HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Computer Vision

Chapter 4: Feature detection and Image matching

Plan

- Edges
  - Detection
  - linking
- Feature extraction
  - Global features
  - Local features
- Image Matching and Applications

Edge detection

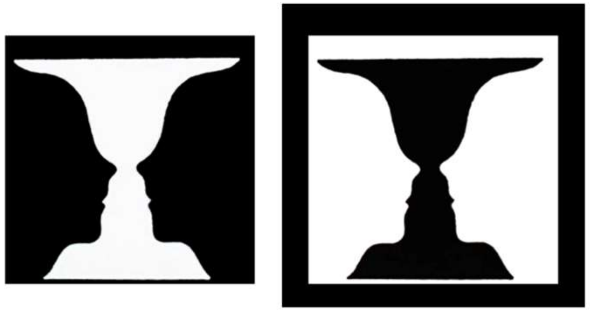
Role of edges & general approach
Image gradient
Edges detectors: sobel, prewitt, Canny detector

What are edges/contours ?

- Local changes in the images
- Typically occur on the boundary between different regions in an image
- Origins of edges:

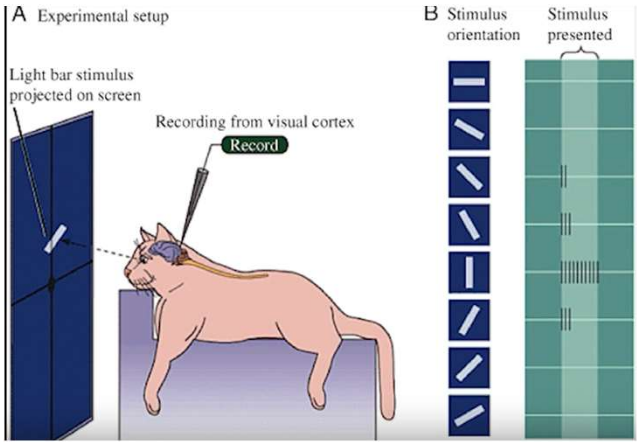
Edge is important?

- What do you see ?

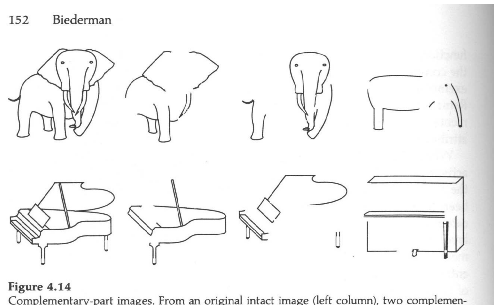


Edge is important?

(A)Cave painting at Chauvet, France, about 30,000 B.C.;
(B)Aerial photograph of the picture of a monkey as part of the Nazca Lines geoglyphs, Peru, about 700 – 200 B.C.;
(C)Shen Zhou (1427-1509 A.D.): Poet on a mountain top, ink on paper, China;
(D)Line drawing by 7-year old I. Lleras (2010 A.D.).



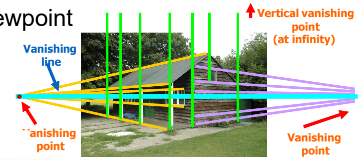
Hubel & Wiesel, 1960s



Can we recognize these objects?

Edge detection

- Goal: Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- Why?
  - Extract information, recognize objects
  - Recover geometry and viewpoint



Source: J. Hayes

How to find edges?

- **Intensity profile** of an image is the set of intensity values taken from regularly spaced points along a line segment or multi-line path in an image

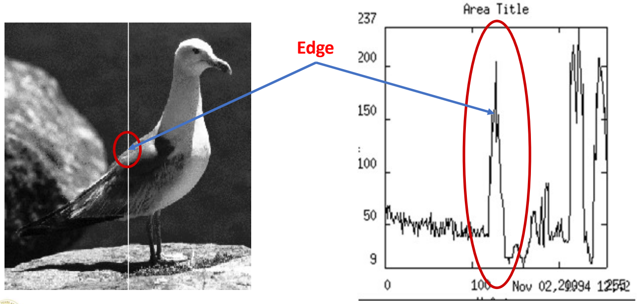
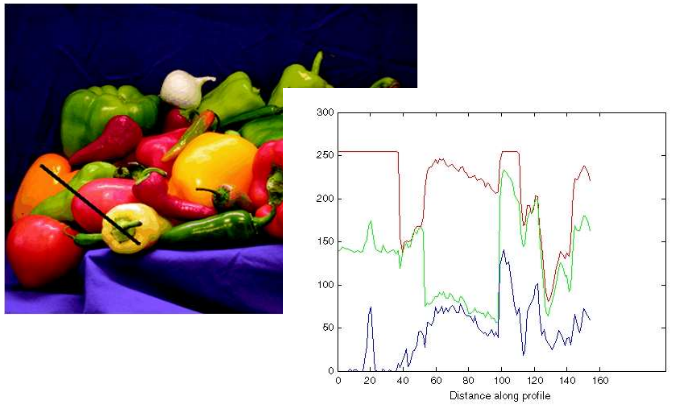


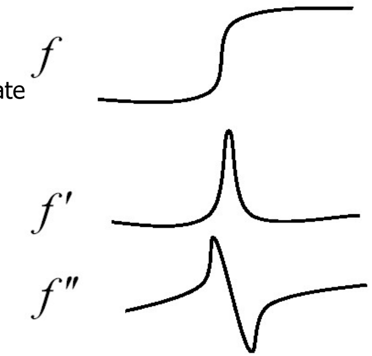
Image profile



https://www.mathworks.com/help/images/intensity-profile-of-images.html

How to find edges?

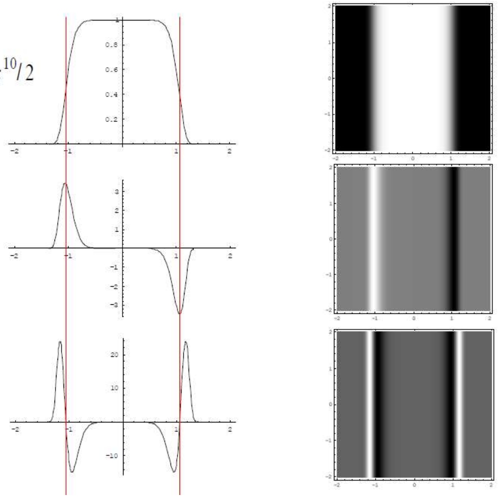
- An edge is a place of rapid change in the image intensity function
  - Extrema of 1st derivate
  - Zero-crossing of 2nd derivate



$$f(x,y)=e^{-x^{10}/2}$$

$$\frac{\partial f}{\partial x}$$

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



Image

The first derivative

The second derivative

Source : Caroline Rougier. Traitement d'images (IFT2730). Univ. de Montréal.

Image gradients

Derivatives

- Derivative in 1D:

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x-\Delta x)}{\Delta x} = f'(x) = f_x$$

- Discrete derivative in 1D

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Types of Discrete derivative in 1D

Backward $\frac{df}{dx} = f(x) - f(x-1) = f'(x)$

Forward $\frac{df}{dx} = f(x) - f(x+1) = f'(x)$

Central $\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$

1D discrete derivate filters

- Backward filter:

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x) \qquad [0 \quad 1 \quad -1]$$

- Forward:

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x) \qquad [-1 \quad 1 \quad 0]$$

- Central:

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x) \qquad [1 \quad 0 \quad -1]$$

1D discrete derivate example

- Backward filter: [0 1 -1]

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$
$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

f(x) :	**0**	**0**	**0**	**0**	**0**	**50**	**50**	**50**	**50**	**50**
f'(x):	**0**	**0**	**0**	**0**	**0**	**50**	**0**	**0**	**0**	**0**

Discrete derivate in 2D

Given function $f(x, y)$

Gradient vector $\nabla f(x,y) = \begin{bmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{bmatrix} = \begin{bmatrix} f_x \\ f_y \end{bmatrix}$

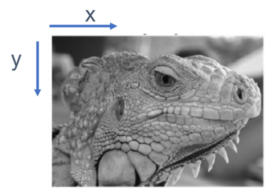
Gradient magnitude $|\nabla f(x,y)| = \sqrt{f_x^2 + f_y^2}$

Gradient direction $\theta = \tan^{-1} \frac{f_x}{f_y}$

Discrete derivate filters wrt. x and y

0	1	-1

0
1
-1



-1	1	0

-1
1
0

1
0
-1

1	0	-1

Discrete derivate filters wrt. x and y

- Robert filter (the first approximation filter for image derivative - 1965)

$$\begin{matrix} \boxed{0} & 1 \\ -1 & 0 \end{matrix} \qquad \begin{matrix} \boxed{1} & 0 \\ 0 & -1 \end{matrix}$$

- Prewitt filter

$$\frac{1}{3} \times \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & -0 & -1 \end{bmatrix} \qquad \frac{1}{3} \times \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

- Sobel filter

$$\frac{1}{4} \times \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & -0 & -1 \end{bmatrix} \qquad \frac{1}{4} \times \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Image I (9 x 8)

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100

M_x: Derivative filter wrt x

1	0	-1

1
0
-1

M_y: Derivative filter wrt y

Ix = I * Mx = the 1st derivative wrt x

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	100	100	0	0	0	0
0	100	100	0	0	0	0
0	100	100	0	0	0	0
0	100	100	0	0	0	0
0	100	100	0	0	0	0

Image I (9 x 8)

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100
0	0	0	100	100	100	100	100	100

M_x: Derivative filter wrt x

1	0	-1

1
0
-1

M_y: Derivative filter wrt y

Iy = I * My = the 1st derivative wrt y

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	100	100	100	100	100
0	0	100	100	100	100	100
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

3x3 image gradient filters

$$\frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Image gradient

- Derivative of image wrt. x + Derivative wrt. y
  - ➔Image gradient: magnitude and direction
- Gradient magnitude: gradient intensity for each pixel (*mostly used*)

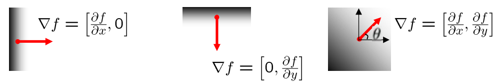
$$|G| = \sqrt{(G_x^2 + G_y^2)} \approx |G_x| + |G_y|$$

- Gradient Direction: main direction of each pixel

$$\theta = \tan^{-1} G_y / G_x$$

Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



- The **gradient vector** points in the direction of most rapid increase in intensity
- The **gradient direction** is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
  - how does this relate to the direction of the edge?
- The *edge strength* is given by the **gradient magnitude**

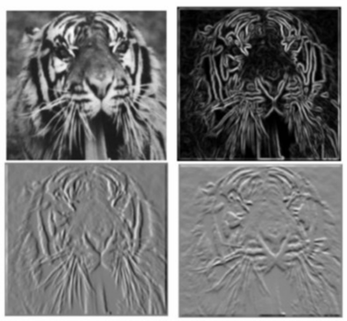
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Source: Steve Seitz

Example

Original Image

Gradient magnitude



x-direction

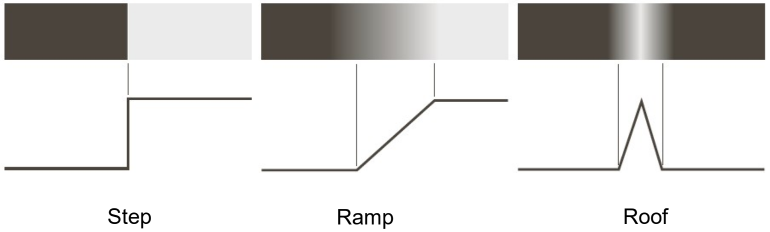
y-direction

- Which one is the gradient in the x-direction? How about y-direction?

CS231: Lecture 3 Juan Carlos Niebles and Ranjay Krishna

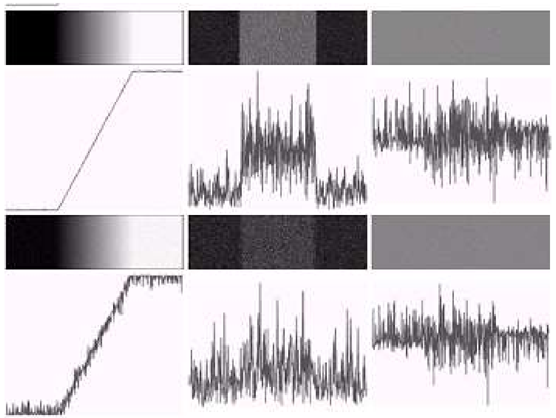
Edge detector

Type of edges



Source : Gonzalez and Woods. Digital Image Processing 3ed. Prentice-Hall, 2008.

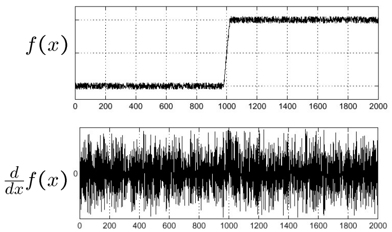
Effects of noise



Source : Gonzalez and Woods. Digital Image Processing 3ed. Prentice-Hall, 2008.

Effects of noise

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

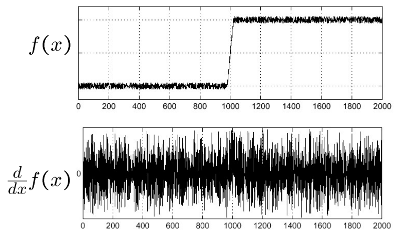


Where is the edge?

Source: S. Seitz

Effects of noise

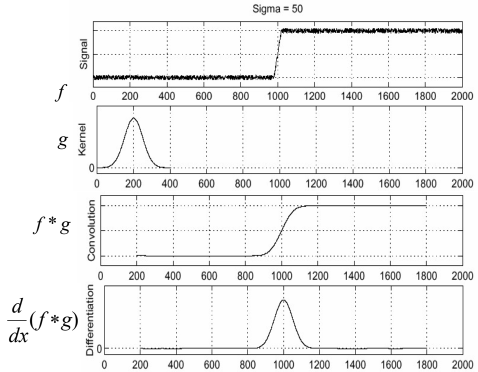
- Solution: smoothing the image



Where is the edge?

Source: S. Seitz

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f*g)$

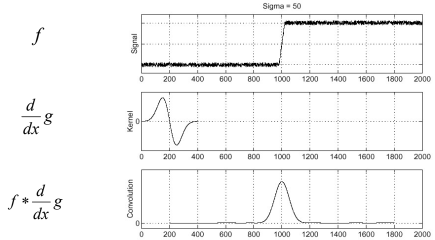
Source: S. Seitz

Derivative theorem of convolution

- This theorem gives us a very useful property:

$$\frac{d}{dx}(f*g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

- This saves us one operation:



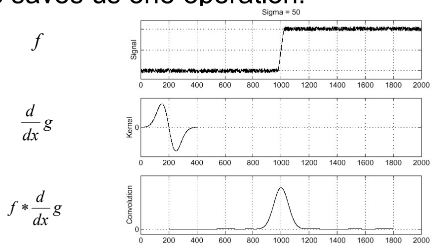
Source: S. Seitz

Derivative theorem of convolution

- This theorem gives us a very useful property:

$$\frac{d}{dx}(f*g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$$

- This saves us one operation:



Source: S. Seitz

Sobel Operator

- Gaussian Smoothing + differentiation

Gaussian smoothing

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1 \quad 0 \quad -1]$$

differentiation

$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} [1 \quad 2 \quad 1]$$

➔ **Less sensible to noise**

Prewitt Operator

- Mean smoothing + differentiation

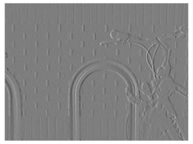
$$Gx = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$Gy = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

➔ **Less sensible to noise**

Simple edge detector with 1st derivative

- Convolve the original image with 2 kernels to calculate approximations of the derivatives

- Compute the gradient magnitude

- Thesholding: choose edges to be the pixel above a threshold T

Simple edge detector with 1st derivative

Original image

Gradient magnitude using Sobel operator

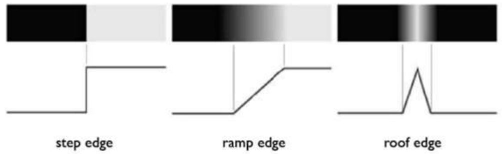
Threshold T = 25

Threshold T = 60

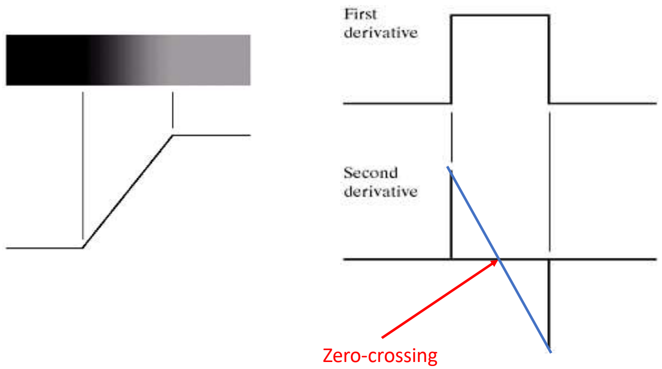
Source : Caroline Rougier. Traitement d'images (IFT2730). Univ. de Montréal.

Problems

- Poor localization (multiple adjacent pixels)
- Thresholding value favors certain directions over others
  - Can miss oblique edges more than horizontal or vertical edges ➔ False negatives



Edge detector with 2nd derivative



Edge detector with 2nd derivative

- 2nd derivative with Laplacian filter:
  - convolution the image with one of 2 filters

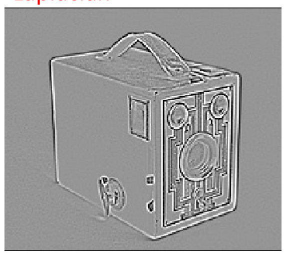
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Edge detection
  - Compute the 2nd derivative of the images
  - Find the zero-crossing pixels ➔ edges

Edge detector with 2nd derivative

Image

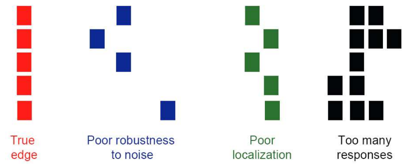
Laplacian



- Single response
- Sensible to noise

“Optimal” edge detector

- Criteria:
  - **Good detection:** the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - **Good localization:** the edges detected must be as close as possible to the true edges
  - **Single response:** the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Canny detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise* ratio and localization

J. Canny, *A Computational Approach To Edge Detection*, IEEE Trans. PAMI, 8:679-714, 1986.

Canny detector

- Optimal:
  - Detection: weak edges detected
  - Good location: close to the real edges
  - Unique response: edge thickness = 1

Canny detector: Steps

1) Apply a gaussian filter on the image

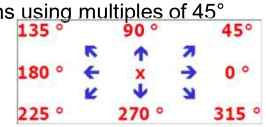
- Lowpass filter to remove noise

2) Compute the gradient intensity in the image

- Sobel filter in X and Y
- Compute the magnitude $|G| = |Gx| + |Gy|$

3) Compute image gradient direction

- Gradient direction $\theta = \arctan(Gy / Gx)$
- Round directions using multiples of 45°

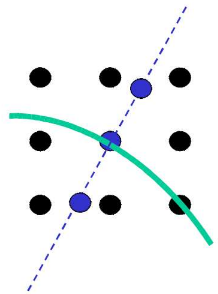


Canny detector: Steps

4) Non-maxima suppression

- If the gradient magnitude of a pixel (x,y) is inferior to the one of its 2 neighbors along the gradient direction

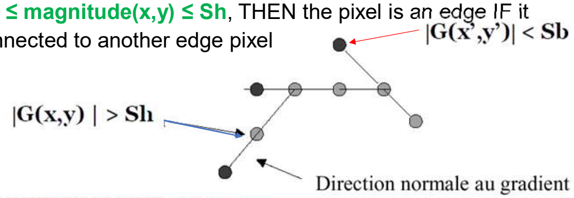
➔ set this magnitude for (x,y) to zero



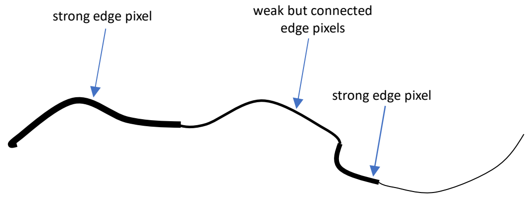
Canny detector: Steps

5) Edge thresholding (hysteresis)

- Using two thresholds: a theshold high (Sh) and a threshold low (Sb)
- For each pixel in the gradient magnitude:
  - IF magnitude(x,y) < Sb, THEN set the pixel to zero (non-edge)
  - IF magnitude(x,y) > Sh, THEN the edge is an edge
  - IF **Sb ≤ magnitude(x,y) ≤ Sh**, THEN the pixel is an edge IF it is connected to another edge pixel

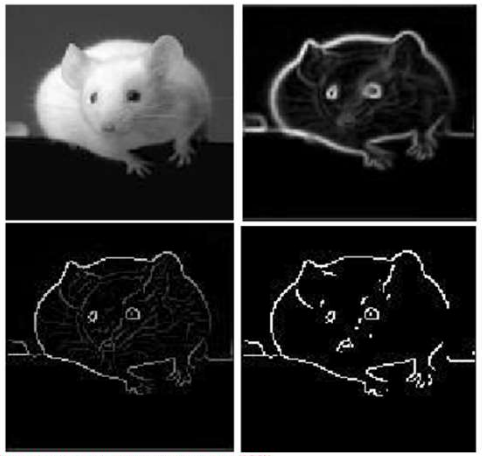


Hysteresis thresholding

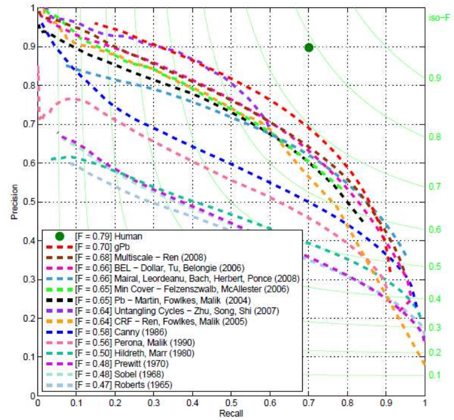


Source: S. Seitz

Canny detector



45 years of boundary detection



Source: Arbelaez, Maire, Fowlkes, and Malik. TPAMI 2011 (pdf)

Edge linking

Hough Transform

RANSAC

Hough transform

Hough transform

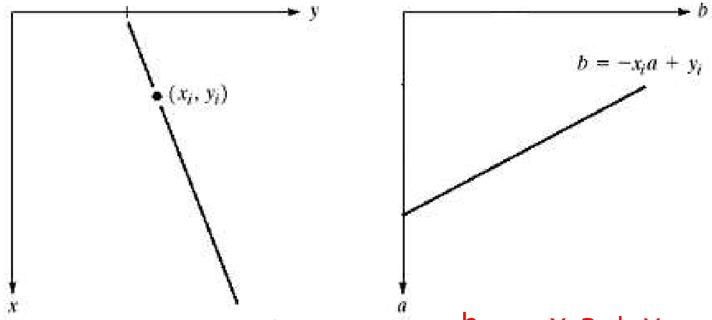
- The Hough transform (HT)
  - can be used to detect lines
  - was introduced in 1962 (Hough 1962) and first used to find lines in images a decade later (Duda 1972)
  - Goal: to find the location of lines in images.
- **Caveat**: Hough transform can *detect lines*, *circles* and other *structures* ONLY if their *parametric equation is known*
- It can give **robust detection** under *noise* and *partial occlusion*

Hough transform

- **Global** approach to detect continuous edges
  - *From the x-y plane to the parametric plane a-b*
- ***x-y plane***
  - $y_i = a x_i + b$
  - an infinity of lines going though one (x_i, y_i) pair
  - one sole line for the (a,b) pair
- ***a-b parametric plane***
  - $b = - x_i a + y_i$
  - one sole line for the (x_i, y_i) pair
  - an infinity of lines going through one (a,b) pair

x-y plane vs a-b plane

x-y plane **a-b parametric plane**

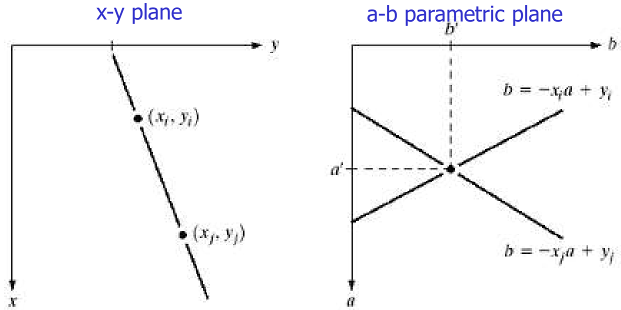


$y_i = a\ x_i + b$ $b = -\ x_i\ a + y_i$

Line vs Points

All the points (x,y) on a line in the x-y plane are going through one sole point (a′, b′) in the a-b parametric plane

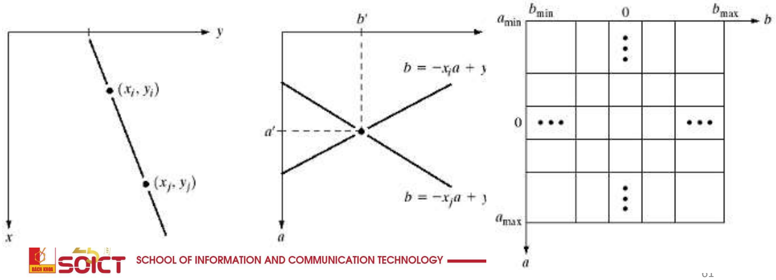
x-y plane a-b parametric plane



$y_i = a\ x_i + b$ $b = -\ x_i\ a + y_i$

Main idea for the Hough transform

- Accumulation cells - Matrix (a,b)
- Build an **voting image**
  - *each point is voting for a particular line*
- The lines receiving more votes are kept



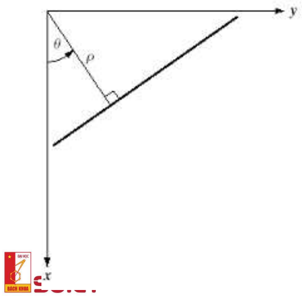
Computing the Hough transform

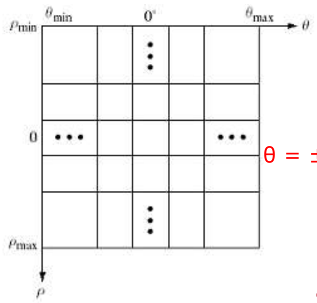
- We compute the contour points of the input image
  - *Sobel, Prewitt, Canny, …*
- For each contour point, we compute a line (a,b)
  - *Result is one line in the a-b plane for each pixel (x,y)*
- The maximum peaks in the a-b parametric plane show the lines with the maximum of points in the x-y plane
  - *The points indicating line crossing in the a-b plane show the real lines existing in the x-y plane*

Problem with the (a,b) space

- Problem: for a vertical line, a=∞ !
- Solution: representing using polar coordinates (ρ,θ)

ρ = x cos θ + y sin θ





θ = ±90

Example with 5 points

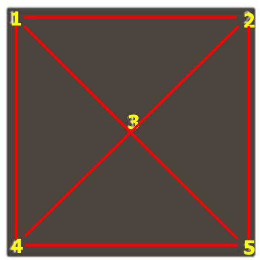
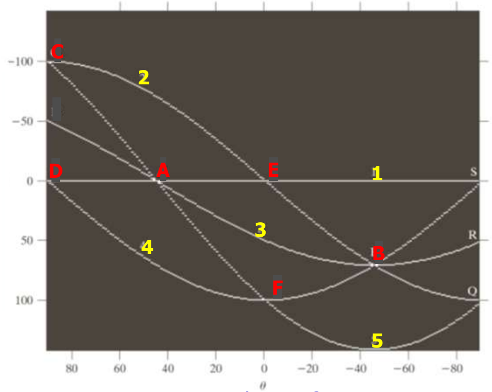


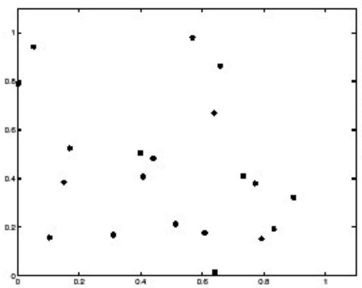
Image containing 5 points

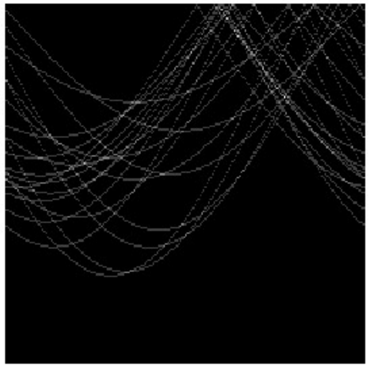


Hough transform

Source : Gonzalez and Woods. Digital Image Processing 3ed. Prentice-Hall, 2008.

Hough transform (random points)

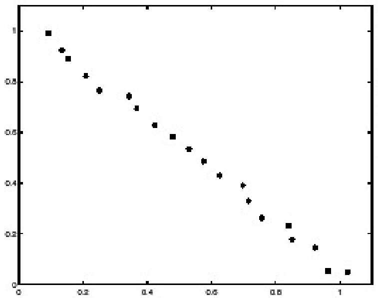


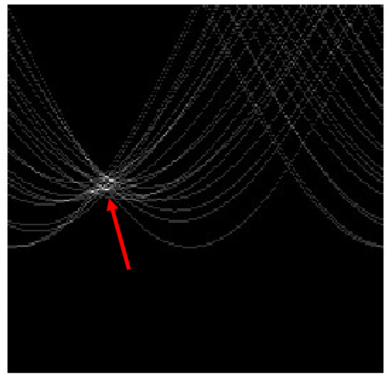


The transform of random points does not give any precise results

Hough transform (straight line)

The transform for aligned points result in a line detection





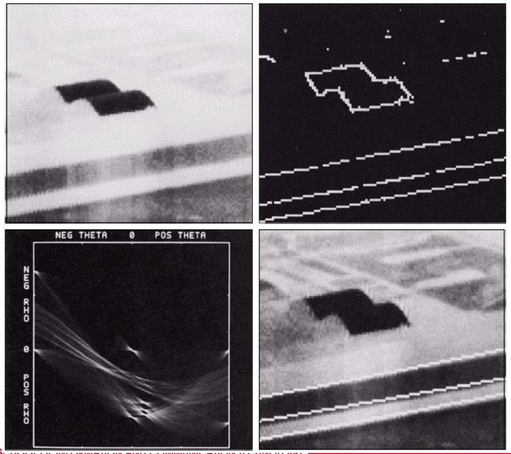
Example

Image

Gradient

Hough

Final



Source : Gonzalez and Woods. Digital Image Processing. Prentice-Hall, 2002.

Other example

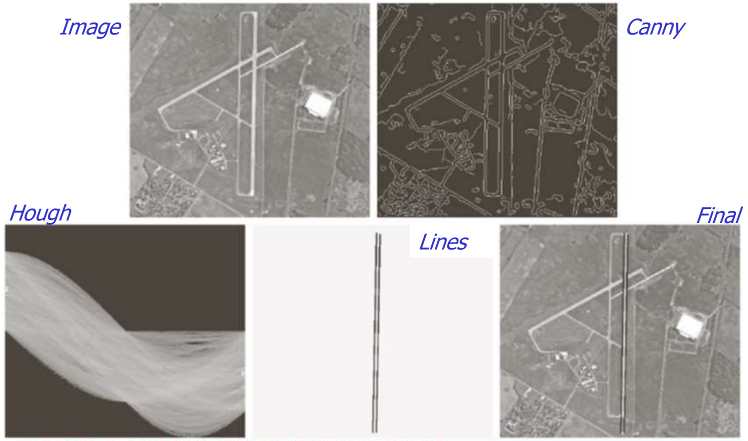
Image

Canny

Hough

Lines

Final



Source : Gonzalez and Woods. Digital Image Processing. Prentice-Hall 3ed, 2008.

Ransac

CS231: Ransac, Juan Carlos Niebles and Ranjay Krishna, Stanford Vision and Learning Lab

Ransac

- A model fitting method:
  - A learning technique to estimate parameters of a model by random sampling of observed data
  - Used for :
    - Line detection
    - Correspondance problem (matching between 2 sets of features)
    - …

Example: Line Fitting

- Why fit lines?
  - Many objects characterized by presence of straight lines



Slide credit: Kristen Grauman

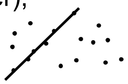
Fitting as search in parametric space

- Choose a parametric model to represent a set of features
- Membership criterion is not local
  - Can't tell whether a point belongs to a given model just by looking at that point.
- Three main questions:
  - What model represents this set of features best?
  - Which of several model instances gets which feature?
  - How many model instances are there?
- Computational complexity is important
  - It is infeasible to examine every possible set of parameters and every possible combination of features

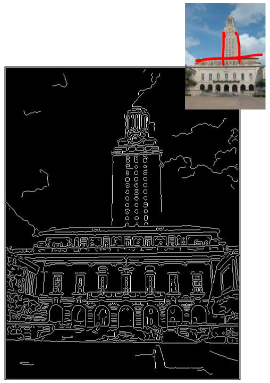
Source: L. Lazebnik

Difficulty of Line Fitting

- Extra edge points (clutter), multiple models:
  - Which points go with which line, if any?



- Only some parts of each line detected, and some parts are missing:
  - How to find a line that bridges missing evidence?
- Noise in measured edge points, orientations:
  - How to detect true underlying parameters?



Slide credit: Kristen Grauman

Voting

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- Voting is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, *but* typically their votes should be inconsistent with the majority of "good" features.
- Ok if some features not observed, as model can span multiple fragments.

Slide credit: Kristen Grauman

RANSAC [Fischler & Bolles 1981]

- RANdom SAmple Consensus
- Approach: we want to avoid the impact of outliers, so let's look for "inliers", and use only those
- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won't have much support from rest of the points.

Slide credit: Kristen Grauman

RANSAC [Fischler & Bolles 1981]

RANSAC loop:

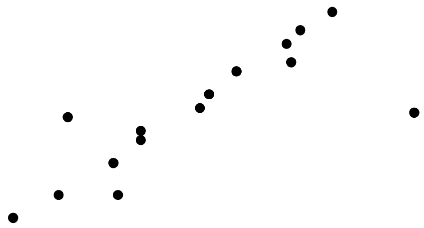
1. Randomly *select a seed group* of points on which to base transformation estimate (e.g., a group of matches)
2. Compute transformation from seed group
3. Find *inliers* to this transformation
4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

Slide credit: Kristen Grauman

RANSAC Line Fitting Example

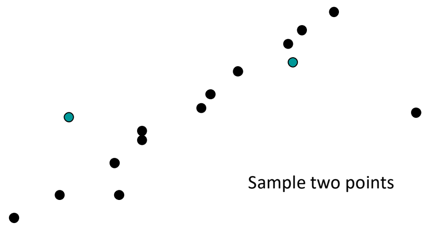
- Task: Estimate the best line
  - *How many points do we need to estimate the line?*



Slide credit: Kristen Grauman

RANSAC Line Fitting Example

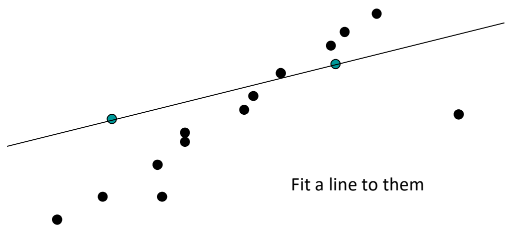
- Task: Estimate the best line



Slide credit: Kristen Grauman

RANSAC Line Fitting Example

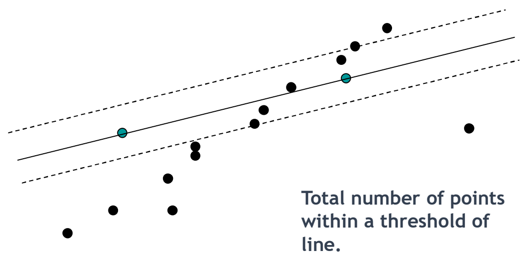
- Task: Estimate the best line



Slide credit: Kristen Grauman

RANSAC Line Fitting Example

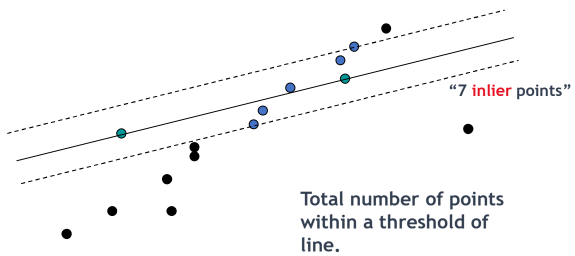
- Task: Estimate the best line



Slide credit: Kristen Grauman

RANSAC Line Fitting Example

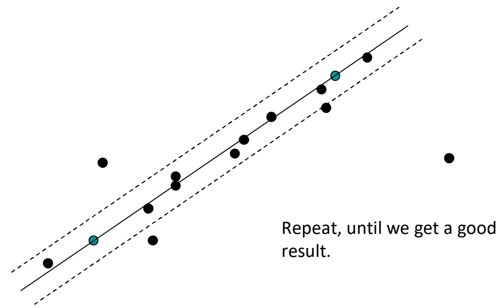
- Task: Estimate the best line



Slide credit: Kristen Grauman

RANSAC Line Fitting Example

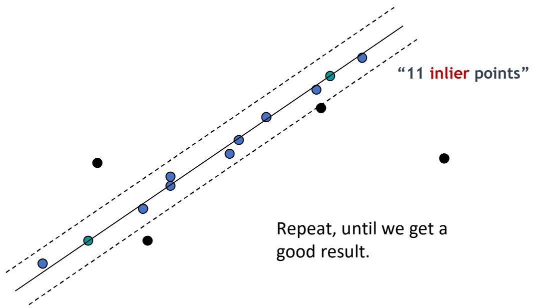
- Task: Estimate the best line



Slide credit: Kristen Grauman

RANSAC Line Fitting Example

- Task: Estimate the best line



Slide credit: Kristen Grauman

Algorithm 15.4: RANSAC: fitting lines using random sample consensus

Determine:
- n — the smallest number of points required
- k — the number of iterations required
- t — the threshold used to identify a point that fits well
- d — the number of nearby points required to assert a model fits well

Until k iterations have occurred
- Draw a sample of n points from the data uniformly and at random
- Fit to that set of n points
- For each data point outside the sample
  - Test the distance from the point to the line against t; if the distance from the point to the line is less than t, the point is close
- end
- If there are d or more points close to the line then there is a good fit. Refit the line using all these points.

end
Use the best fit from this collection, using the fitting error as a criterion

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

RANSAC: How many samples?

- How many samples are needed?
  - Suppose w is fraction of inliers (points from line)
  - n points needed to define hypothesis (2 for lines)
  - k samples chosen
  - p: desired probability that we get a good sample
- Prob. that a single sample of n points is correct: w^n
- Prob. that all k samples fail is: $(1-w^n)^k$

$\Rightarrow$ Choose k high enough to keep this below desired failure rate (1-p).

Slide credit: David Lowe

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

RANSAC: Computed k (p=0.99)

Sample size	Proportion of outliers						
n	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

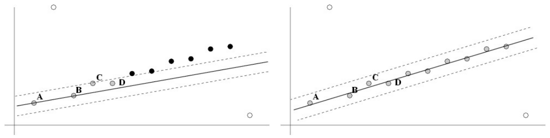
p: desired probability that we get a good sample

Slide credit: David Lowe

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

After RANSAC

- RANSAC divides data into inliers and outliers and yields estimate computed from minimal set of inliers.
- Improve this initial estimate with estimation over all inliers (e.g. with standard least-squares minimization).
- But this may change inliers, so alternate fitting with re-classification as inlier/outlier.



Slide credit: David Lowe

RANSAC: Pros and Cons

- **Pros**:
  - General method suited for a wide range of model fitting problems
  - Easy to implement and easy to calculate its failure rate
- **Cons**:
  - Only handles a moderate percentage of outliers without cost blowing up
  - Many real problems have high rate of outliers (but sometimes selective choice of random subsets can help)
- A voting strategy, the Hough transform can handle high percentage of outliers

Plan

- Edges
  - Detection
  - linking
- Feature extraction (next lesson)
  - Global features
  - Local features
- Image Matching and Applications (next lesson)

References

- Lecture 3: CS231 - Juan Carlos Niebles and Ranjay Krishna, Stanford Vision and Learning Lab
- Vision par Ordinateur, Alain Boucher, IFI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict