CS3002: Artificial Intelligence

Images and Vision

Yongmin Li

Acknowledgement

- **Sarbjeet Singh**
- **Antonio Torralba**
- **Srinivasa Narasimhan**

Overview – Early Vision

- **Image Fundamentals**
- **Edge Detection**
- **Corner Detection**
- **Line Detection by Hough Transform**

1-bit Images

- Each pixel is stored as a single bit (0 or 1), so also referred to as binary image.
- Such an image is also called a 1-bit monochrome image since it contains no colour.

8-bit Gray-Level Images

- Each pixel has a gray-value between 0 and 255. Each pixel is represented by a single byte; e.g., a dark pixel might have a value of 10, and a bright one might be 230.

24-bit Colour Images

- In a colour 24-bit image, each pixel is represented by three bytes, usually representing RGB.

24-bit Colour Images

(a)

(b)

(c)

(d)

(a): Example of 24-bit colour image “forestfire.bmp”. (b, c, d): R, G, and B colour channels for this image

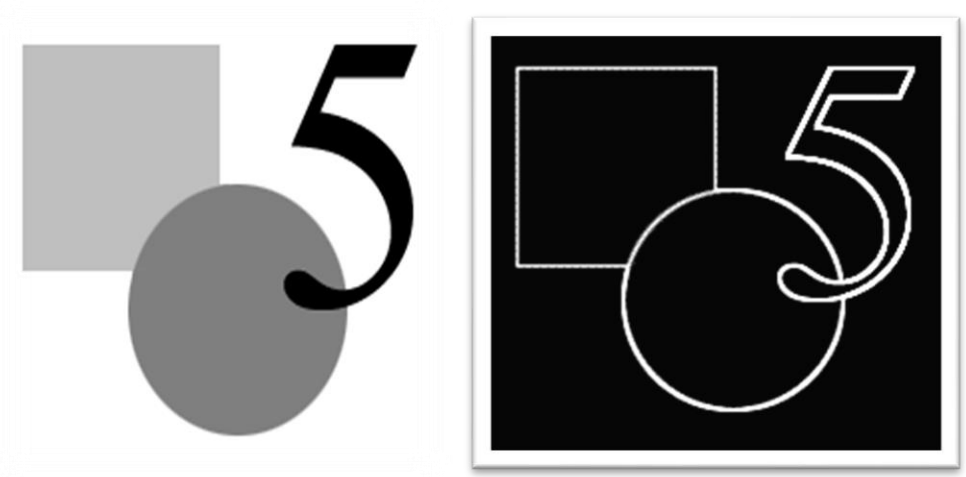
Other Colour Images

- **8-bit colour images**: using 8-bit index to 24-bit real colours.
- **32-bit colour images**: many 24-bit colour images are actually stored as 32-bit images, with the extra byte of data for each pixel used to store an *alpha* value representing special effect information (e.g., transparency).

Edge Detection

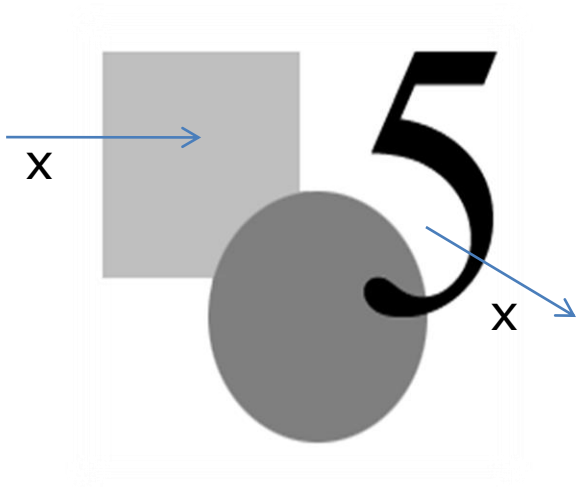
Edges in Images

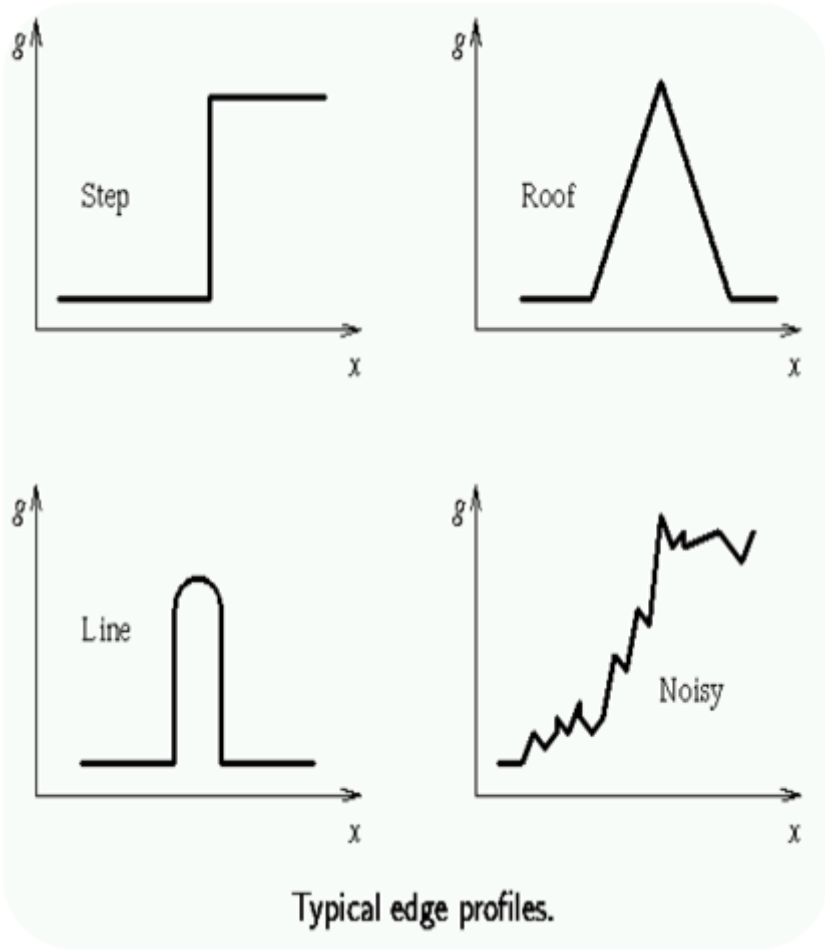
- Edge - Area of significant change in the image intensity / contrast
- Edge Detection – Locating areas with strong intensity contrasts
- Use of Edge Detection – Extracting information about the image. E.g. location of objects present in the image, their shape, size, image sharpening and enhancement



Types of Edges

- Variation of Intensity / Gray Level
  - Step Edge
  - Line Edge
  - Roof Edge





Typical edge profiles.

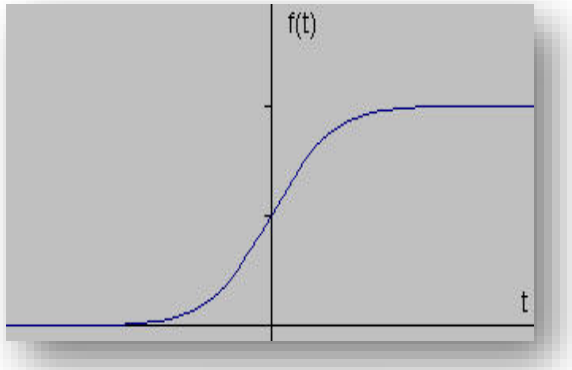
Steps in Edge Detection

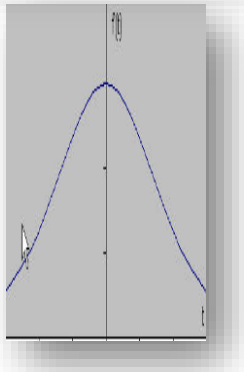
- Filtering – Filter image to improve performance of the Edge Detector wrt noise
- Enhancement – Emphasize pixels having significant change in local intensity
- Detection – Identify edges - thresholding
- Localization – Locate the edge accurately, estimate edge orientation

Methods of Edge Detection

- First Order Derivative / Gradient Methods
  - Roberts Operator
  - Sobel Operator
  - Prewitt Operator
- Second Order Derivative
  - Laplacian
  - Laplacian of Gaussian
  - Difference of Gaussian
- Optimal Edge Detection
  - Canny Edge Detection

First Derivative





- At the point of greatest slope, the first derivative has maximum value
  - E.g. For a Continuous 1-dimensional function f(t)

Gradient

- For a continuous two dimensional function Gradient is defined as

$$G[f(x,y)] = \begin{bmatrix} Gx \\ Gy \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$|G| = \sqrt{Gx^2 + Gy^2} \approx |Gx| + |Gy|$$

$$\theta = \tan^{-1}\left(\frac{Gy}{Gx}\right)$$

Gradient

- Approximation of Gradient for a discrete two dimensional function
- **Convolution** Mask

$$Gx \cong f[i, j+1] - f[i, j]$$

$$Gy \cong f[i, j] - f[i+1, j]$$

Gx=

-1	1

-1	1
-1	1

Gy =

1
-1

1	1
-1	-1

- Differences are computed at the interpolated points [i, j+1/2] and [i+1/2, j]

Roberts Operator

- **Provides an approximation to the gradient**

$$G[f(i,j)] = |Gx| + |Gy| = |f(i,j) - f(i+1,j+1)| + |f(i+1,j) - f(i,j+1)|$$

- **Convolution Mask**

Gx=

1	0
0	-1

Gy =

0	-1
1	0

Roberts Operator - Example

- The output image has been scaled by a factor of 5
- Spurious dots indicate that the operator is susceptible to noise

Sobel Operator

- The 3x3 convolution mask smoothes the image by some amount , hence it is less susceptible to noise. But it produces thicker edges. So edge localization is poor
- Convolution Mask
  - **Gx =**

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

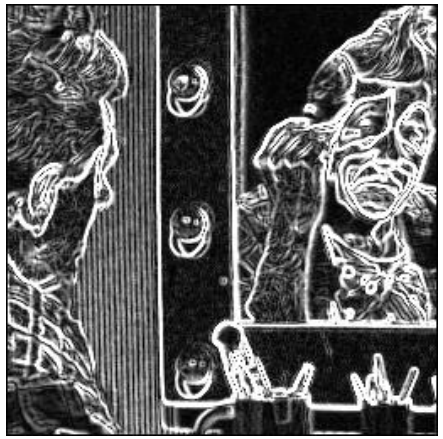
Gy=

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

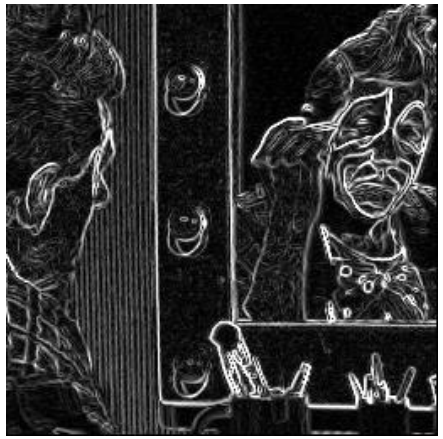
- The differences are calculated at the centre pixel of the mask.

Sobel Operator - Example

- Compare the output of the Sobel Operator with that of the Roberts Operator:
  - The spurious edges are still present but they are relatively less intense compared to genuine lines
  - Roberts operator has missed a few edges
  - Sobel operator detects thicker edges



Sobel



Roberts

Prewitt Operator

- It is similar to the Sobel operator but uses slightly different masks
- Convolution Mask

Px =

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

Py =

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

First Order Derivative Methods

- Noise – simple edge detectors are affected by noise – filters can be used to reduce noise
- Edge Thickness – Edge is several pixels wide for Sobel operator– edge is not localized properly
- Roberts operator is very sensitive to noise
- Sobel operator goes for averaging and emphasizes on the pixel closer to the centre of the mask. It is less affected by noise and is one of the most popular Edge Detectors.

Canny Edge Detector

- Step 1
  - Noise is filtered out – usually a Gaussian filter is used
  - Width is chosen carefully
- Step 2
  - Edge strength is found out by taking the gradient of the image
  - A Roberts mask or a Sobel mask can be used

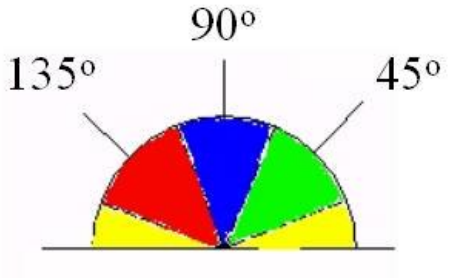
$$|G| = \sqrt{Gx^2 + Gy^2} \approx |Gx| + |Gy|$$

Canny Edge Detector

- Step 3
  - Find the edge direction

$$\theta = \tan^{-1}\left(\frac{Gy}{Gx}\right)$$

- Step 4
  - Resolve edge direction



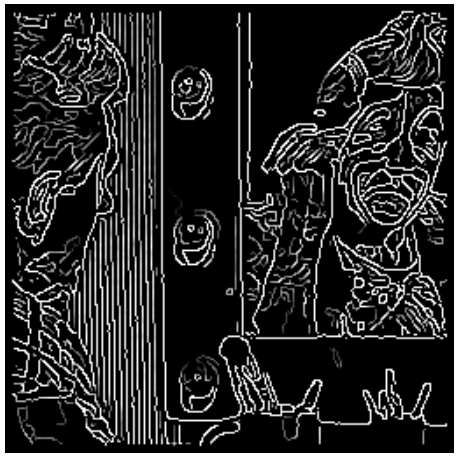
Canny Edge Detector

- Step 5
  - Non-maxima suppression – trace along the edge direction and suppress any pixel value not considered to be an edge. Gives a thin line for edge
- Step 6
  - Use double / hysterisis thresholding to eliminate streaking

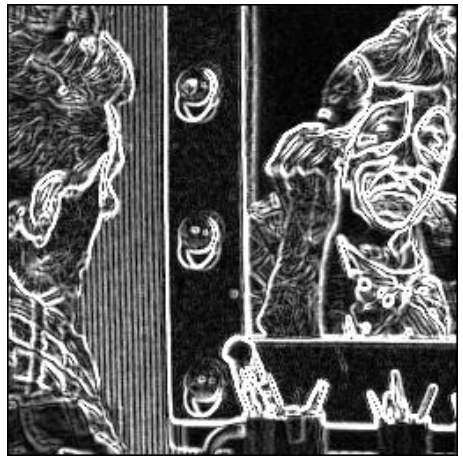
Canny Edge Detector

- Compare the results of Canny and Sobel

Input image



Canny

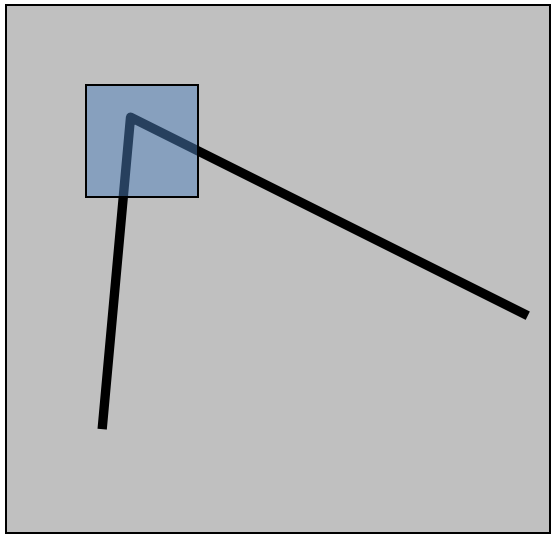


Sobel

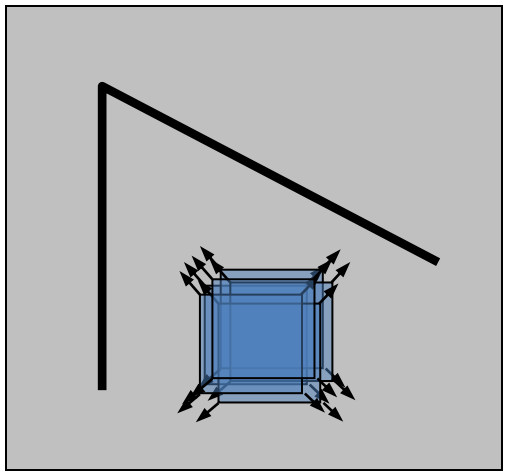
Corner Detection

The Basic Idea

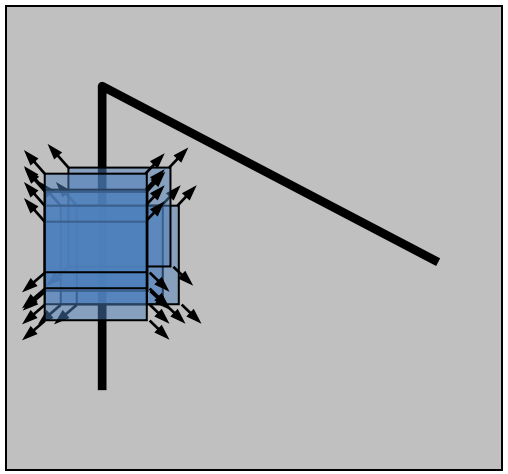
- We should easily localize the point by looking through a small window
- Shifting a window in *any direction* should give *a large change* in intensity



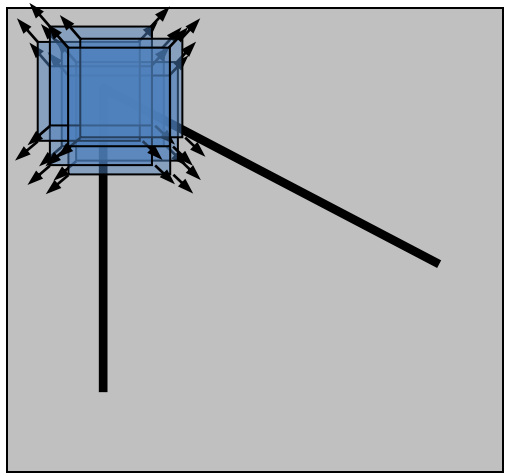
Harris Detector: Basic Idea



"flat" region:
no change as shift window in all directions



"edge":
no change as shift window along the edge direction



"corner":
significant change as shift window in all directions

Harris Detector: Mathematics

Window-averaged change of intensity induced by shifting the image data by $[u,v]$:

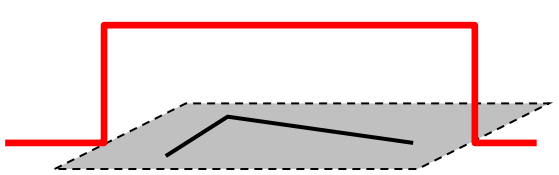
$$E(u,v) = \sum_{x,y} w(x,y)\left[I(x+u,y+v) - I(x,y)\right]^2$$

Window function

Shifted intensity

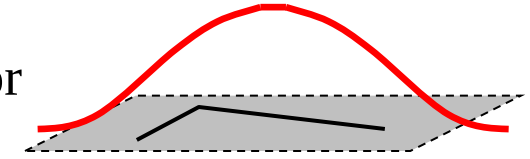
Intensity

Window function $w(x,y)$ =



1 in window, 0 outside

or



Gaussian

Taylor Series Approximation

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

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

$$= \sum_{x,y} w(x,y) \begin{pmatrix} u & v \end{pmatrix} \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

Harris Detector: Mathematics

Expanding I(x,y) in a Taylor series expansion, we have, for small shifts $[u,v]$, a *bilinear* approximation:

$$E(u,v) \cong \begin{bmatrix} u,v \end{bmatrix} \quad M \quad \begin{bmatrix} u \\ v \end{bmatrix}$$

where *M* is a 2×2 matrix computed from image derivatives:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

M is also called “structure tensor”

Harris Detector: Mathematics

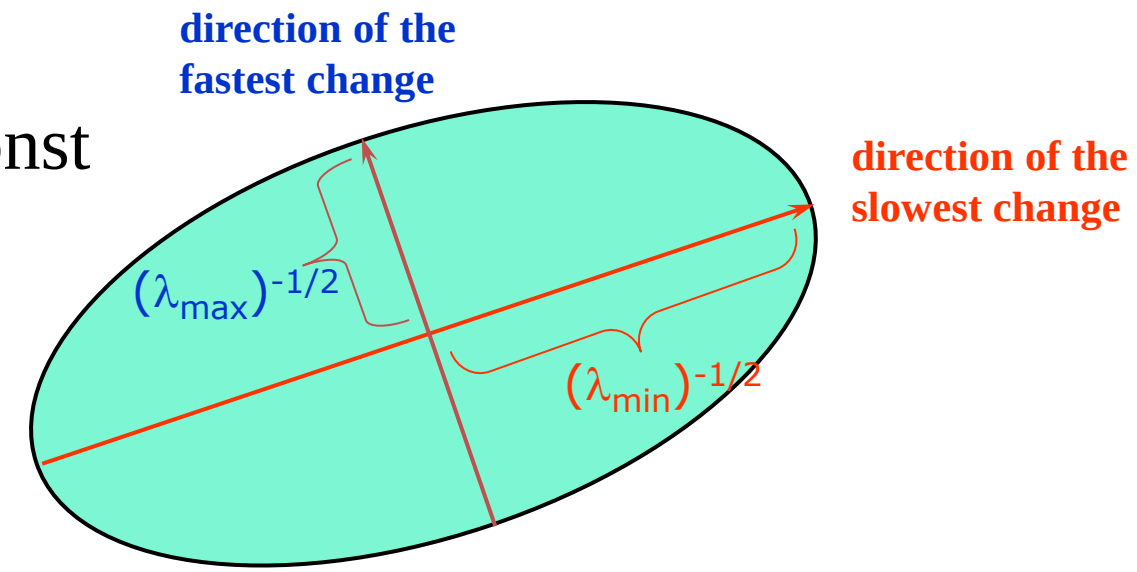
Intensity change in shifting window: eigenvalue analysis

$$E(u,v) \cong \begin{bmatrix} u, v \end{bmatrix} \; M \; \begin{bmatrix} u \\ v \end{bmatrix}$$

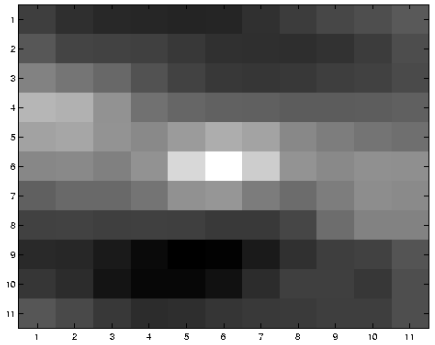
λ_1, λ_2 – eigenvalues of M

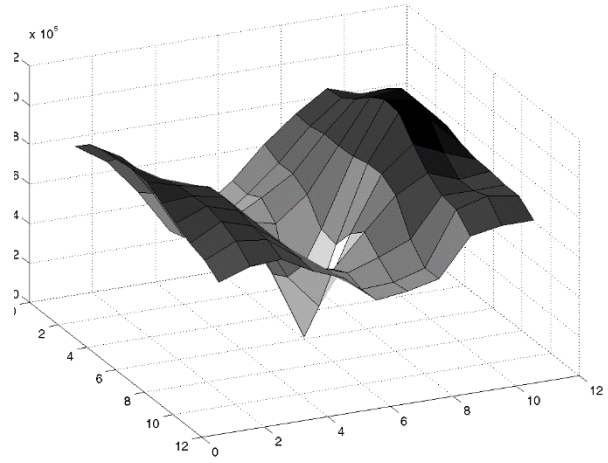
Ellipse $E(u,v)$ = const

Iso-intensity contour of $E(u,v)$



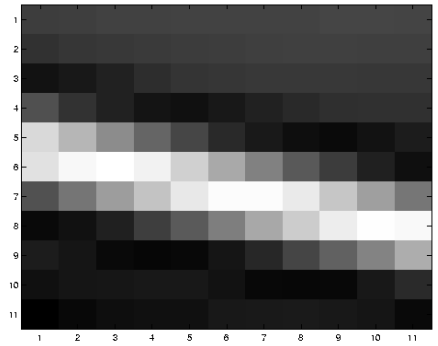
Selecting Good Features

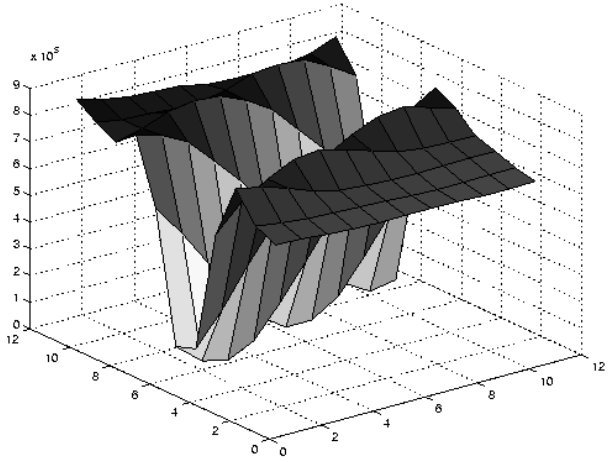




λ_1 and λ_2 are large

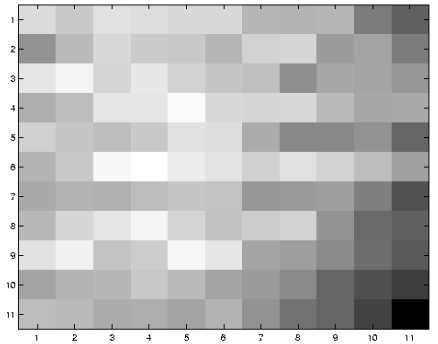
Selecting Good Features

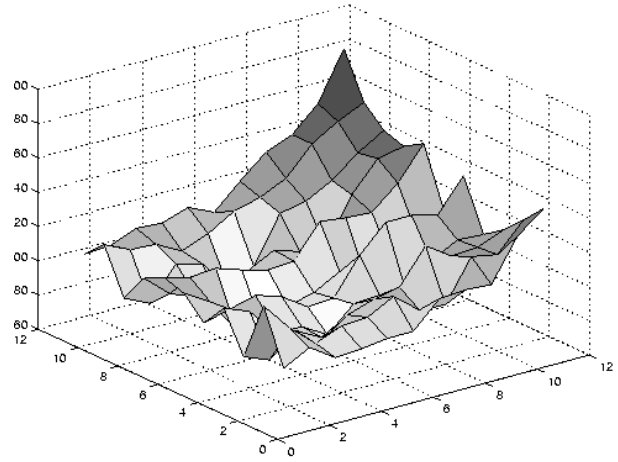




large λ_1, small λ_2

Selecting Good Features

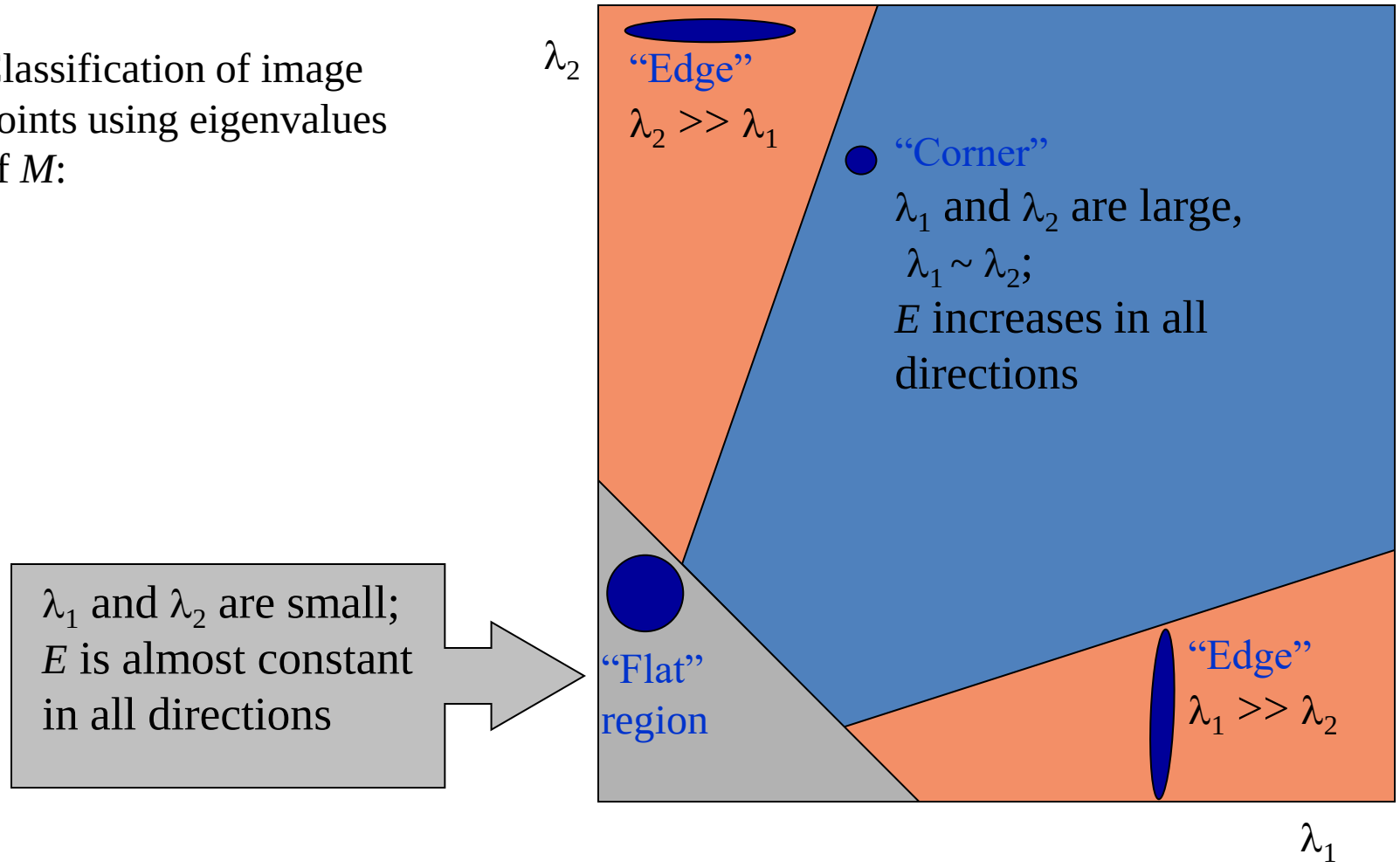




small λ_1, small λ_2

Harris Detector: Mathematics

Classification of image points using eigenvalues of M:



Harris Detector: Mathematics

Measure of corner response:

$$R = \det M - k\left(\operatorname{trace} M\right)^2$$

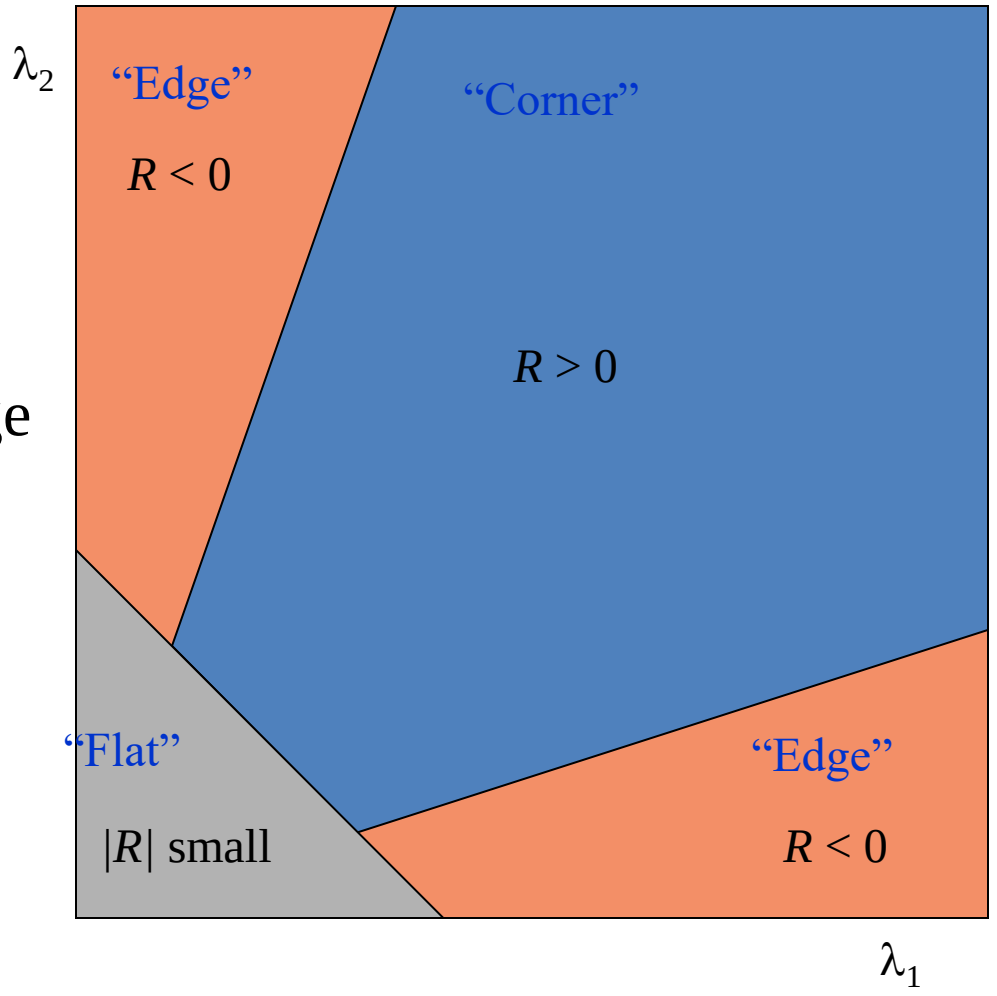
This expression does not requires computing the eigenvalues.

$$\det M = \lambda_1 \lambda_2$$
$$\operatorname{trace} M = \lambda_1 + \lambda_2$$

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

Harris Detector: Mathematics

- 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



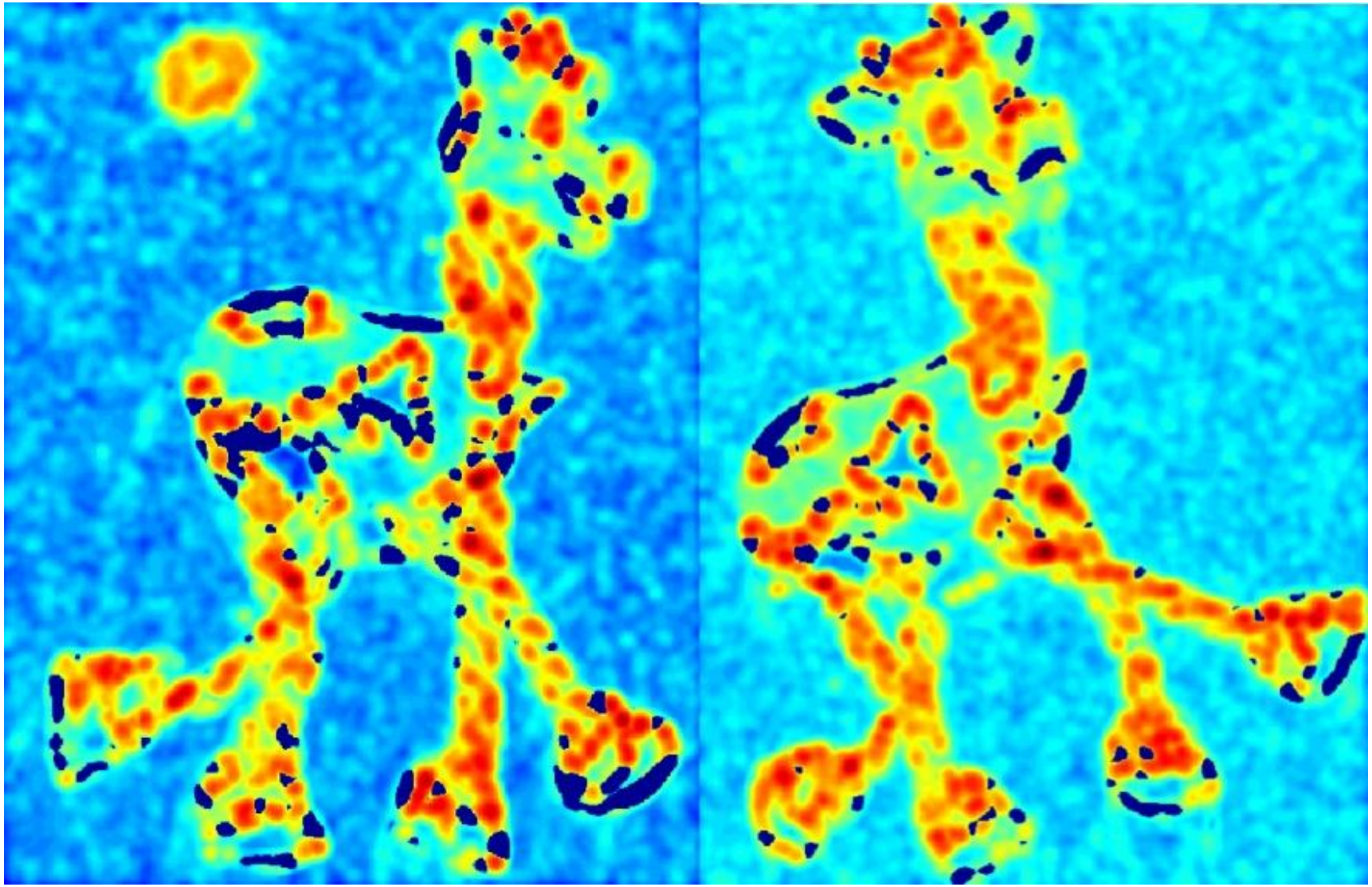
Harris Detector

- The Algorithm:
  - Find points with large corner response function R (R > threshold)
  - Take the points of local maxima of R

Harris Detector: Workflow

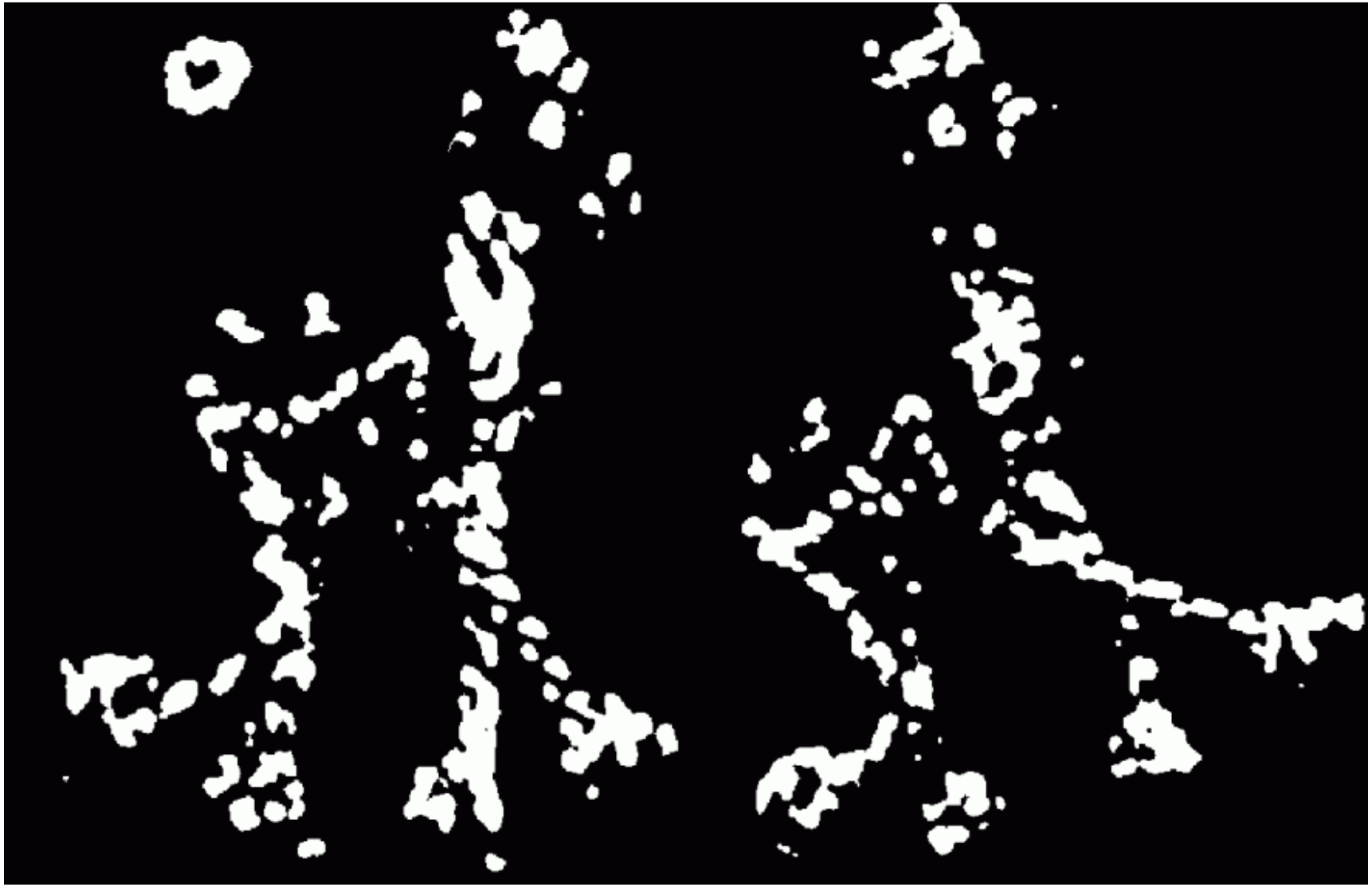
Harris Detector: Workflow

Compute corner response R



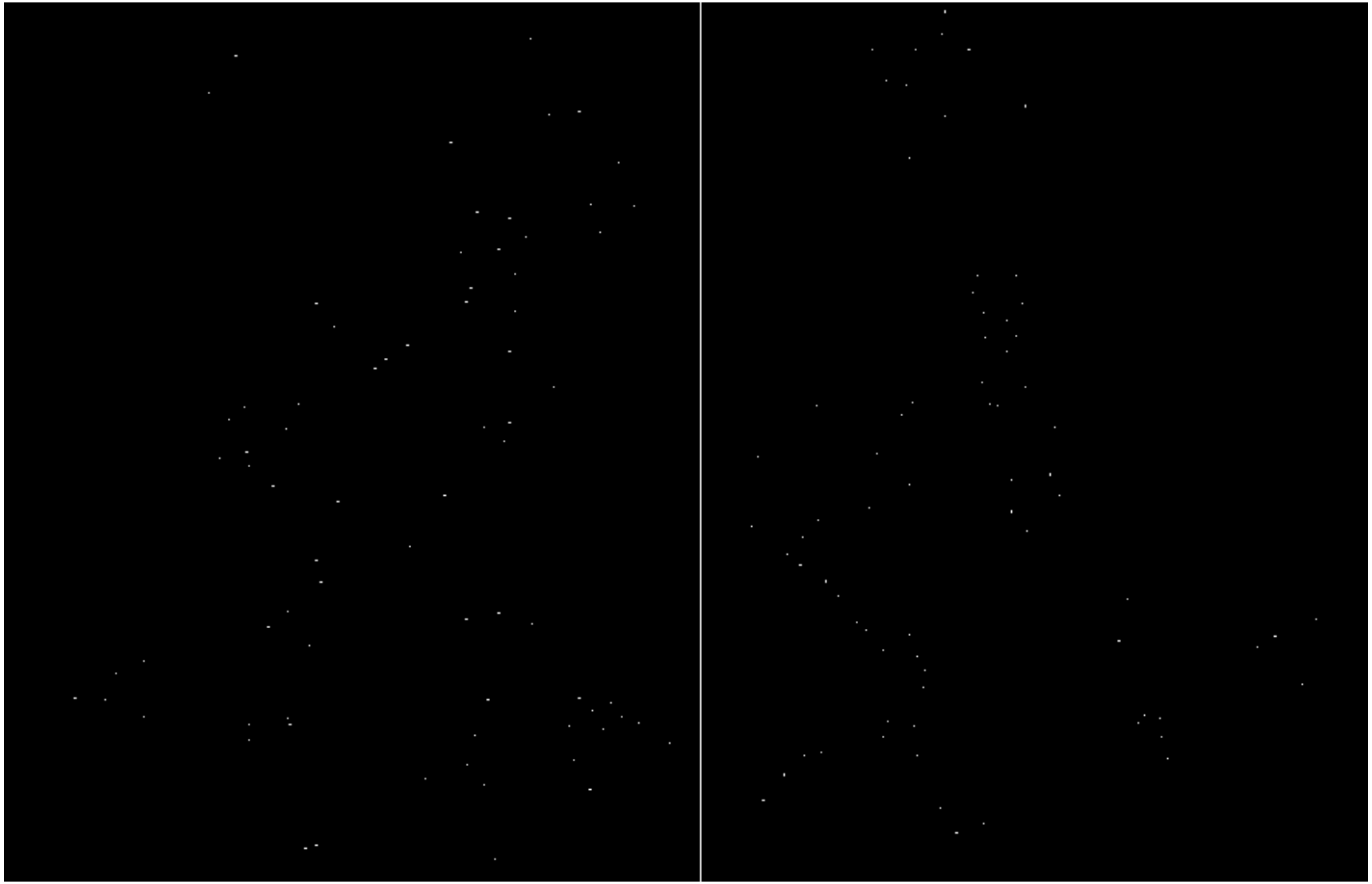
Harris Detector: Workflow

Find points with large corner response: R>threshold



Harris Detector: Workflow

Take only the points of local maxima of R



Harris Detector: Workflow

Harris Detector: Summary (1)

- Estimate the structure tensor M – an indicator of intensity changes:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

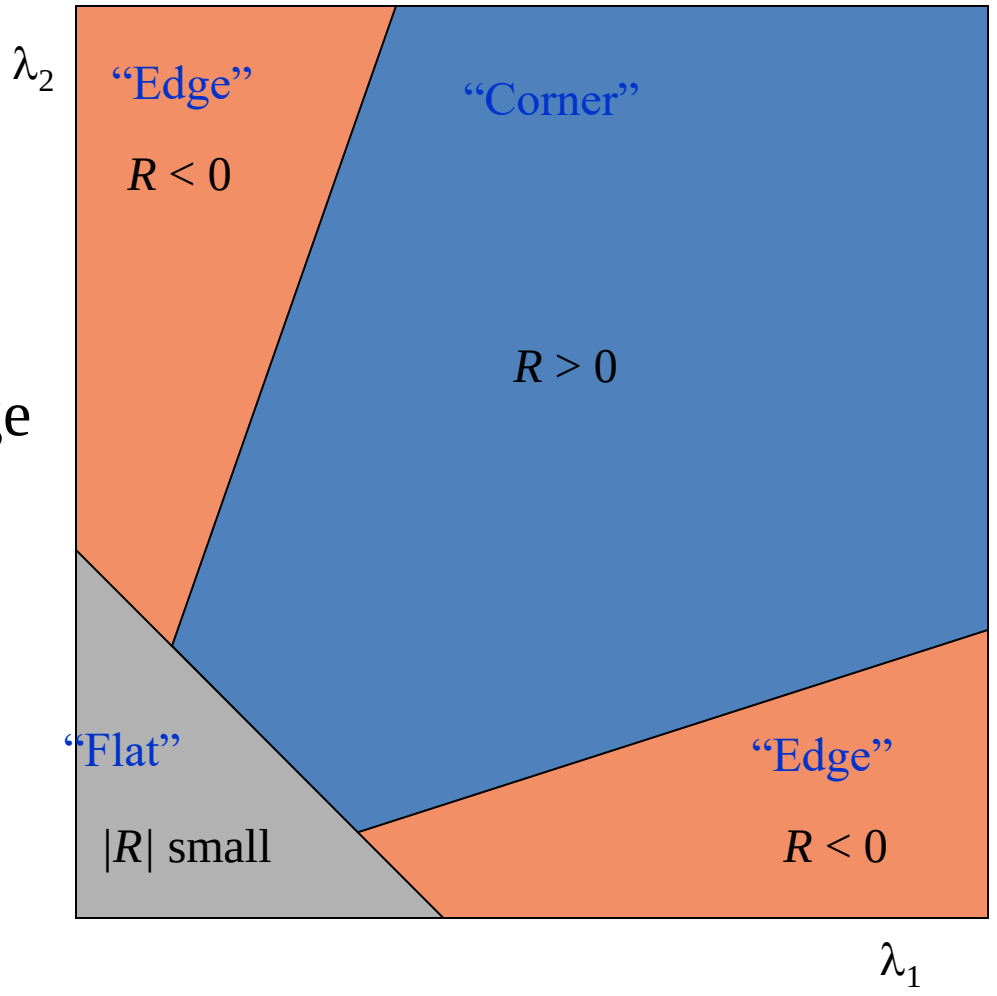
- Calculate the *corner response* R from the eigenvalues of M:

$$R = \lambda_1 \lambda_2 - k\left(\lambda_1 + \lambda_2\right)^2$$

- A good (corner) point should have a *large intensity change* in *all directions*, i.e. R should be large positive

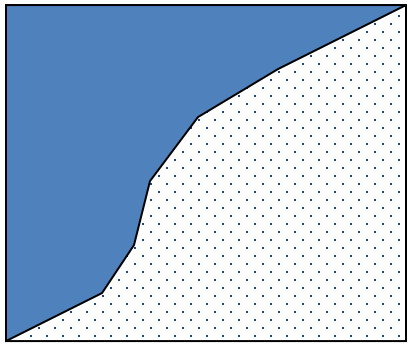
Harris Detector: Summary (2)

- 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



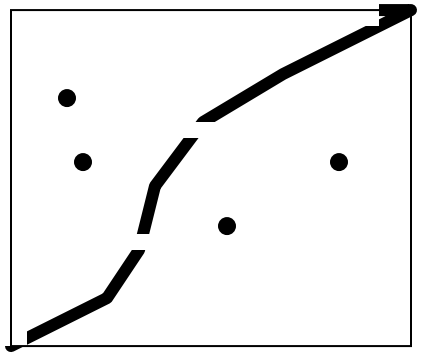
Line Detection

Preprocessing Edge Images



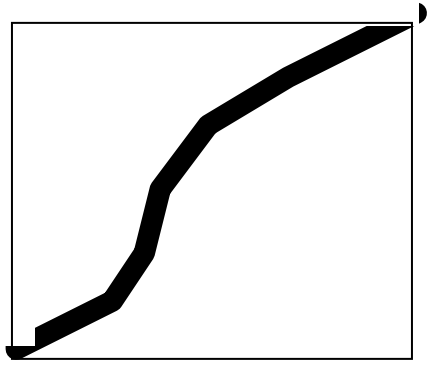
Image

Edge Detection
and Thresholding

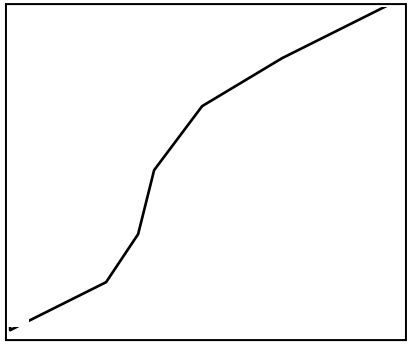


Noisy edge image
Incomplete boundaries

Shrink and Expand



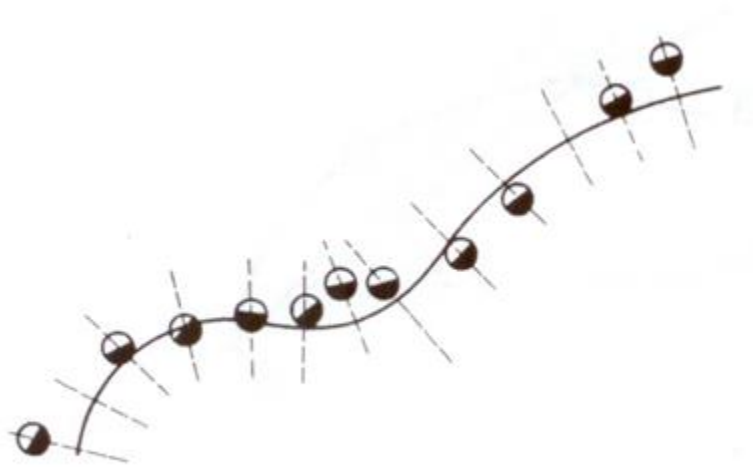
Thinning



Edge Tracking Methods

Adjusting a priori Boundaries:

Given: Approximate Location of Boundary
Task: Find Accurate Location of Boundary



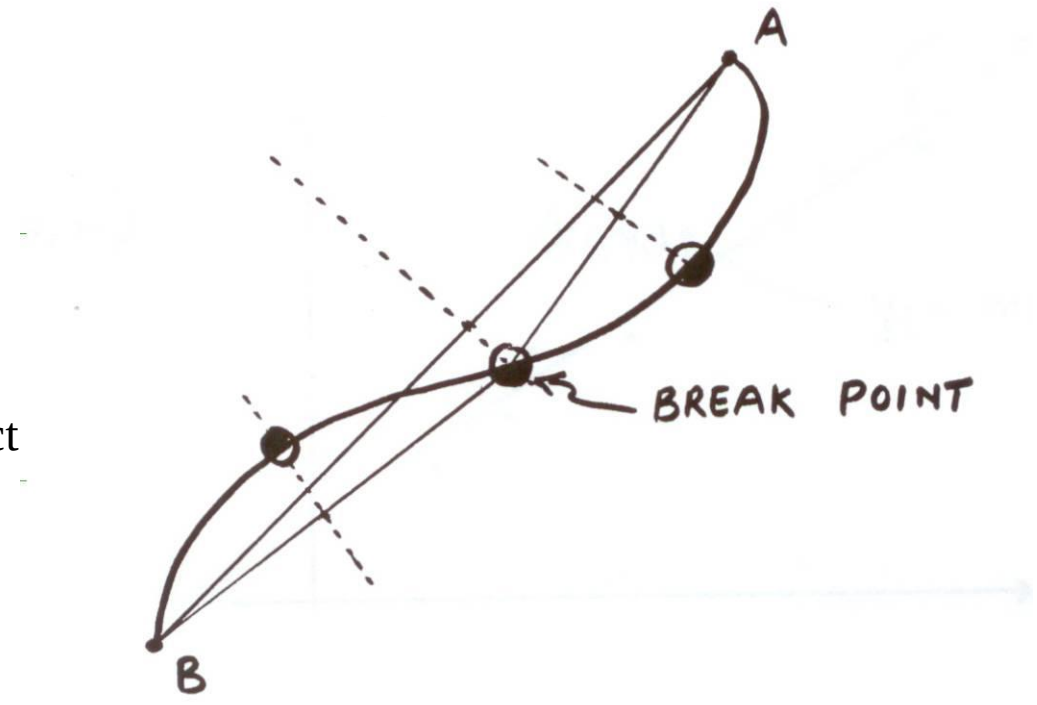
- Search for STRONG EDGES along normals to approximate boundary.
- Fit curve (eg., polynomials) to strong edges.

Edge Tracking Methods

Divide and Conquer:

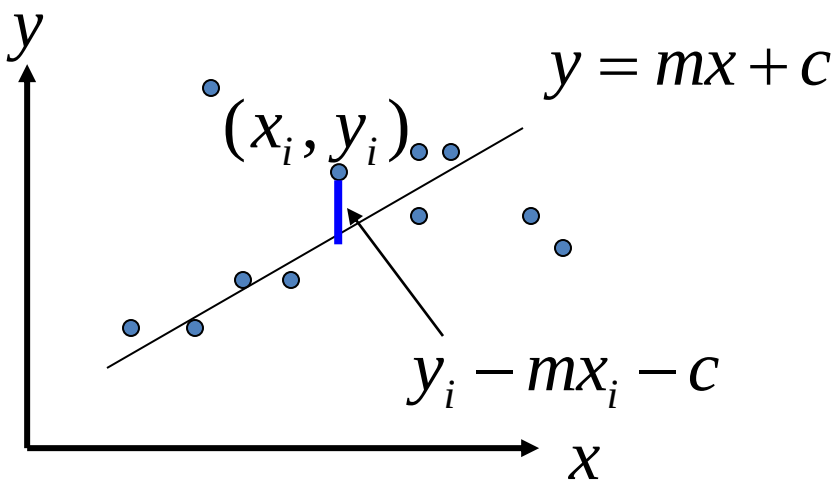
Given: Boundary lies between points A and B
Task: Find Boundary

- Connect A and B with Line
- Find strongest edge along line bisect
- Use edge point as break point
- Repeat



Fitting Lines (Least Squares)

Given: Many (x_i, y_i) pairs
Find: Parameters (m, c)



Minimize: Average square distance:

$$E = \sum_i \frac{(y_i - mx_i - c)^2}{N}$$

Using:

$$\frac{\partial E}{\partial m} = 0 \quad \& \quad \frac{\partial E}{\partial c} = 0$$

Note:

$$\bar{y} = \frac{\sum_i y_i}{N} \qquad \bar{x} = \frac{\sum_i x_i}{N}$$

$$c = \bar{y} - m\,\bar{x}$$

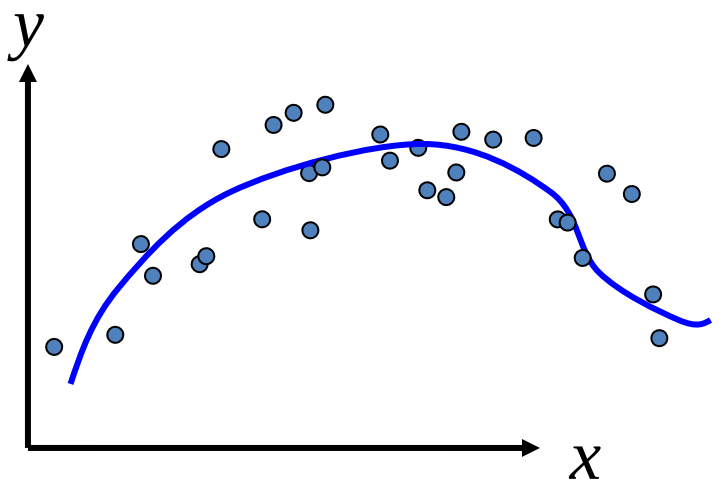
$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2}$$

Curve Fitting

Find Polynomial:

$$y = f(x) = ax^3 + bx^2 + cx + d$$

that best fits the given points (x_i, y_i)



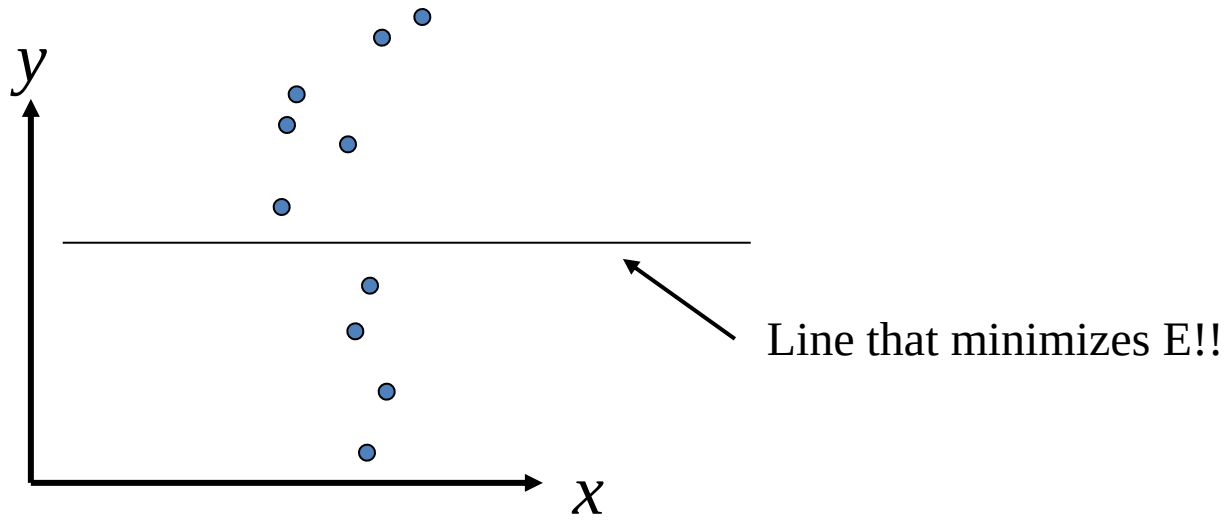
Minimize:

$$\frac{1}{N}\sum_i [y_i - (ax_i^3 + bx_i^2 + cx_i + d)]^2$$

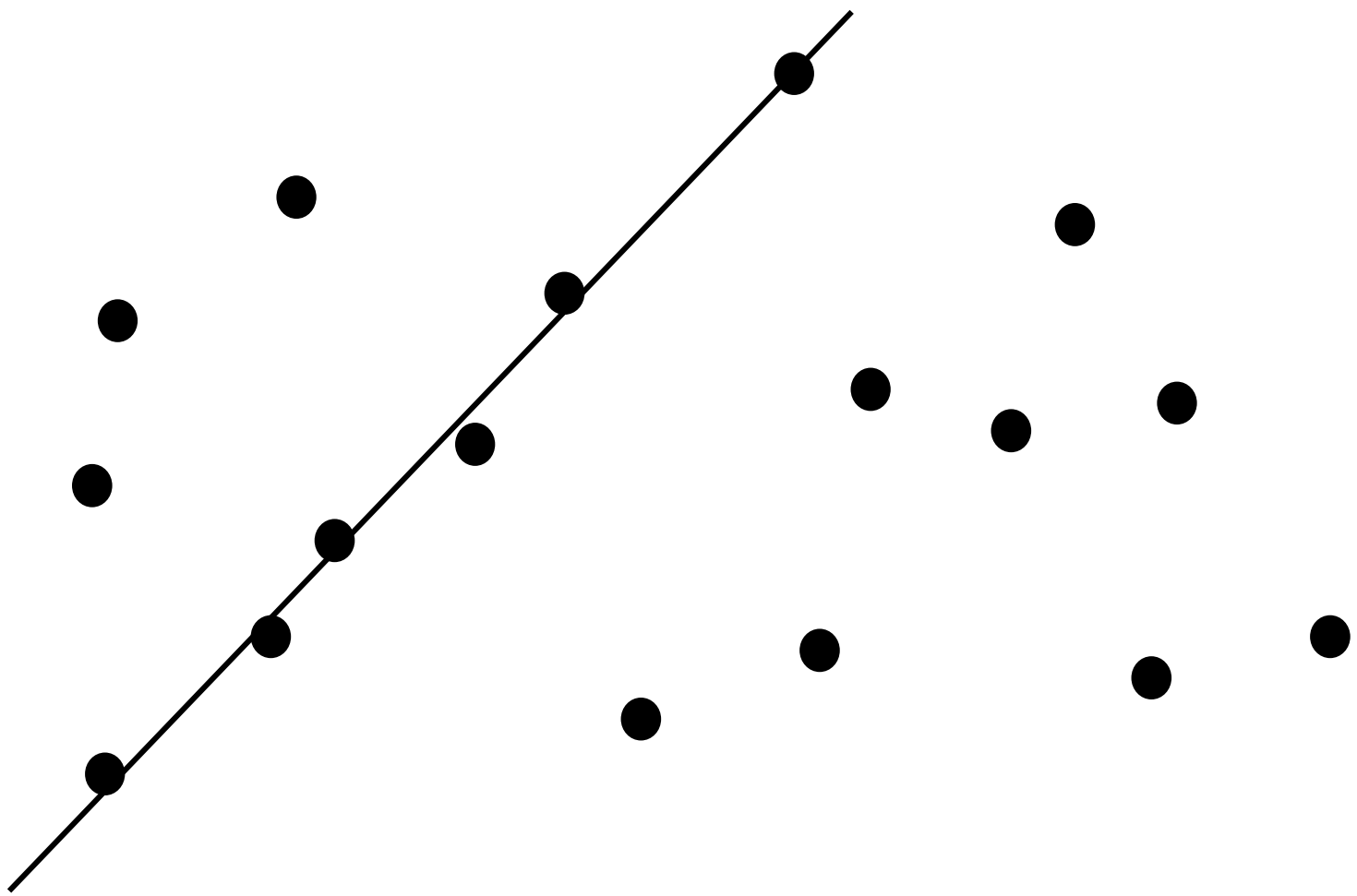
Using: $\frac{\partial E}{\partial a} = 0$, $\frac{\partial E}{\partial b} = 0$, $\frac{\partial E}{\partial c} = 0$, $\frac{\partial E}{\partial d} = 0$

Note: $f(x)$ is LINEAR in the parameters (a, b, c, d)

Problem with Parameterization



Line Grouping Problem



This is difficult because of:

- Extraneous data: clutter or multiple models
  - We do not know what is part of the model?
  - Can we pull out models with a few parts from much larger amounts of background clutter?
- Missing data: only some parts of model are present
- Noise
- **Cost:**
  - It is not feasible to check all combinations of features by fitting a model to each possible subset

Hough Transform

- Elegant method for direct object recognition
- Edges need not be connected
- Complete object need not be visible
- Key Idea: Edges VOTE for the possible model

Image and Parameter Spaces

Equation of Line: $y = mx + c$

Find: (m, c)

Consider point: (x_i, y_i)

$y_i = mx_i + c \quad or \quad c = -x_i m + y_i$

Parameter space also called Hough Space

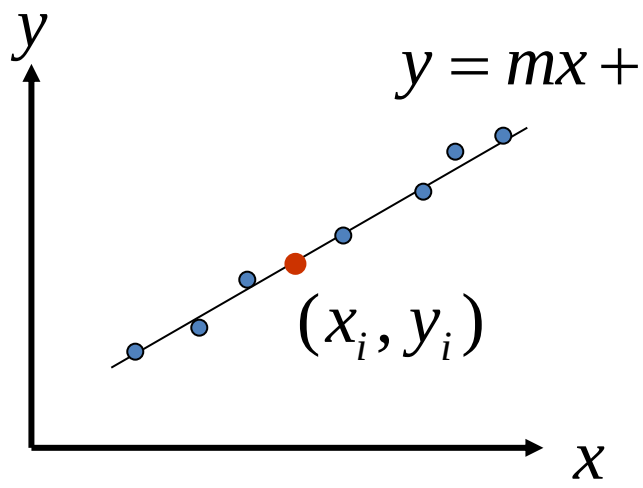
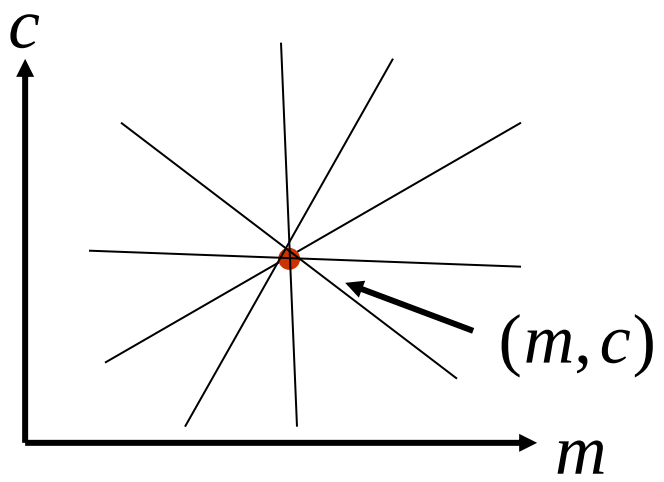


Image Space



Parameter Space

Line Detection by Hough Transform

Algorithm:

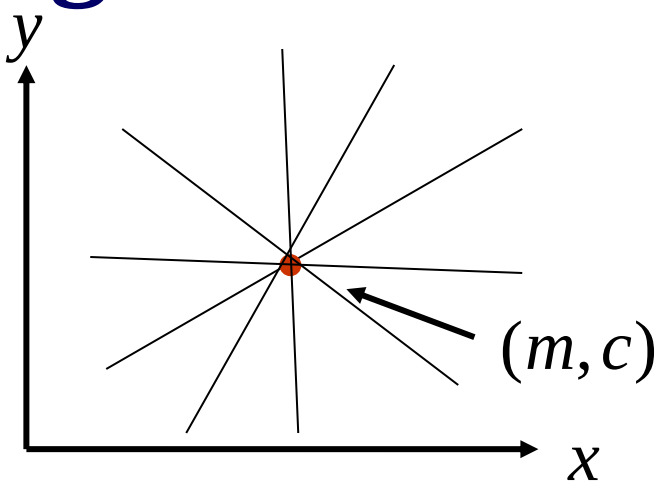
- Quantize Parameter Space (m, c)
- Create Accumulator Array $A(m, c)$
- Set $A(m, c) = 0 \quad \forall m, c$
- For each image edge (x_i, y_i) increment:

$$A(m, c) = A(m, c) + 1$$

If (m, c) lies on the line:

$$c = -x_i m + y_i$$

- Find local maxima in $A(m, c)$



Parameter Space

$A(m, c)$

	1						1		
		1				1			
			1		1				
				2					
			1		1				
		1				1			
	1						1		

Better Parameterization

m: slope
c: intercept

NOTE: $-\infty \le m \le \infty$
Large Accumulator
More memory and computations
(Finite Accumulator Array Size)

Improvement:
Line equation: $\rho = x\cos\theta + y\sin\theta$

Here $0 \le \theta \le 2\pi$

$0 \le \rho \le \rho_{\max}$

Given points (x_i, y_i) find (ρ, θ)

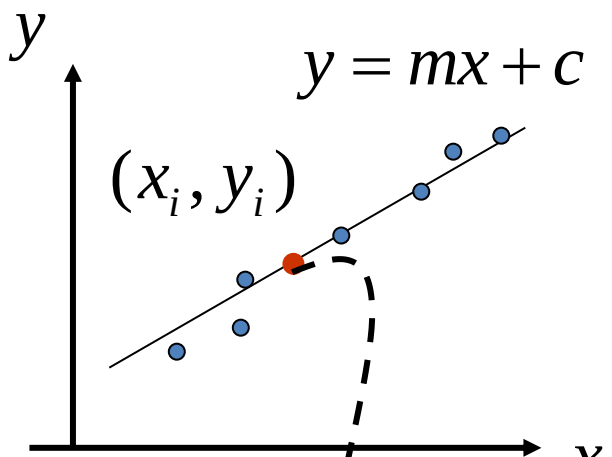
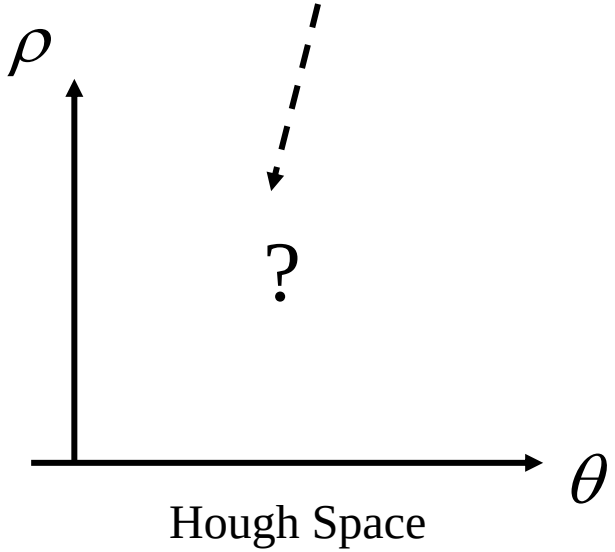


Image Space



Hough Space

Hough Space Sinusoid

Example 1

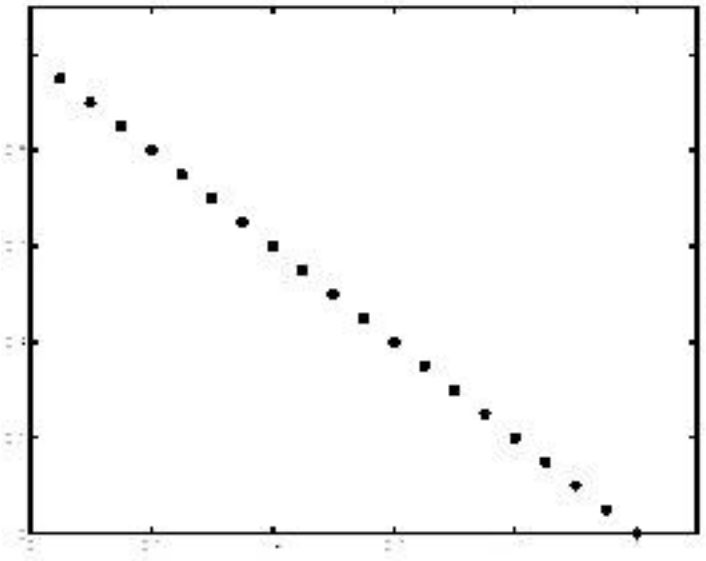
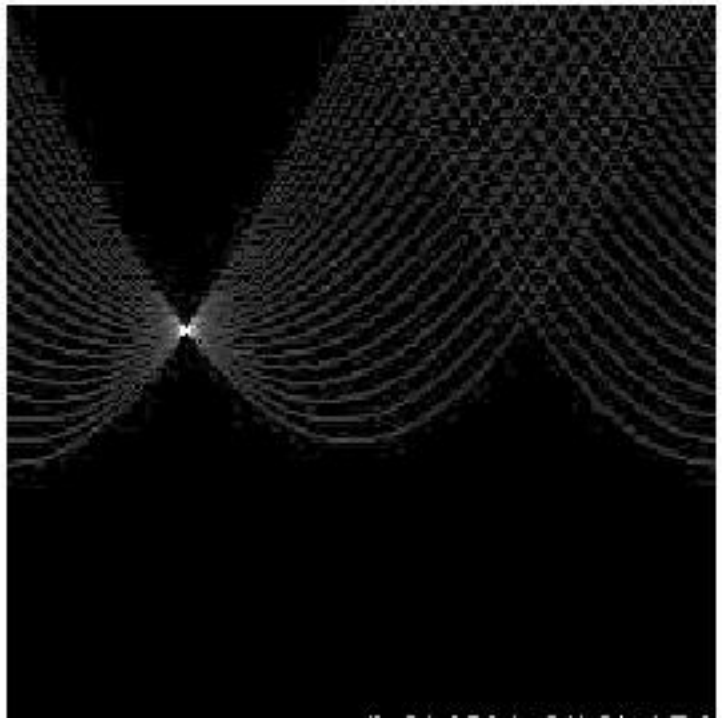


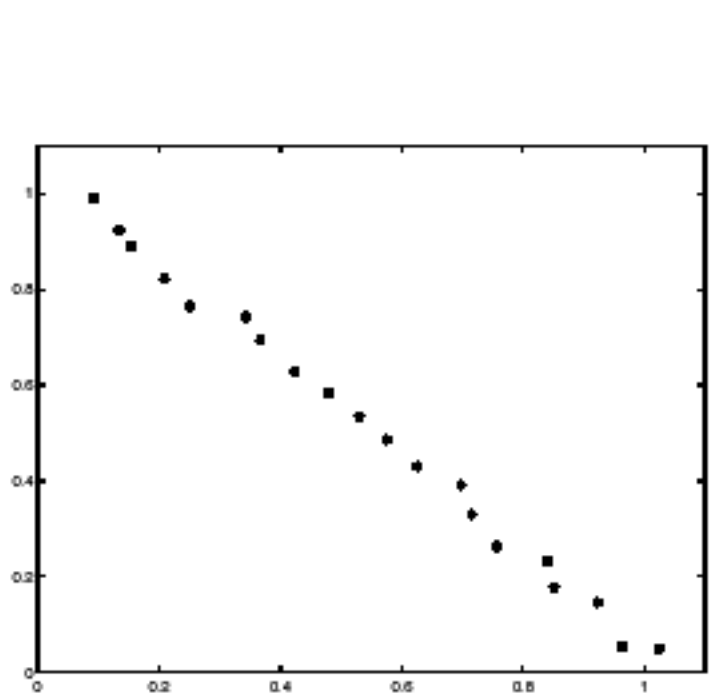
Image space



Votes

Horizontal axis is θ,
vertical is rho.

Example 2



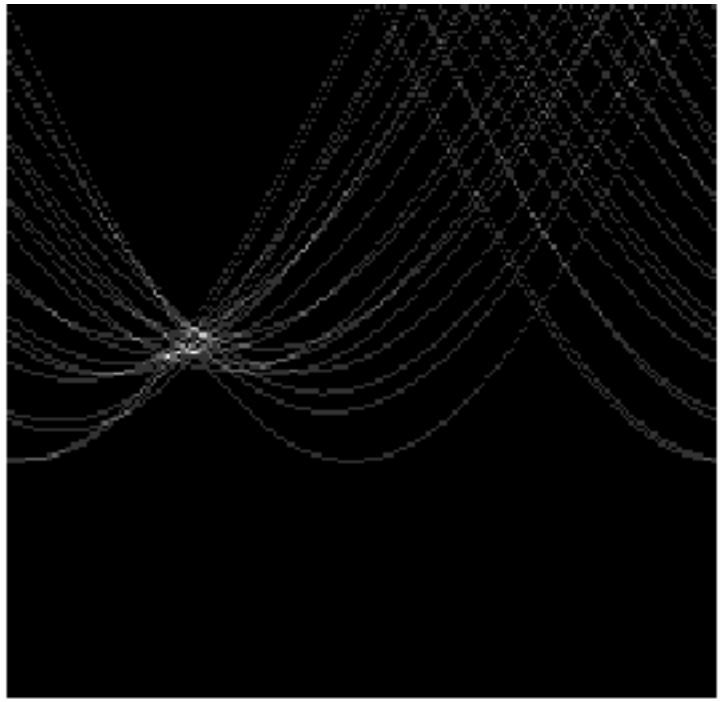
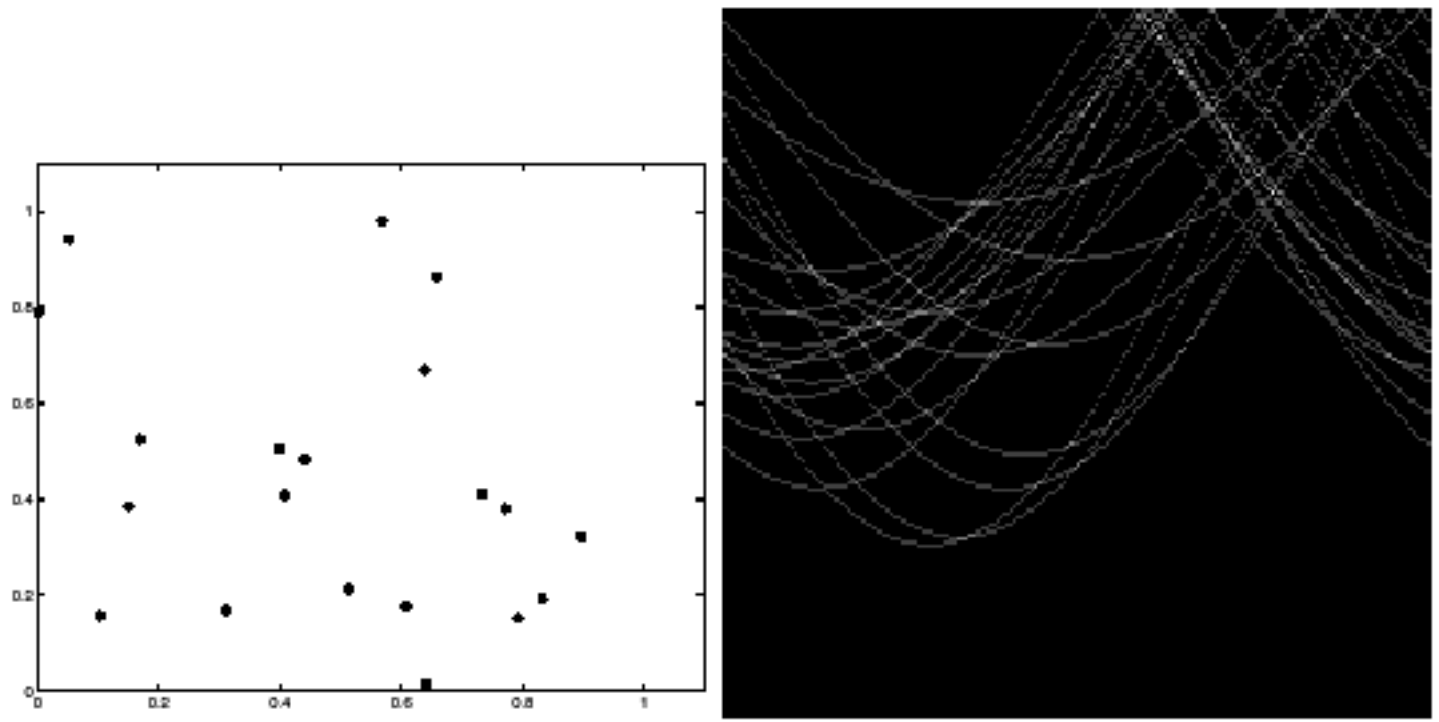


Image space

votes

Example 3



Application Issues

- Difficulties
  - how big should the cells be? (too big, and we merge quite different lines; too small, and noise causes lines to be missed)
- How many lines?
  - Count the peaks in the Hough array
  - Treat adjacent peaks as a single peak
- Which points belong to each line?
  - Search for points close to the line
  - Solve again for line and iterate

Reading

- Not required, but if you really, really want…
- Try this:
  Hlavac, Sonka & Boyle (2014). Image Processing, Analysis, and Machine Vision. 4th edition. Nelson Engineering.
  Chapter 5 (5.1, 5.2 and 5.3)