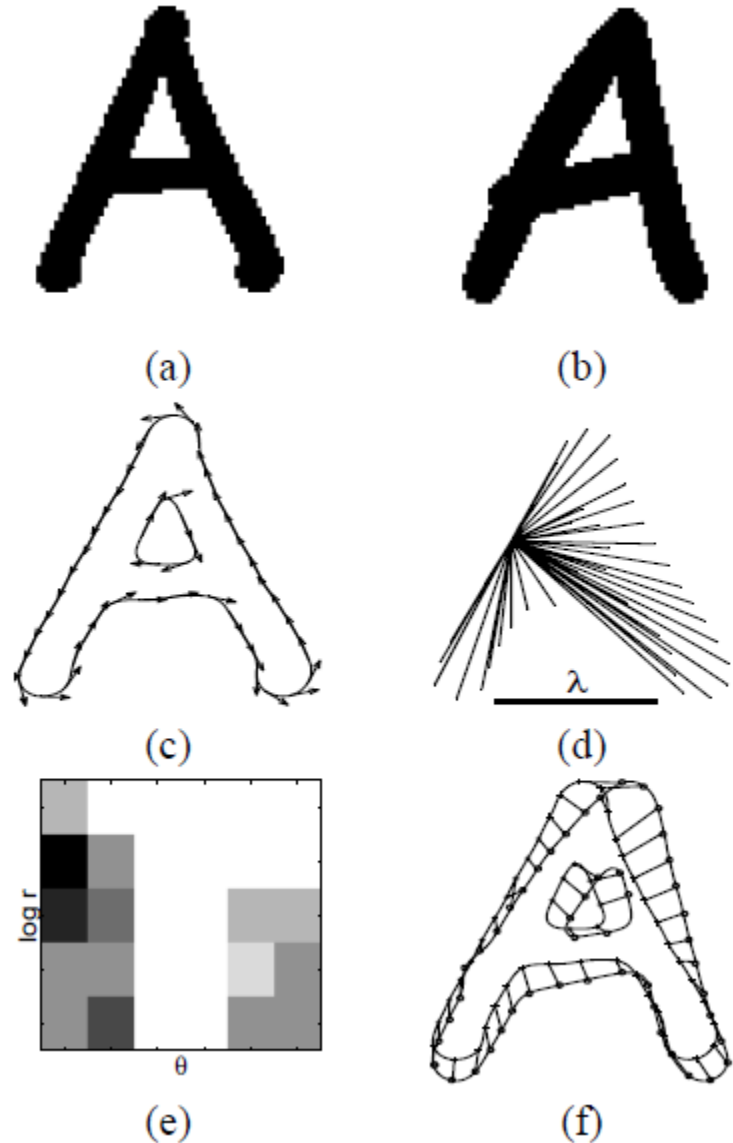




# Shape Matching by Shape Contexts

273

- (a,b) Original image pair.
- (c) Edges and tangents of first letter with 50 sample points.
- (d) Vectors from a sample point (at left, middle) to all other points.
- (d) histogram of vectors with 5 and 6 bins, respectively. (Dark=large value.)
- (f) Correspondences found using Hungarian method, with weights given by sum of two terms: histogram dissimilarity and tangent angle dissimilarity.

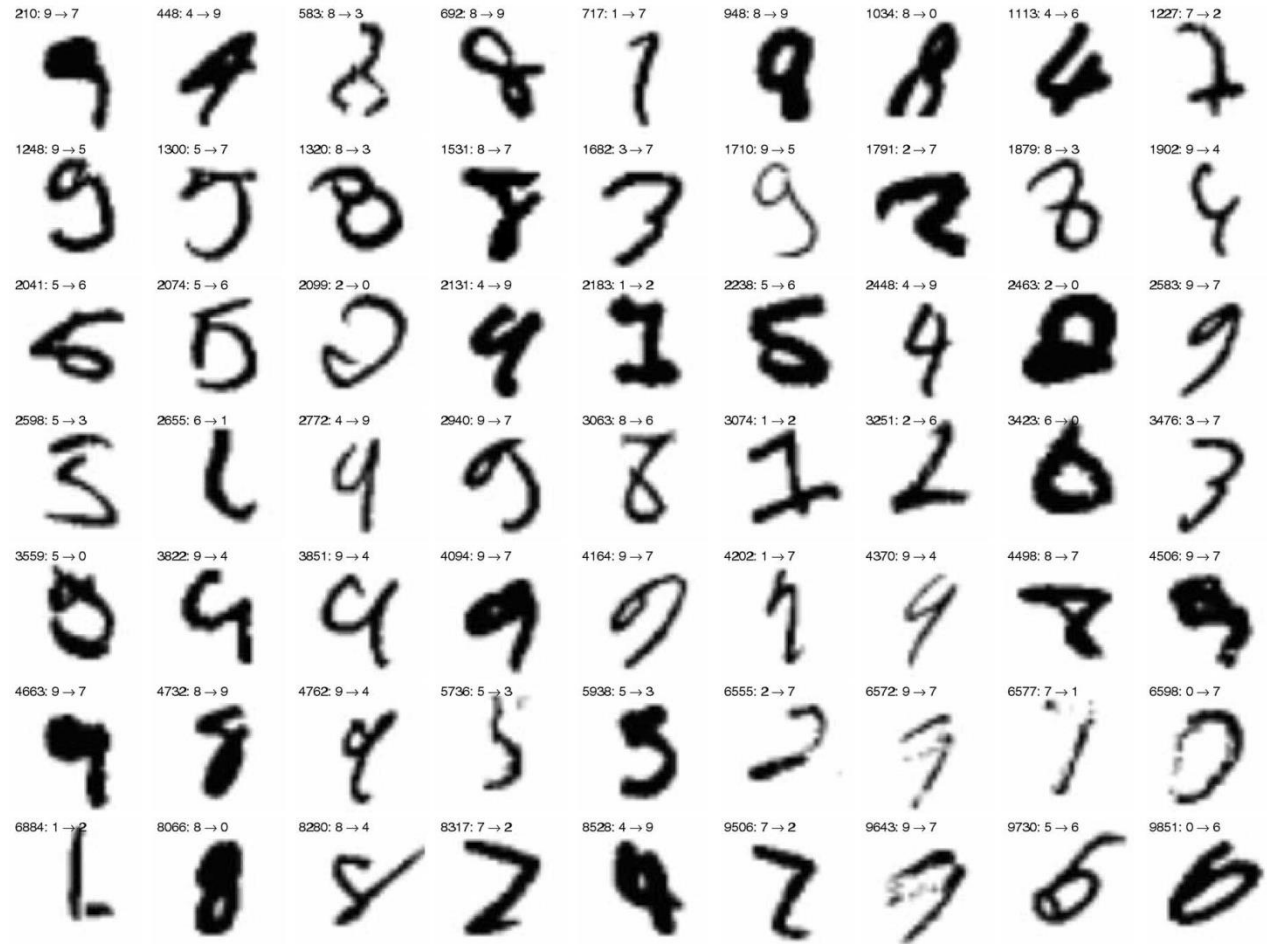


# Application: Digit Recognition

Training set  
size:  
20,000

Test set  
size:  
10,000

Error:  
0.63%



# Application: Breaking CAPTCHA

polish

join

weight

horse

again

jewel

spade

sound

mine

space

flag

rice

mark

canvas

porter

sock

92% success rate  
on  
EZ-Gimpy

# Application: Trademark Retrieval

- Can be used to find different shapes with similar elements.
- Useful to determine cases of trademark infringement.



query



1: 0.086



2: 0.108



3: 0.109



query



1: 0.066



2: 0.073



3: 0.077



query



1: 0.046



2: 0.107



3: 0.114



query



1: 0.046



2: 0.107



3: 0.114

# Shape Matching by Shape Contexts – Invariance

277

- Translational invariance come naturally to shape context.
- Scale invariance is obtained by normalizing all radial distances by the mean distance between all the point pairs in the shape although the median distance can also be used.
- Shape contexts are empirically demonstrated to be robust to deformations, noise, and outliers using synthetic point set matching experiments.
- One can provide complete rotation invariance in shape contexts. One way is to measure angles at each point relative to the direction of the tangent at that point (since the points are chosen on edges).

# Other Method for Shape Matching

278

- ❑ **Matching Images/Video Using Local Self-Similarities**
- ❑ **The Pyramid Match Kernel**
- ❑ **Hierarchical Matching of Deformable Shapes**
- ❑ **A Boundary-Fragment-Model for Object Detection**

# Features Post Processing

279

- Number and types of features have a direct effect on the performance of classification and clustering techniques besides the model efficiency in time and memory
- Two Approaches for analyzing features the
  - ▣ Dimensionality Reduction
  - ▣ Feature Selection