

Image Processing & Pattern Recognition

Introduction to CP467

Dr. Zia Ud Din

Outline

- Course Logistics
- Overview of Image Processing

Course Staff: Instructor

□ Instructor

■ Dr. Zia Ud Din (zuddin@wlu.ca)

■ Education

- Bachelors in Civil Engineering
- Masters in Computer Science
- Ph.D. in Computer Science

■ Research Interests

- Computer Vision
- Image Processing
- Machine Learning

The Course

- Assumes some basic mathematical maturity and some intermediate programming skills
- We will use **Python** programming language
- We will use **OpenCV** library
- You are free to use any IDE (or none)
- Let's go through the **course syllabus** for more details

Textbook (recommended, not required)

- Digital Image Processing by Rafael Gonzalez and Richard Woods, 4th Ed. (2018), Pearson.

Course Topics and Weekly Schedule

Week #	Topic
1	Introduction and Digital Image Fundamentals
2	Intensity Transformations and Histogram Processing
3	Filtering
4	Edge Detection
5	Geometric Primitive Extraction
6	Feature Detection
7	Feature Description and Matching
8	Image Segmentation
9	Clustering
10	Machine Learning and Image Classification
11	Presentations
12	Presentations; Final Exam Review

Assessment and Grading

Assessment	Weighting
Assignments	25%
Project	20%
Presentation	10%
Final Exam	45%
Total	100%

Assessment	Due
Assignment 1	Week 3
Assignment 2	Week 4
Assignment 3	Week 5
Assignment 4	Week 6
Assignment 5	Week 7
Project	Week 11
Presentation	Week 11 and 12

Read the syllabus on MLS to know more about the assessments and other aspects of the course

Assignments

- Individual
- 5 in total
- Best 4 will be counted
- Have equal weight

Project

- Programming, research and writing
- Group (group size=3)

Presentation

- Research paper presentation
- Group (group size=3)
- 20 minutes per group
- A list of papers will be provided

Late Assignment & Project Submission Policy

- 20% deductions if late up to 1 day
- No make-up for missed assignments/project and no due date extensions

Academic Integrity

- Cheating is unpleasant for everyone
- Discuss ideas with others but do not see/show solutions
 - Even incomplete and/or incorrect ones
- Do not use internet to post questions or find solutions
- If found involved in cheating, you will be reported to Dean's office
- Please read the relevant policy at Laurier's academic integrity website before submitting your first work

How to Get Good Grades?

1. Attend classes regularly.
2. Read the lecture slides before the class.
3. Revise each lecture within 24 hours.
 - Retention of information has a high correlation with early and frequent revisions.
4. Do not miss a deadline.
5. Ask if something is not clear.

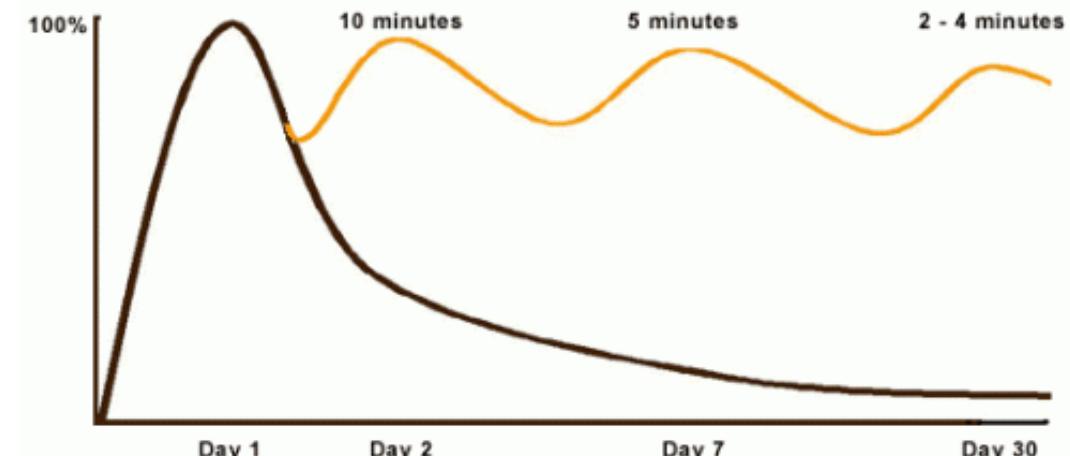


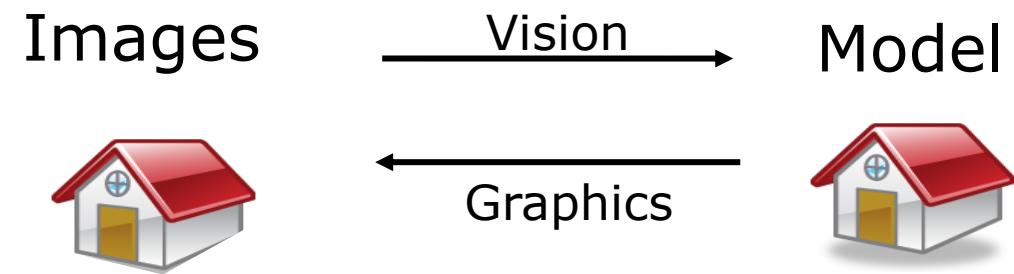
Image source: <https://uwaterloo.ca/campus-wellness/curve-forgetting>

Outline

- Course Logistics
- Overview of Image Processing

Related Disciplines

- Image Processing
- Pattern Recognition
- Computer Vision
- Computer Graphics
- Artificial Intelligence
- Machine Learning



Why Image Processing?

Images and video are everywhere!



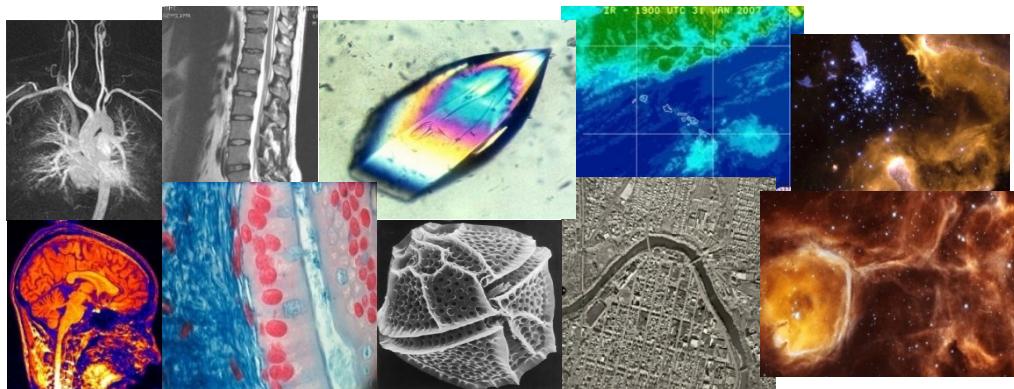
Personal photo albums



Movies, news, sports



Surveillance and security

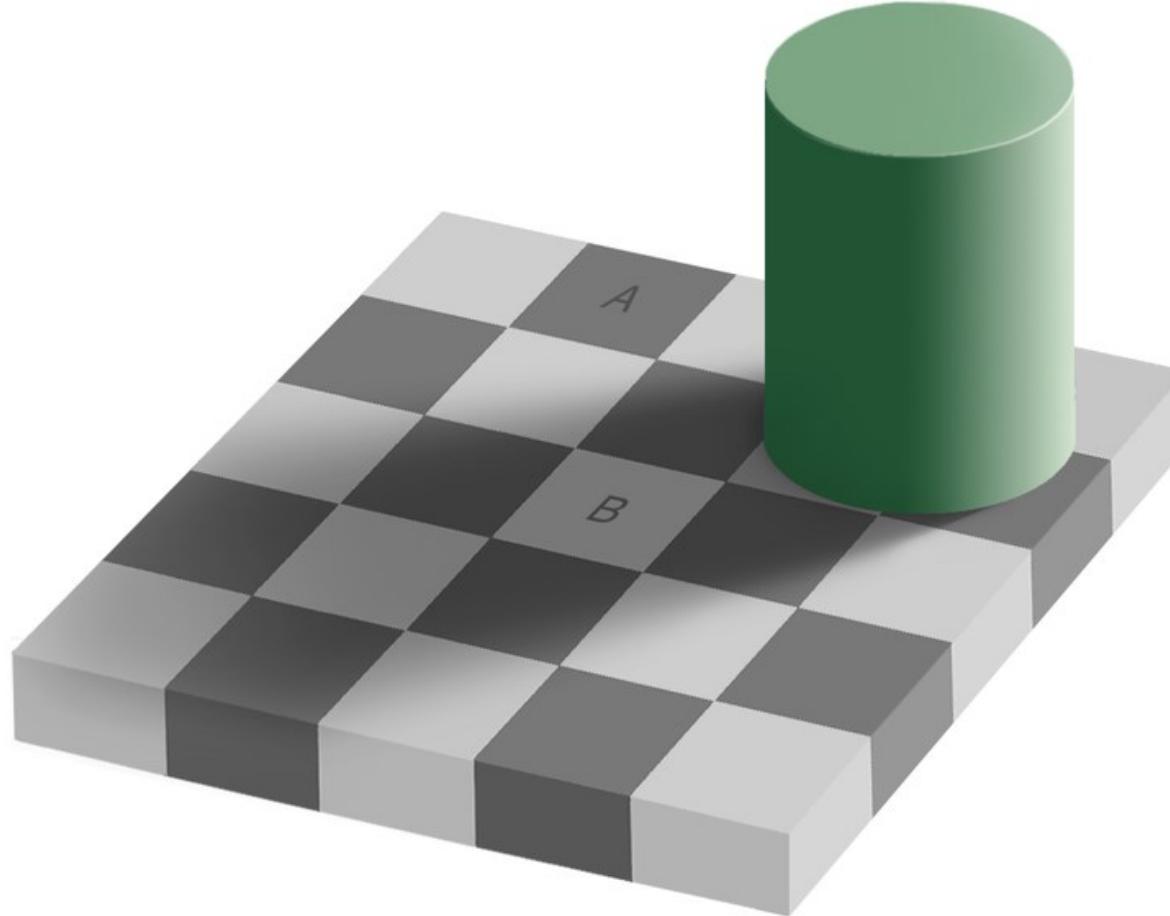


Medical and scientific images

Can Computers Match (or Beat) Human Vision?

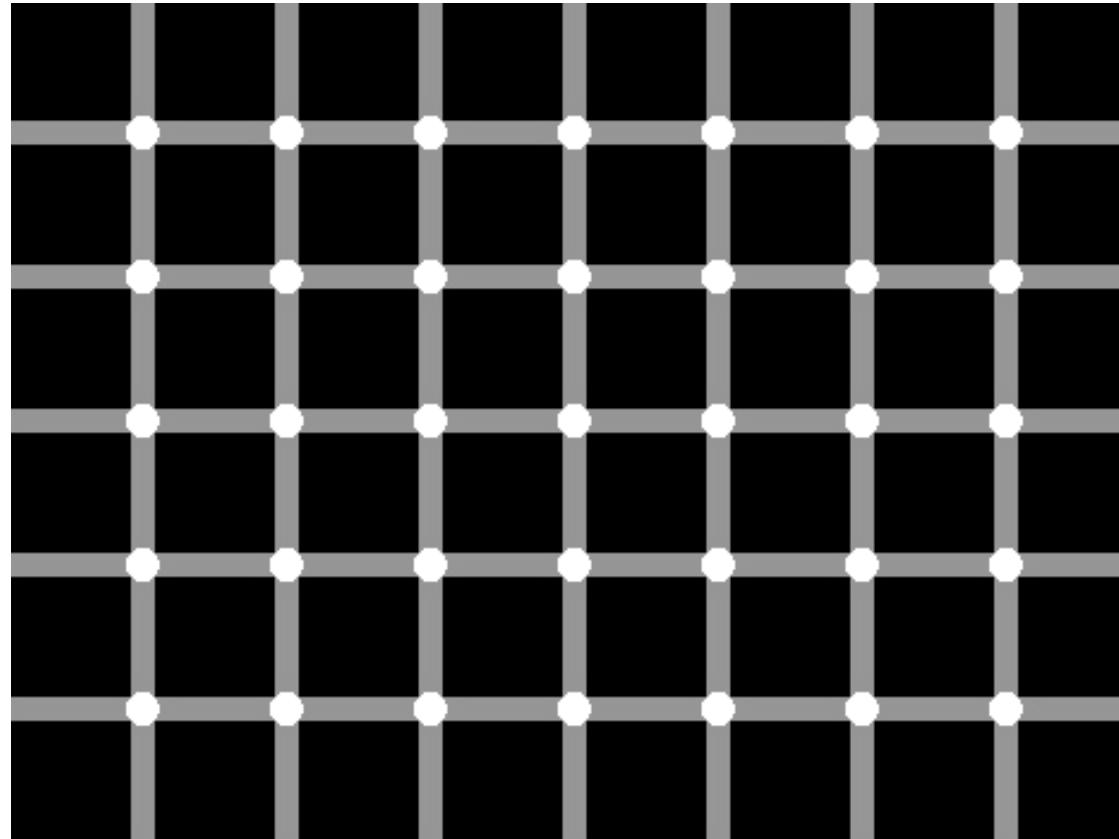
- Until the last decade, the answer was mostly no!
- Recent developments in deep learning have enabled computer to excel in areas such as Image Classification, Object Detection and Tracking, Medical Imaging, Large Scale Visual Search etc.
- Still humans have certain edges such as a better "hardware", integration with other senses, learning and adaptability etc.

Human Perception Has its Shortcomings...



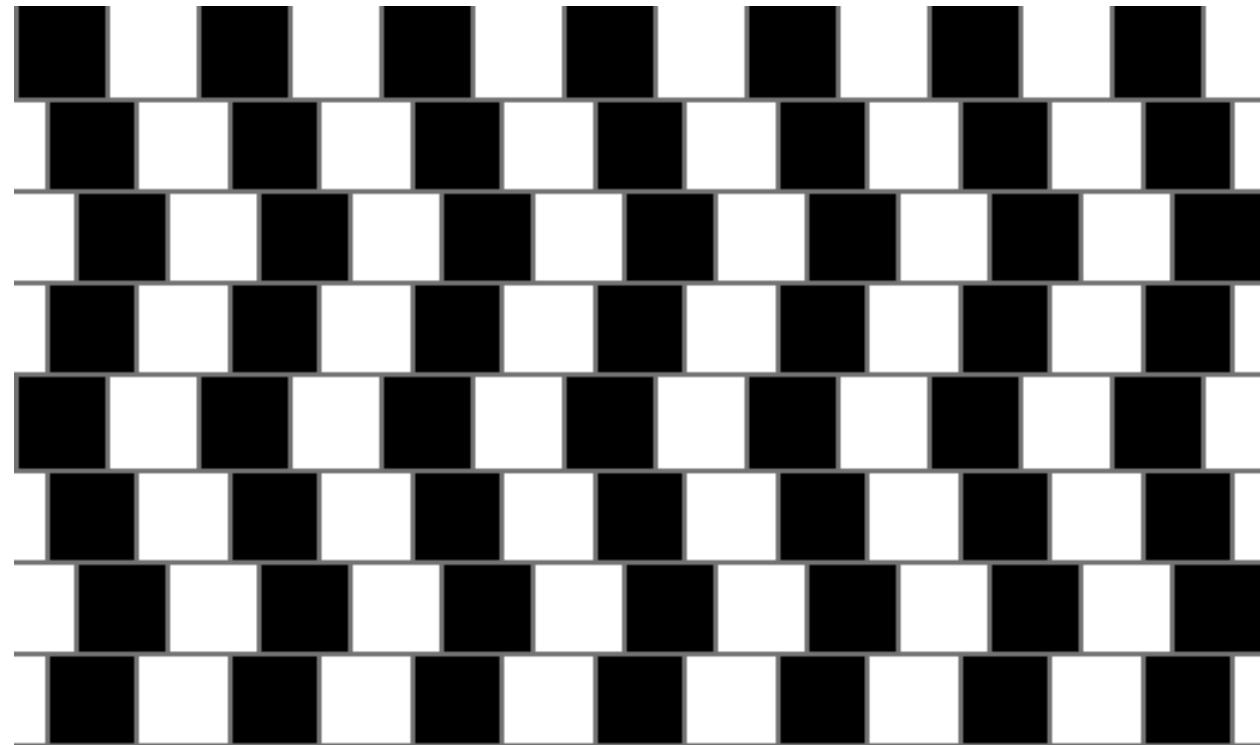
Which block is darker: A or B?

Human Perception Has its Shortcomings...



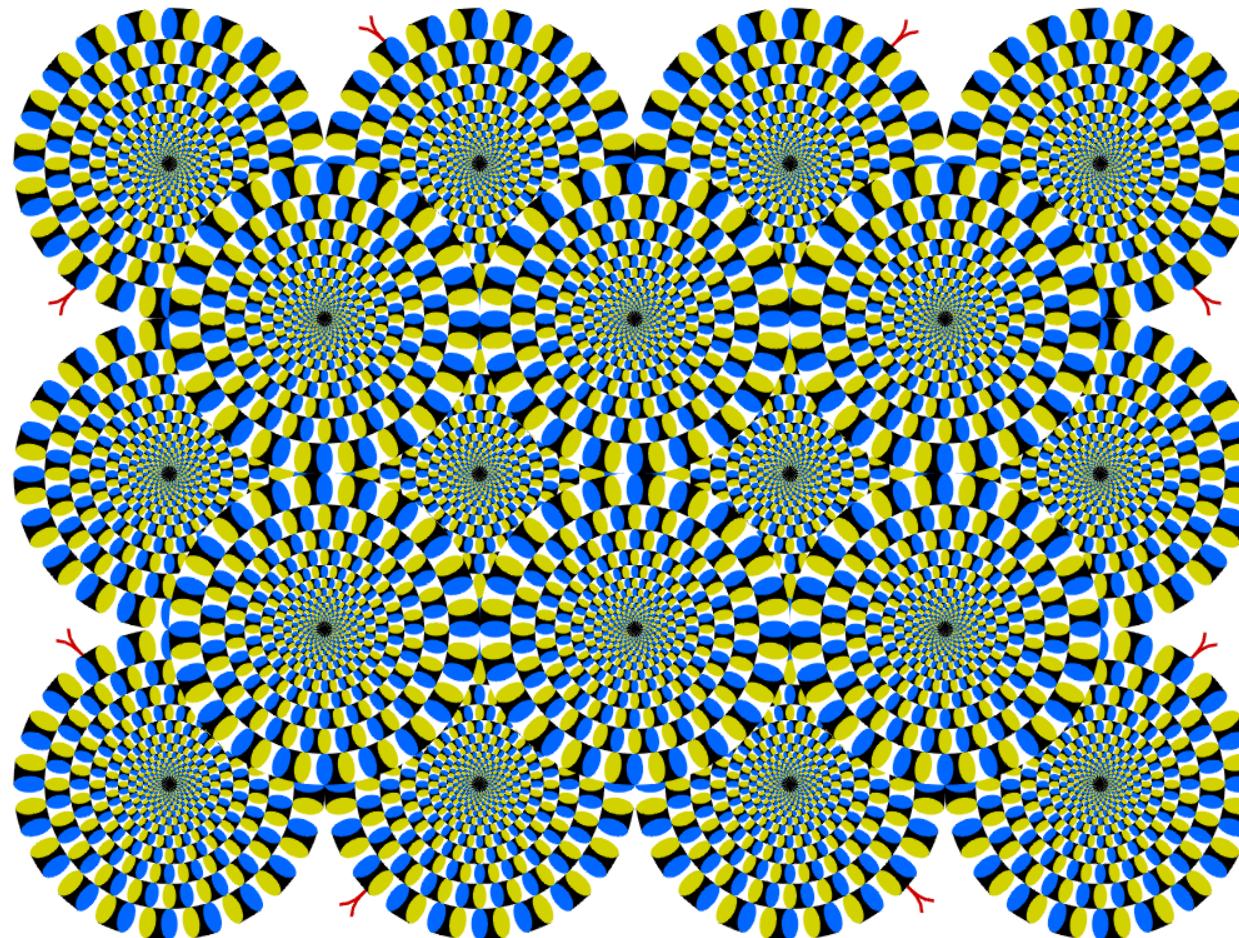
How many black dots are there?

Human Perception Has its Shortcomings...



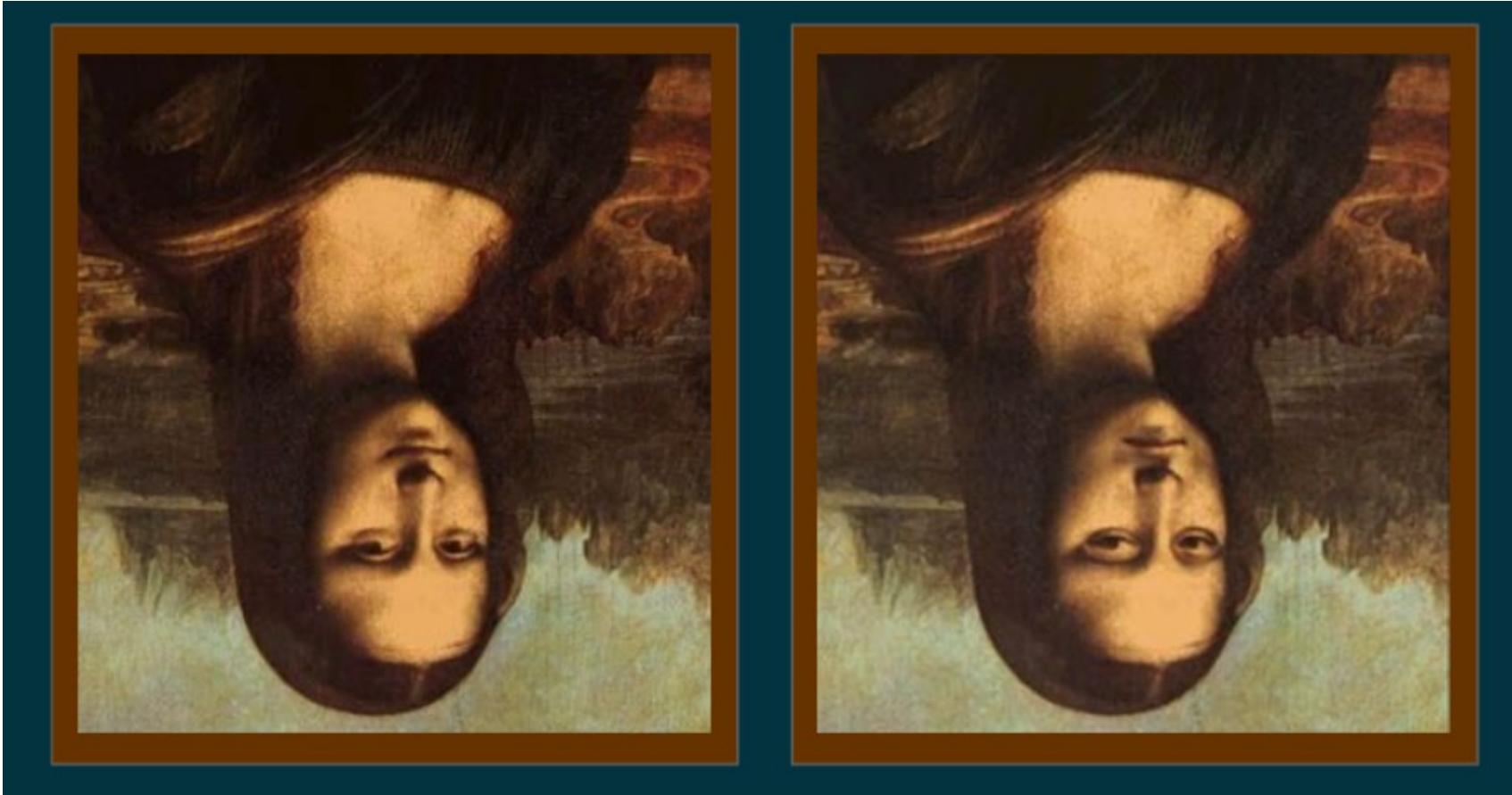
Are the lines straight or are they curved?

Human Perception Has its Shortcomings...



How many circles are moving?

Human Perception Has its Shortcomings...



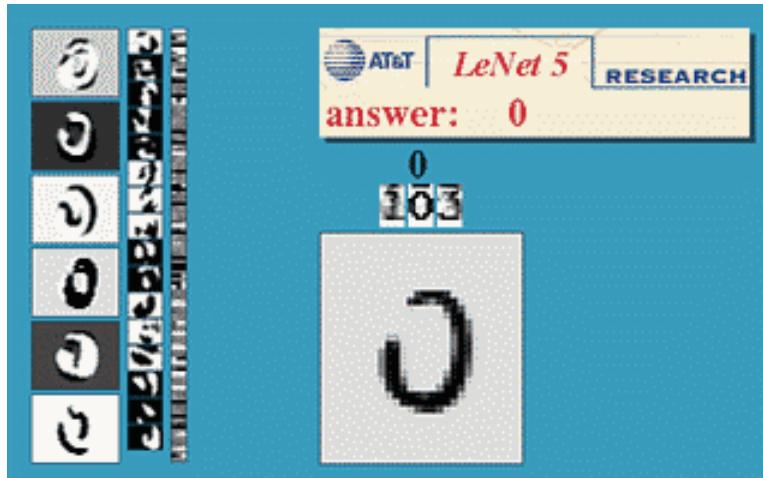
Which Mona Lisa is the correct one?

Current Applications of Image Processing

- The next slides show some examples of what current image processing systems can do

Optical Character Recognition (OCR)

- OCR is perhaps the most commercially prolific application of Image Processing research



Digit recognition, AT&T labs
<http://yann.lecun.com/exdb/lenet/>



Automatic check processing



License plate readers
http://en.wikipedia.org/wiki/Automatic_number_plate_recognition

ANPR: Automatic Number Plate Recognition

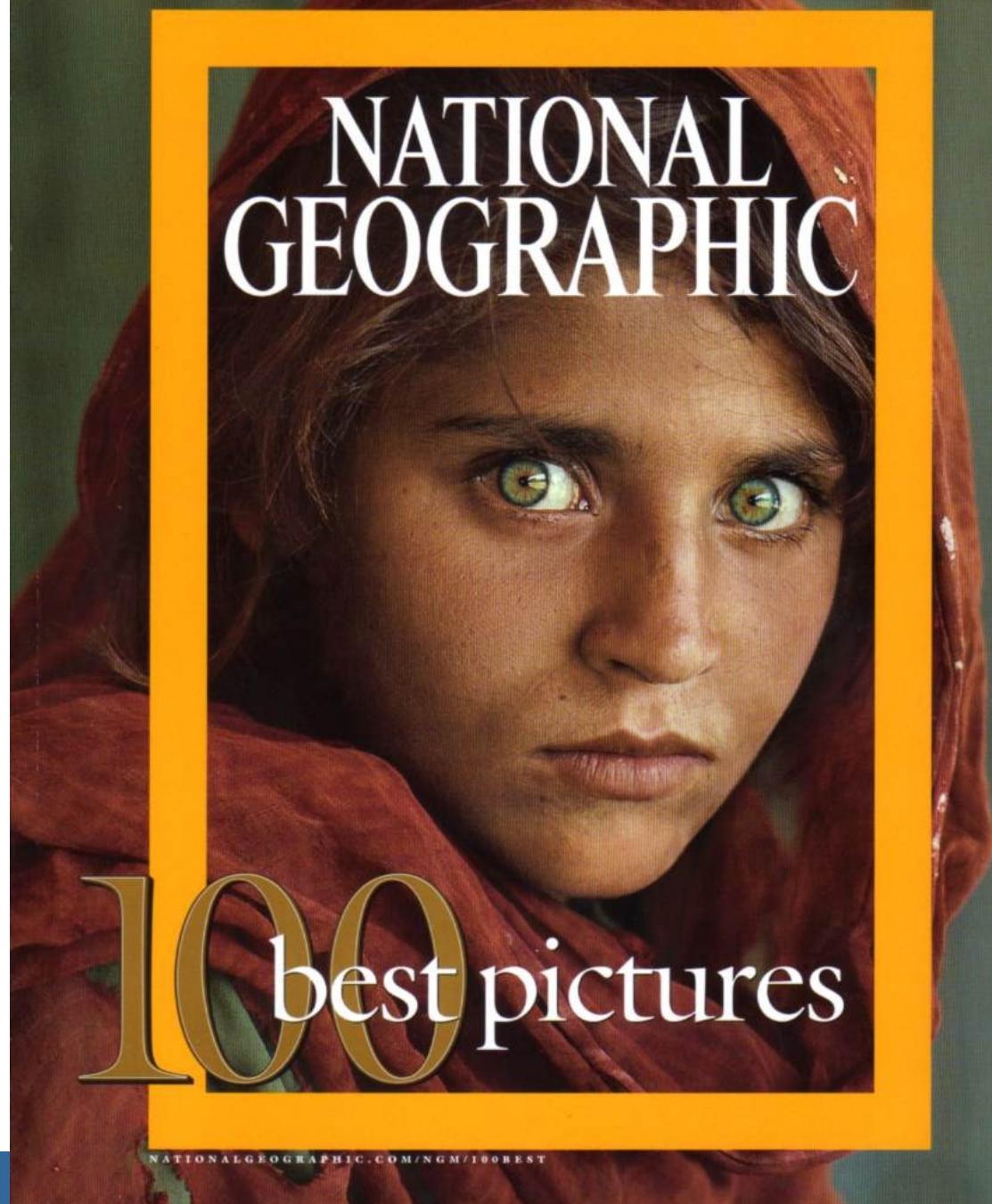
- Plate localisation - responsible for finding and isolating the plate on the picture
- Plate orientation and sizing - compensates for the skew of the plate and adjusts the dimensions to the required size
- Normalisation - adjusts the brightness and contrast of the image
- Character segmentation - finds the individual characters on the plates
- Optical character recognition
- Syntactical/Geometrical analysis - check characters and positions against country specific rules



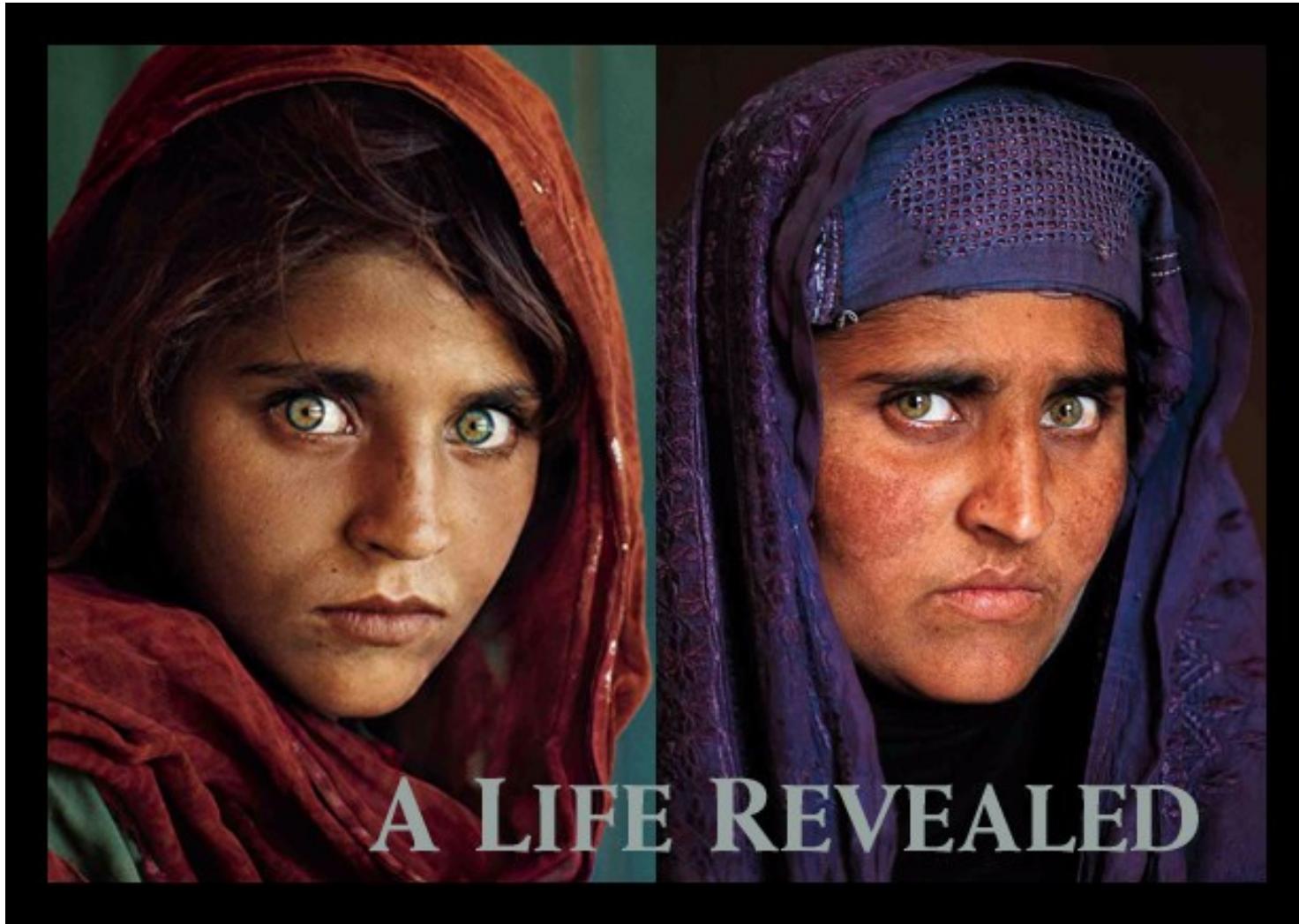
Biometric Recognition



SPECIAL MEMBER'S EDITION VOL.



Biometric Recognition (cont.)



Biometric Recognition (cont.)



Photograph by Alexandra Boulat

<https://www.nationalgeographic.com/pages/article/afghan-girl-home-afghanistan>

Movies: Special Effects



Video



Video

Object Recognition in Mobile Phones



Login without a Password



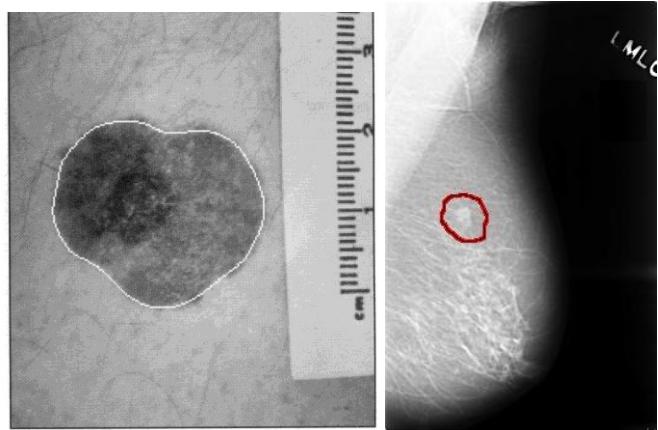
FastAccess™
FACIAL RECOGNITION

Use your **face** to login
to your **computer**
and **websites!**

A woman is using a laptop with facial recognition software. The screen displays a grid over her face and the text "ANALYZING FACE".

Medical Imaging

Skin/Breast Cancer Detection



3D imaging
MRI, CT

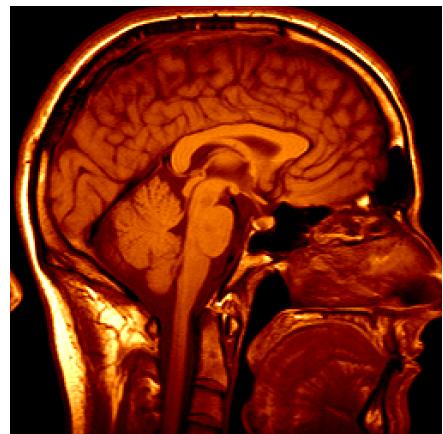
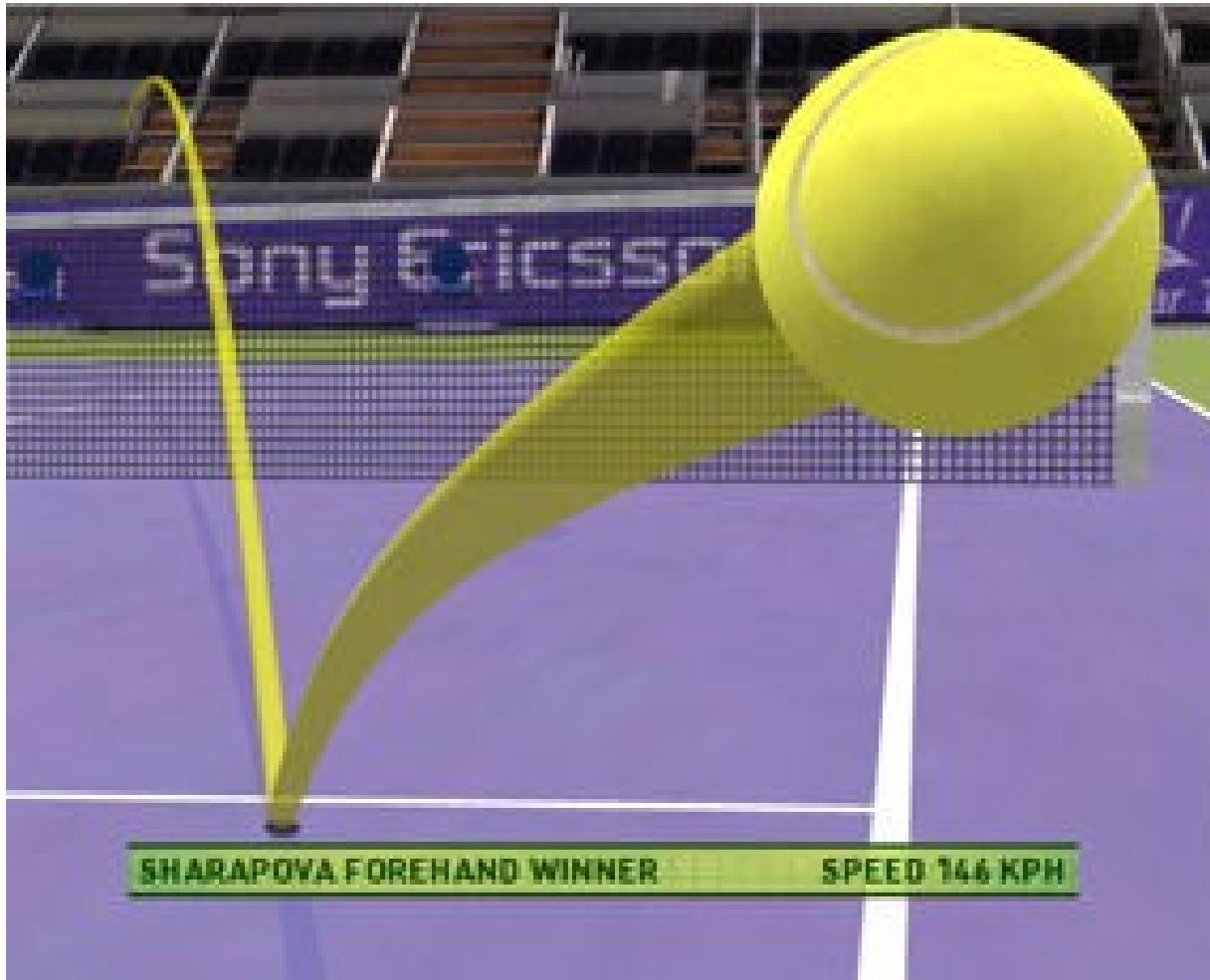


Image guided surgery
Grimson et al., MIT



Enable surgeons to visualize internal structures through an automated overlay of 3D reconstructions of internal anatomy on top of live video views of a patient.

Sports: Hawk-Eye



Sports: Hawk-Eye



3D Urban Modeling - Google Street View

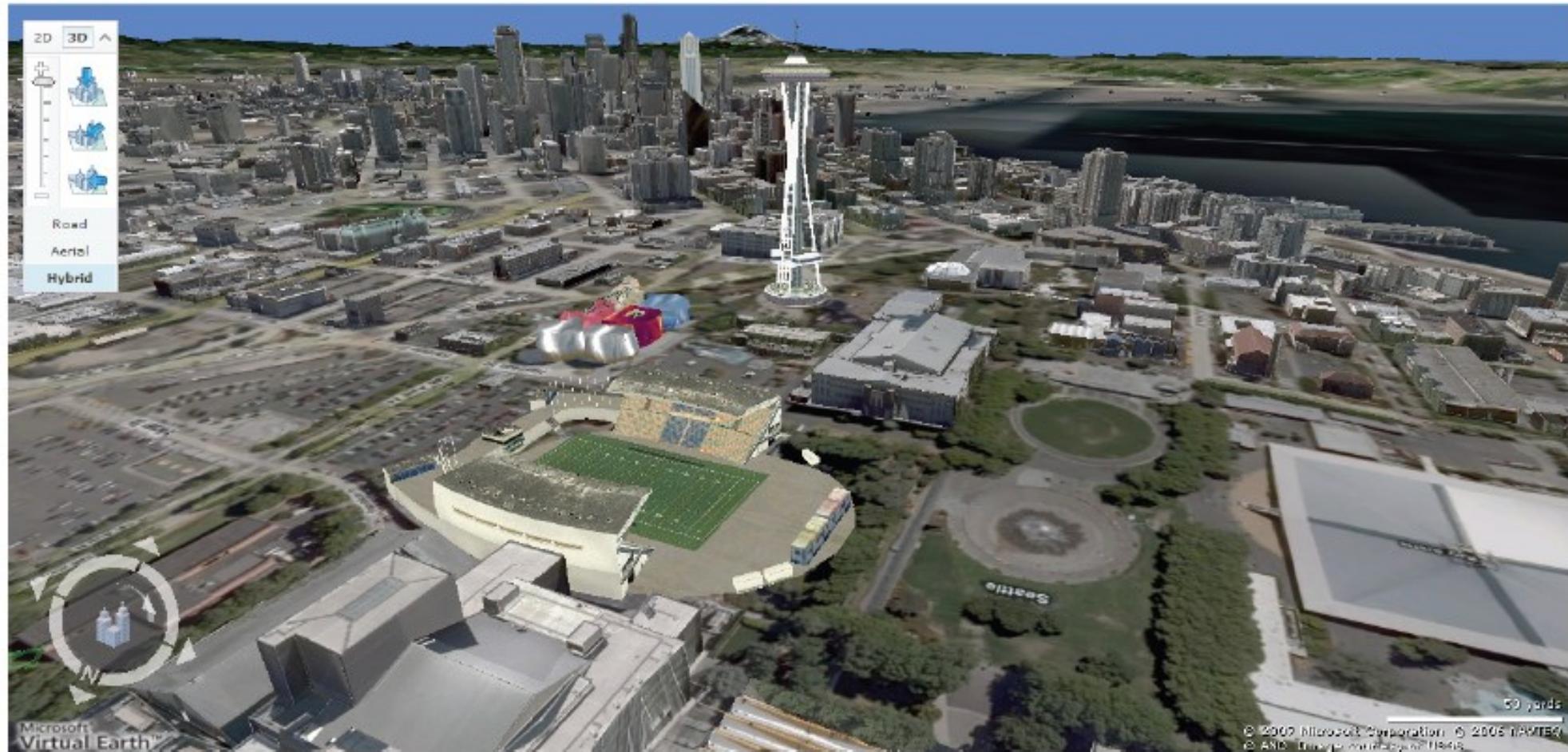


Image Processing in Supermarkets



Robotic Bin Picking



http://robohub.org/team-rbo-from-berlin-wins-amazon-picking-challenge-convincingly/amazon_pick_banner_robot-2/

<https://www.tudelft.nl/en/2016/tu-delft/team-delft-wins-amazon-picking-challenge/>

Self-Driving Cars

L. Lane Radius: 4.74km
R. Lane Radius: 1.09km
C. Position: -0.40m
Close Vehicles: 2



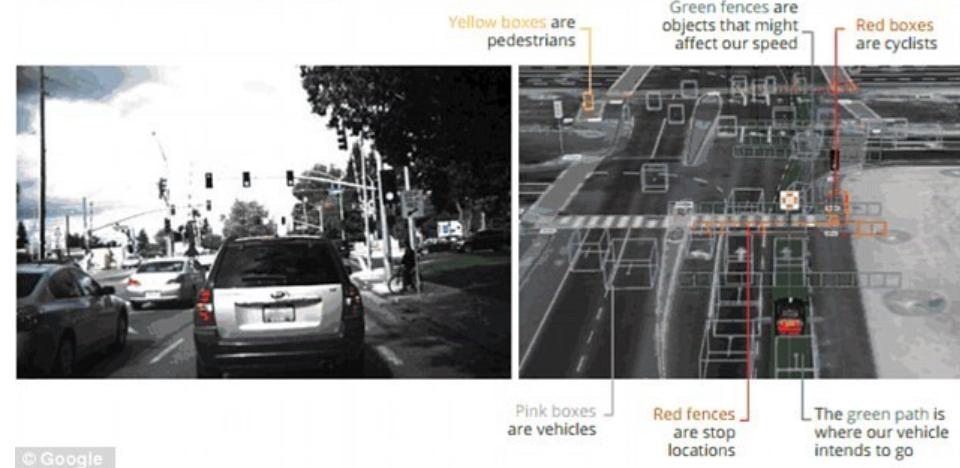
<https://medium.com/@ricardo.zuccolo/self-driving-cars-opencv-and-svm-machine-learning-with-scikit-learn-for-vehicle-detection-on-the-bf88860e055a>



<https://www.mobileye.com/our-technology/>



A self-driving car's view of the world



<http://www.dailymail.co.uk/sciencetech/article-2641300/Google-launches-25-mph-driverless-car-fitted-ONLY-stop-button.html>

References and Credits

- Lecture Slides by Prof. Trevor Darrell
- <http://www.brainbashers.com>
- <http://www.echalk.co.uk>
- <http://www.sensiblevision.com/en-us/home.aspx>
- www.moving-picture.com

Image Processing & Pattern Recognition

Digital Image Fundamentals

Dr. Zia Ud Din

Outline

- Digital Image
- Image Sampling and Quantization
- Image Interpolation
- Relationships between Pixels
- Mathematical Tools used in DIP

Image Processing Basics

- **Image**
 - A 2-D function $f(x, y)$
 - x and y are spatial coordinates,
 - Amplitude of f at (x, y) is called the intensity or gray level at that point
- **Digital Image**
 - x, y , and f are all finite, discrete quantities
- **Digital Image Processing**
 - Processing digital images by digital computer
- **Pixel**
 - Each spatial location (x, y)
- **Can images be analog?**
 - Yes
- **What is an analog image?**
 - x and y are real numbers in contrast to digital image where x and y are integers

Image Processing Basics

- How digital images are represented?
 - Binary
 - Grayscale
 - Color
- How to increase brightness of a digital image?
 - Just add some constant in the grayscale value at each pixel
- How to increase contrast of a digital image?
 - Just multiply the grayscale value at each pixel by some constant (>1)
- Spatial resolution
 - The precision of a measurement with respect to space
- Grayscale resolution
 - The number of shades of gray utilized in preparing the image for display

Image Processing Basics

- Typical grayscale resolution is 8 bits. Why?
 - Avg. human eye can perceive < 100 so 256 gray values are sufficient
- How to represent RGB images?
 - 3 channels one each for R, G and B (typically 8-bit each)
- Data storage size depends on resolution (spatial and grayscale)
 - $512 * 512$ 8-bit = 256 KB
 - $1024 * 1024$ 8-bit = 1 MB
 - $2048 * 2048$ 24-bit = 12 MB
 - But then why images on your computer show much smaller size?
 - They are compressed

Electromagnetic (EM) Spectrum

- EM spectrum can be expressed as:
 - Wavelength (λ) (unit: m)
 - Frequency (ν) (unit: hertz)
 - Energy (E) (unit: electron volt)
- Relations
 - $\lambda = c/\nu$ (c is speed of light)
 - $E = h\nu$ (h is Planck constant)

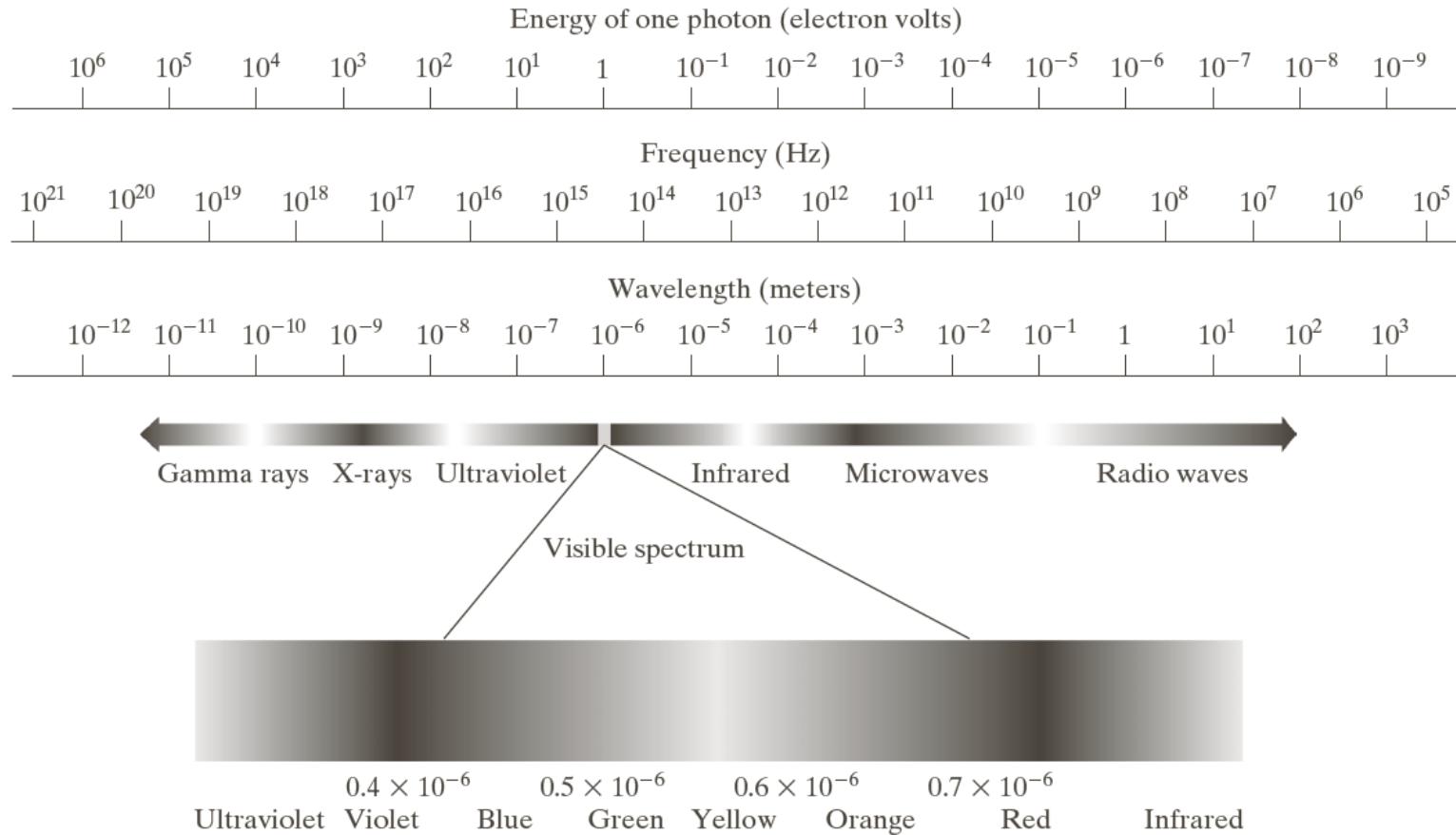


Image source: Gonzalez and Woods

Simple Image Formation Model

$$f(x, y) = i(x, y)r(x, y)$$

$$0 \leq i(x, y) < \infty$$

$$0 \leq r(x, y) \leq 1$$

where i : illumination, r : reflectance

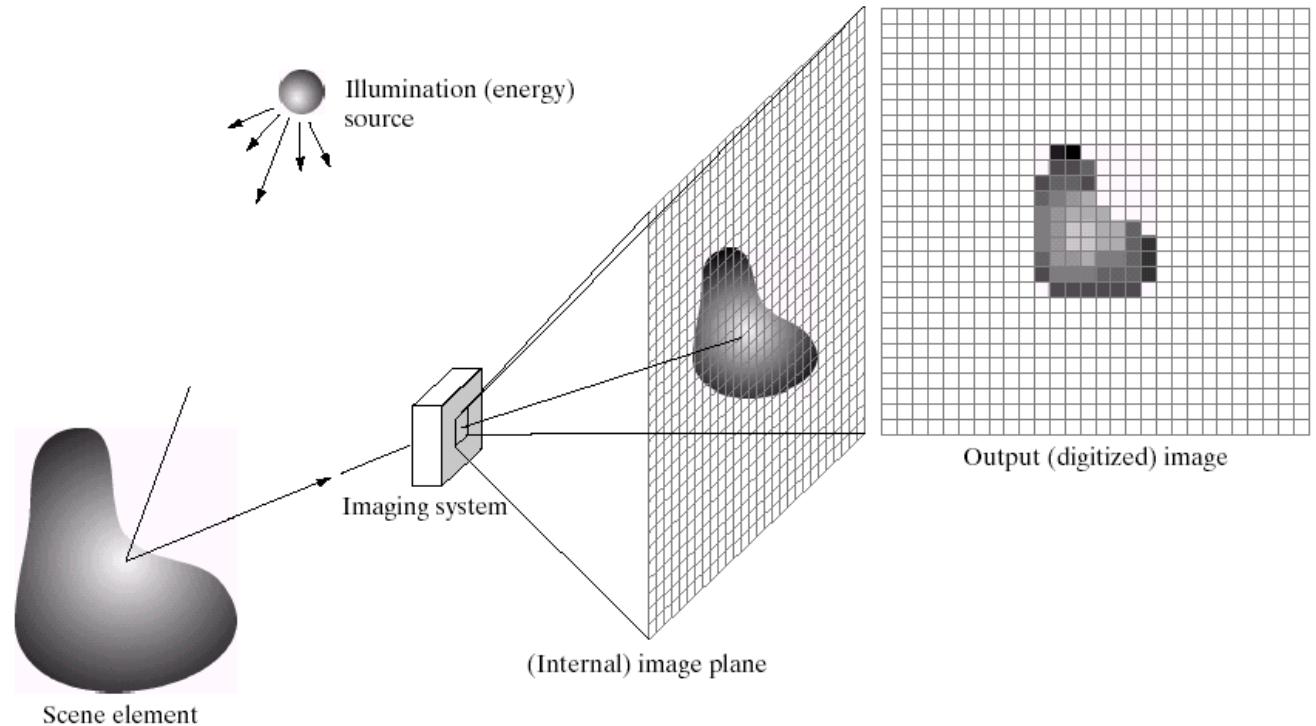


Image source: Gonzalez and Woods

Simple Image Formation Model...

- On a clear day, the sun may produce in excess of 90,000 lm/m^2 of illumination on the surface of the earth.
- This value decreases to less than 10,000 lm/m^2 on a cloudy day.
- The typical illumination level in a commercial office is about 1,000 lm/m^2 .
- Similarly, the following are typical values of $r(x, y)$: 0.01 for black velvet, 0.65 for stainless steel, 0.80 for flat-white wall paint, 0.90 for silver-plated metal, and 0.93 for snow.

Simple Image Formation Model...

- Let the intensity (gray level) of a monochrome image at any coordinates (x, y) be denoted by $\ell = f(x, y)$

$$L_{\min} \leq \ell \leq L_{\max}$$

$$L_{\min} = i_{\min}r_{\min} \text{ and } L_{\max} = i_{\max}r_{\max}.$$

- For a typical commercial office, $L_{\min} \approx 10$ and $L_{\max} \approx 1000$
- The interval $[L_{\min}, L_{\max}]$ is called the intensity (or gray) scale.
 - Common practice is to shift this interval numerically to the interval $[0, 1]$, or $[0, C]$, where 0 is considered black and C is considered white on the scale.
 - Most common value of C is 255.
 - All intermediate values are shades of gray varying from black to white.

Outline

- Digital Image
- **Image Sampling and Quantization**
- Image Interpolation
- Relationships between Pixels
- Mathematical Tools used in DIP

Sampling and Quantization

- Output of most sensors is a continuous voltage waveform
- To create a digital image, we need to convert it into a digital format
- Sampling
 - Digitizing coordinates values
- Quantization
 - Digitizing amplitude values

Sampling and Quantization

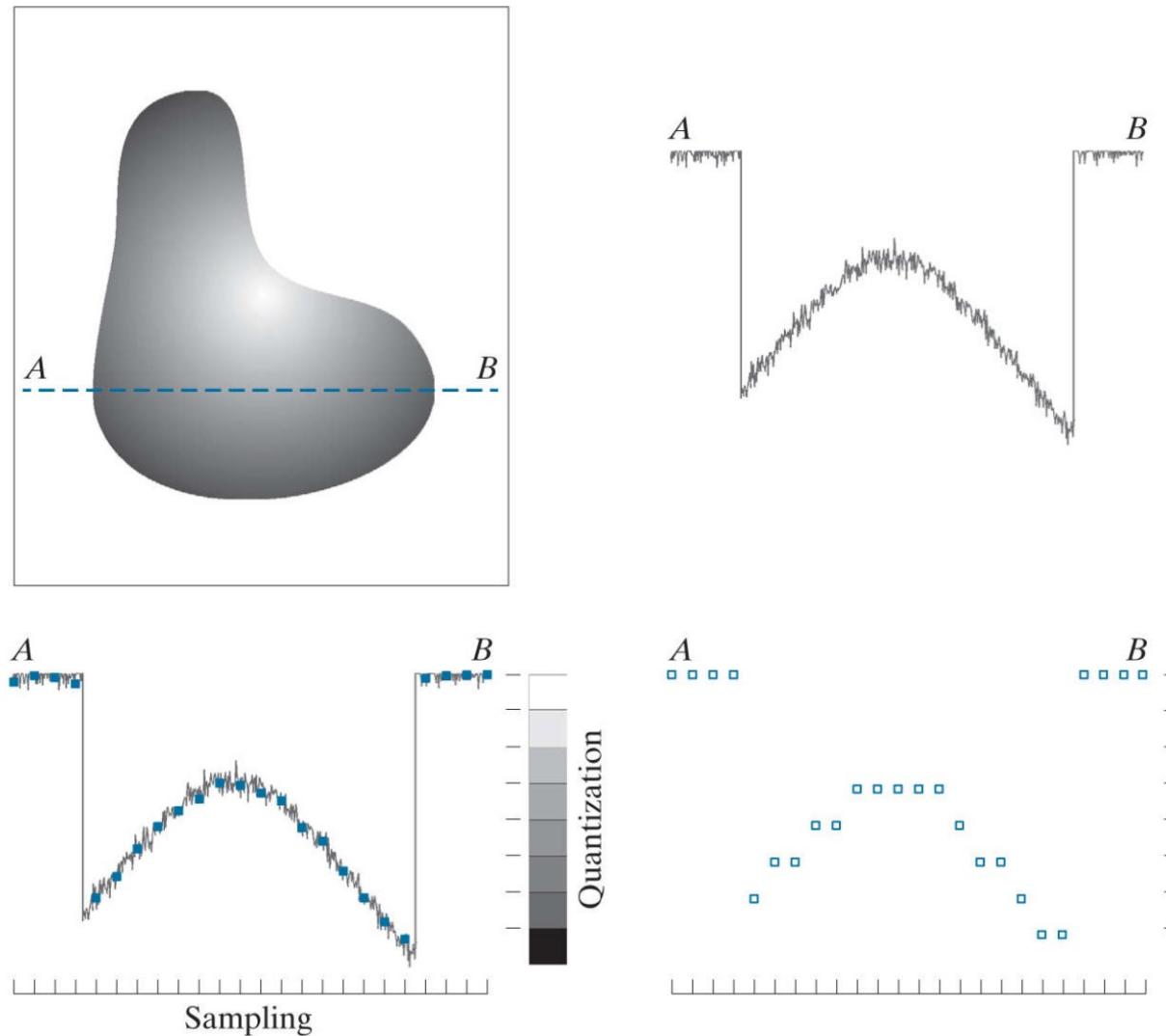


Image source: Gonzalez and Woods

Sampling and Quantization

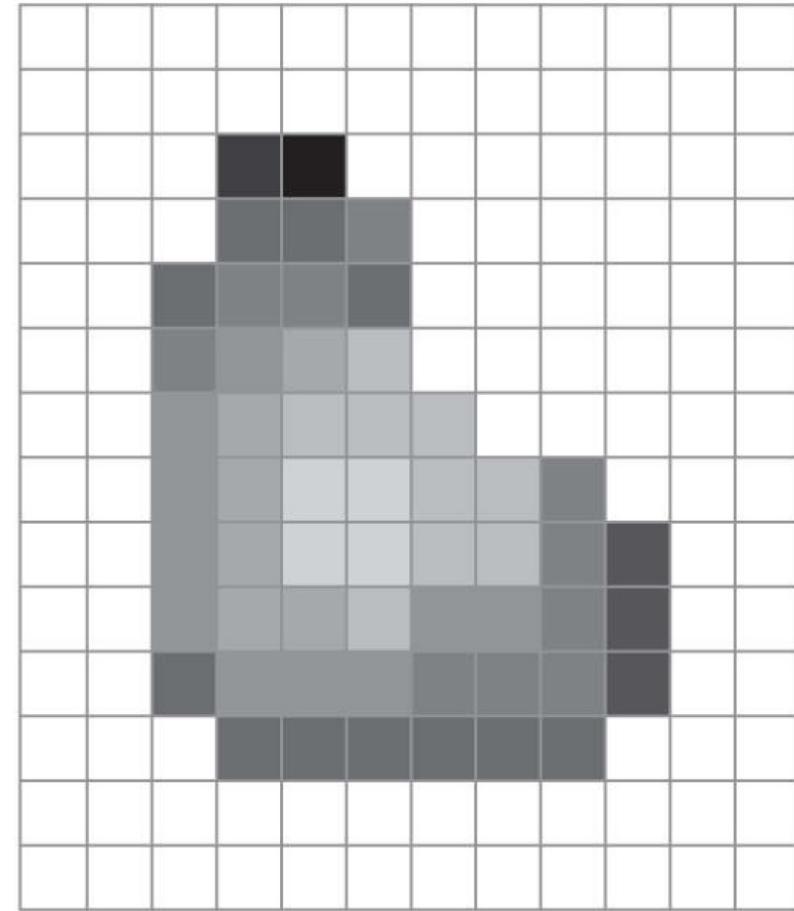
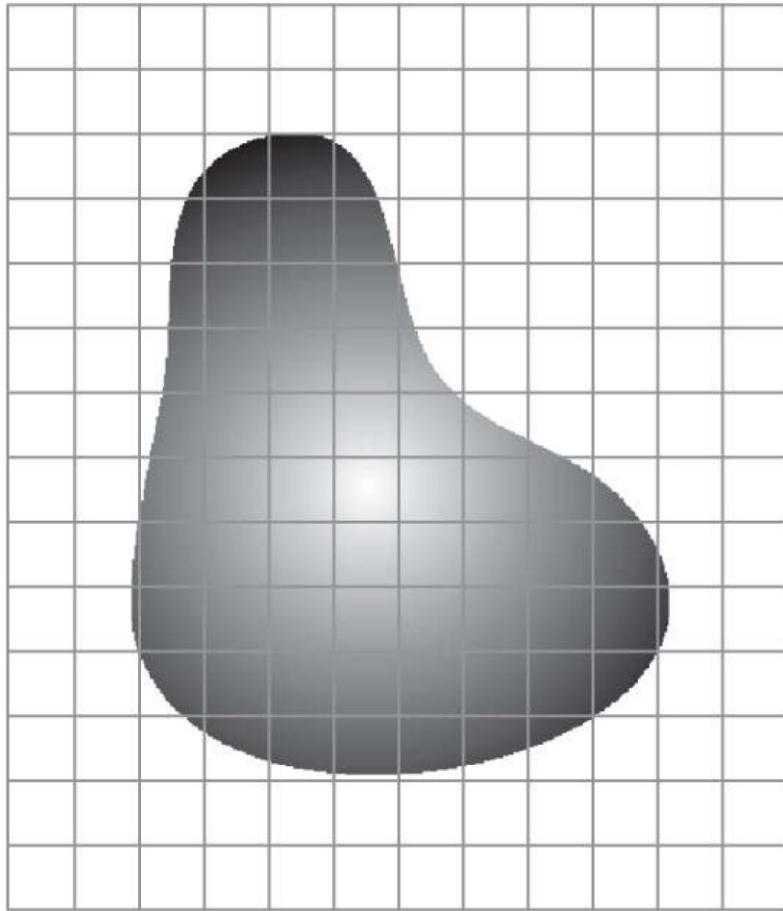


Image source: Gonzalez and Woods

Digital Image Conventions

Coordinate convention used to represent digital images. Because coordinate values are integers, there is a one-to-one correspondence between x and y and the rows (r) and columns (c) of a matrix.

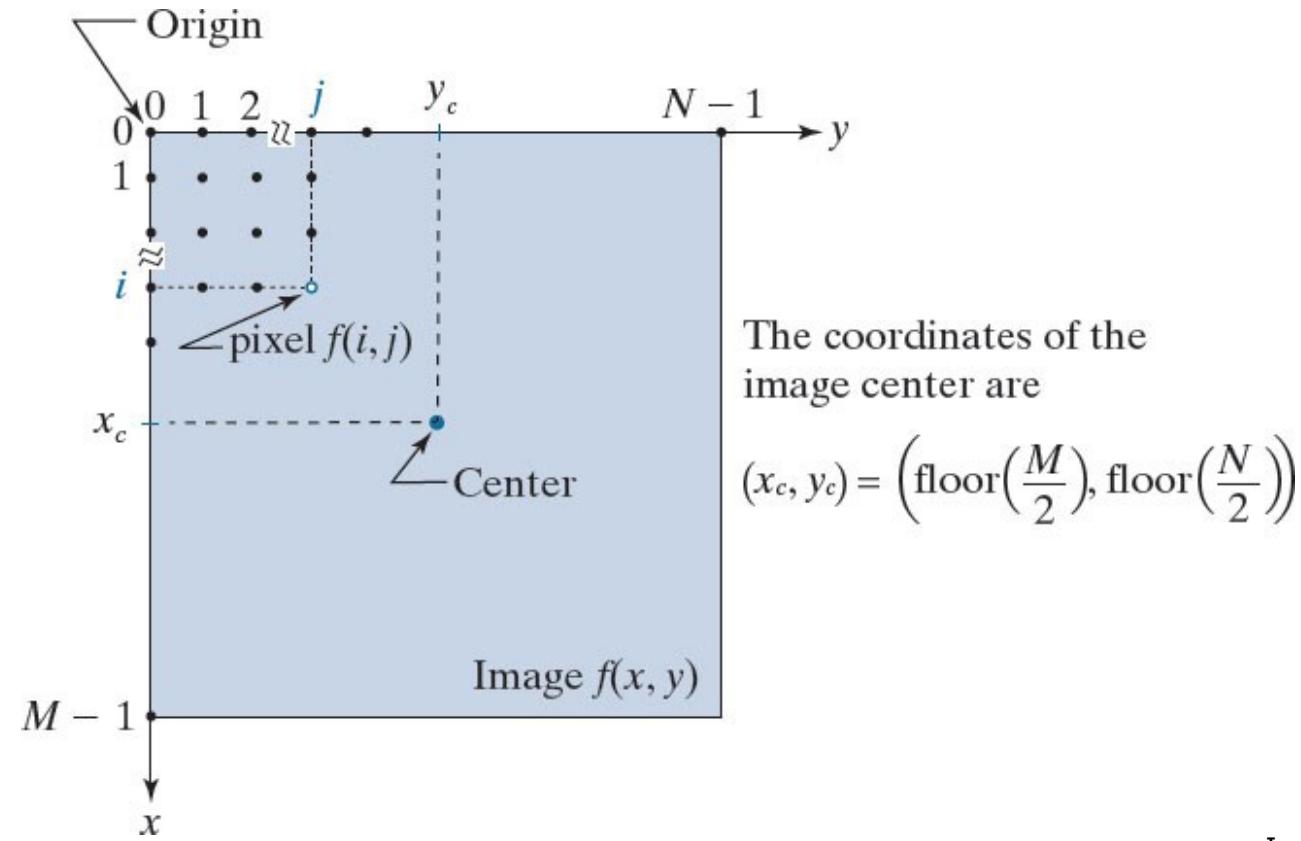


Image source: Gonzalez and Woods

Effect of Reducing Spatial Resolution

Effects of reducing spatial resolution. The images shown are at: (a) 930 dpi, (b) 300 dpi, (c) 150 dpi, and (d) 72 dpi.

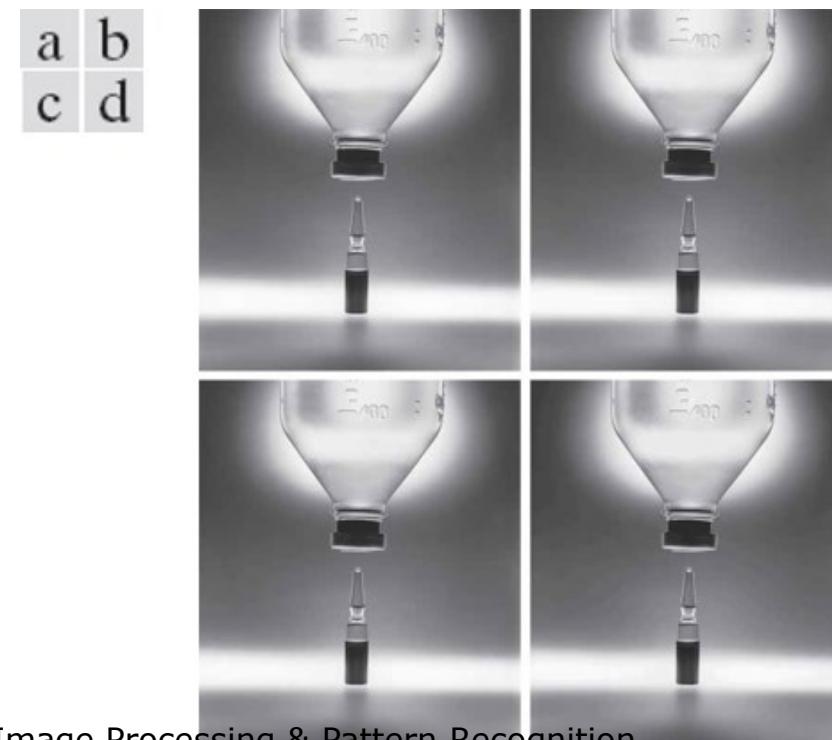
a
b
c
d



Image source: Gonzalez and Woods

Effect of Reducing Gray Level Resolution

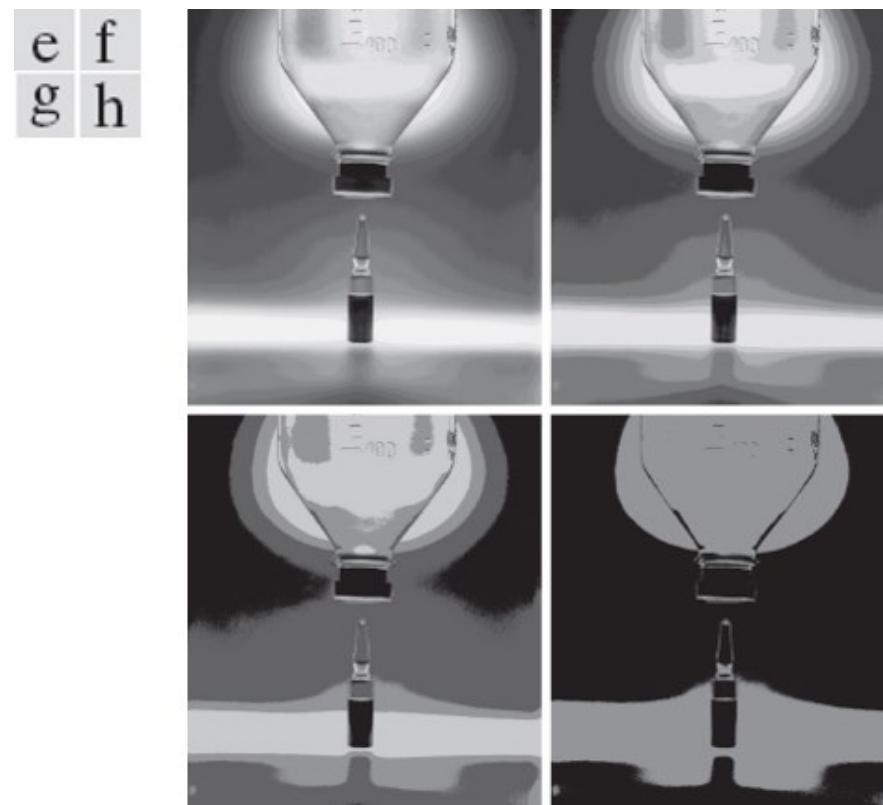
(a) 2022×1800 , 256-level image. (b)-(d) Image displayed in 128, 64, and 32 intensity levels, while keeping the image size constant.



CP467: Image Processing & Pattern Recognition

Effect of Reducing Gray Level Resolution...

(Continued) (e)-(h) Image displayed in 16, 8, 4, and 2 intensity levels.



Outline

- Digital Image
- Image Sampling and Quantization
- Image Interpolation**
- Relationships between Pixels
- Mathematical Tools used in DIP

Image Interpolation

- An image $f(x,y)$ tells us the intensity values at the integral lattice locations, i.e., when x and y are both **integers**
- Image interpolation refers to the “guess” of intensity values at **missing** locations, i.e., x and y can be arbitrary
- Note that it is just a **guess** (all sensors have finite sampling distance)

Engineering Motivations

- Why do we need image interpolation?
 - We want **big** images
 - When we see a video clip on a PC, we like to see it in the full screen mode
 - We want **good** images
 - If some block of an image gets damaged during the transmission, we want to repair it
 - We want **cool** images
 - Manipulate images digitally can render fancy artistic effects as we often see in movies

Scenario I: Resolution Enhancement

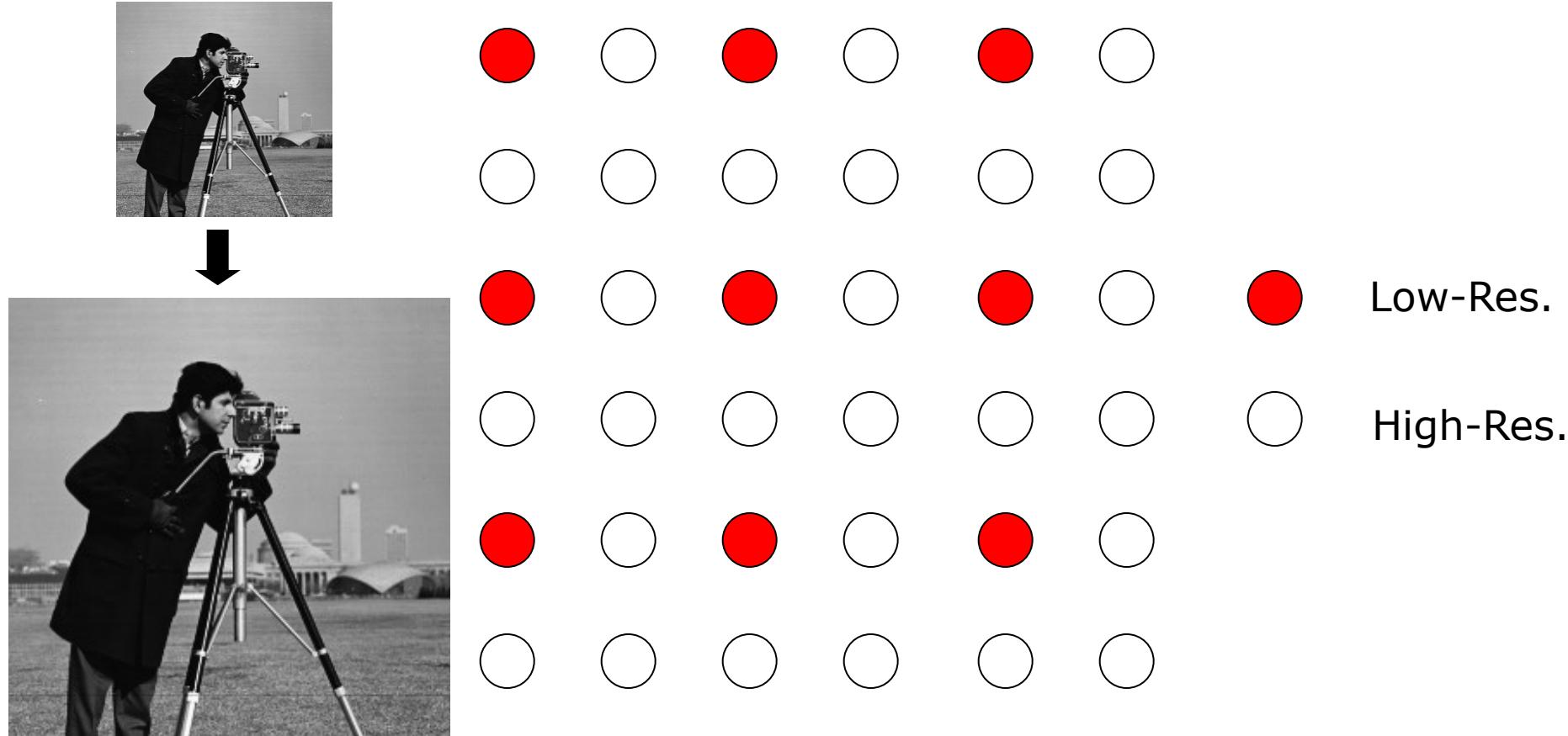


Image source: Xin Li

Scenario II: Image Inpainting

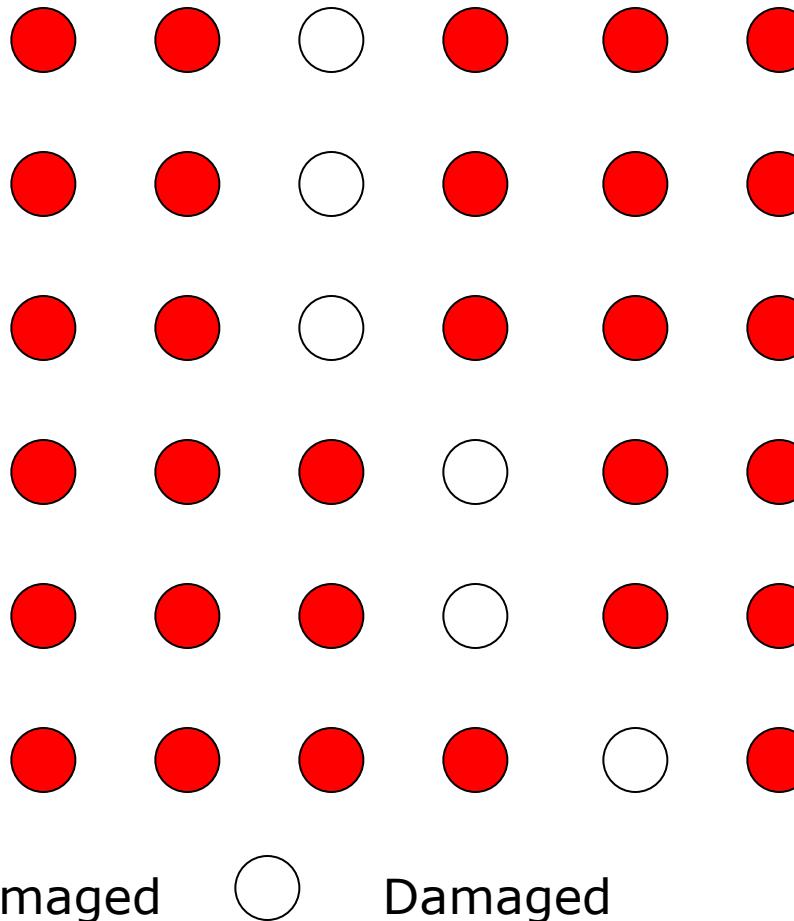
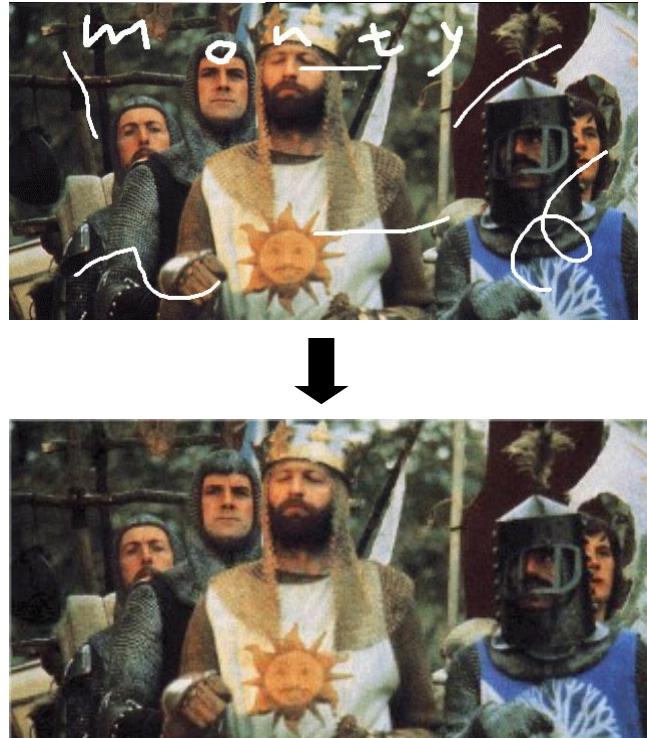


Image source: Xin Li

Scenario III: Image Warping

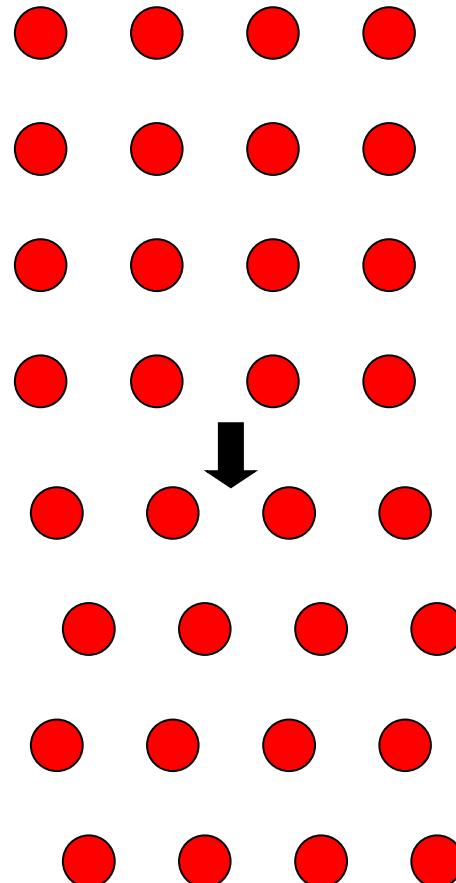
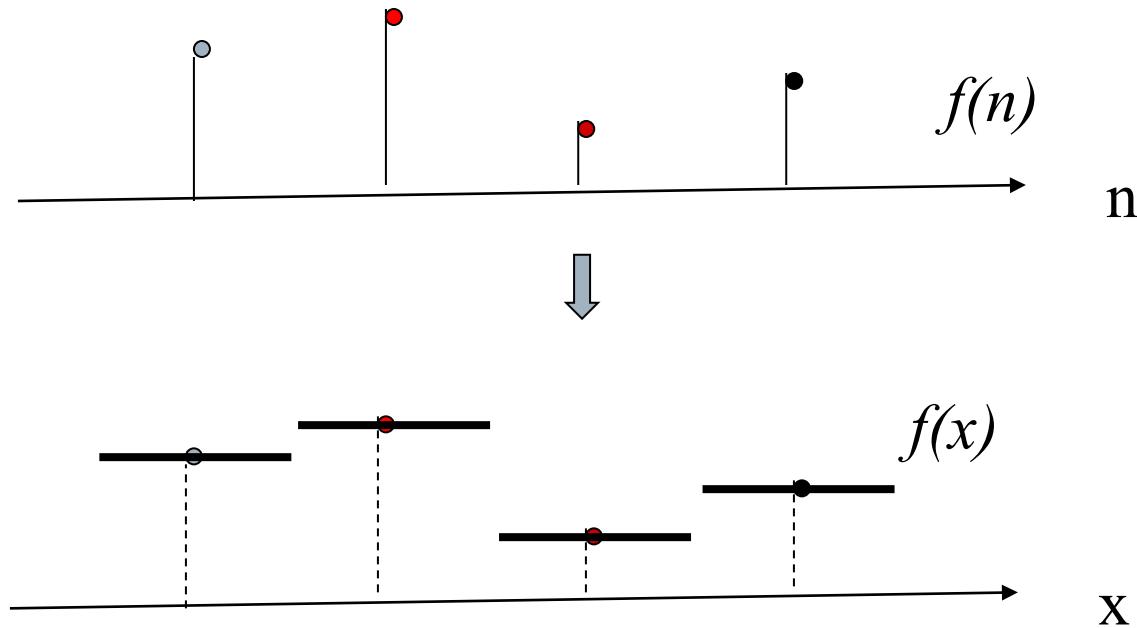


Image source: Xin Li

Image Interpolation

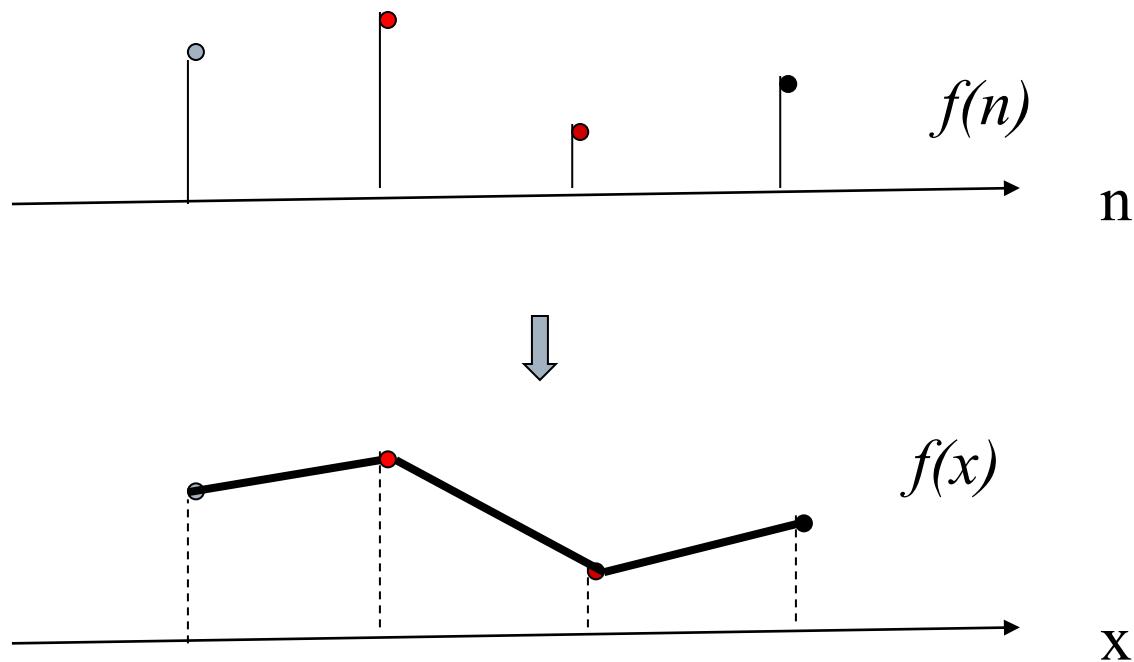
- Interpolation is the process of using known data to estimate values at unknown locations.
- Image Interpolation Techniques
 - 1. Nearest neighbor interpolation
 - Nearest pixel only
 - 2. Bilinear interpolation
 - 4 nearest neighbors
 - 3. Bicubic interpolation
 - 16 nearest neighbors

1D Nearest Neighbor Interpolation



- Simple and fast technique but results in distortion of features especially straight edges

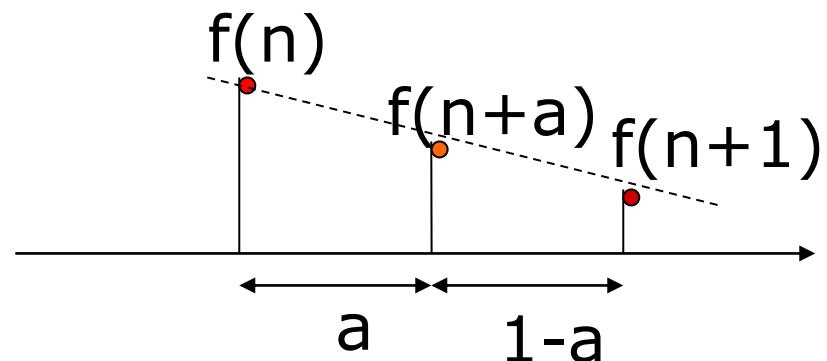
1D Linear Interpolation



- Gives a more aesthetically pleasing result

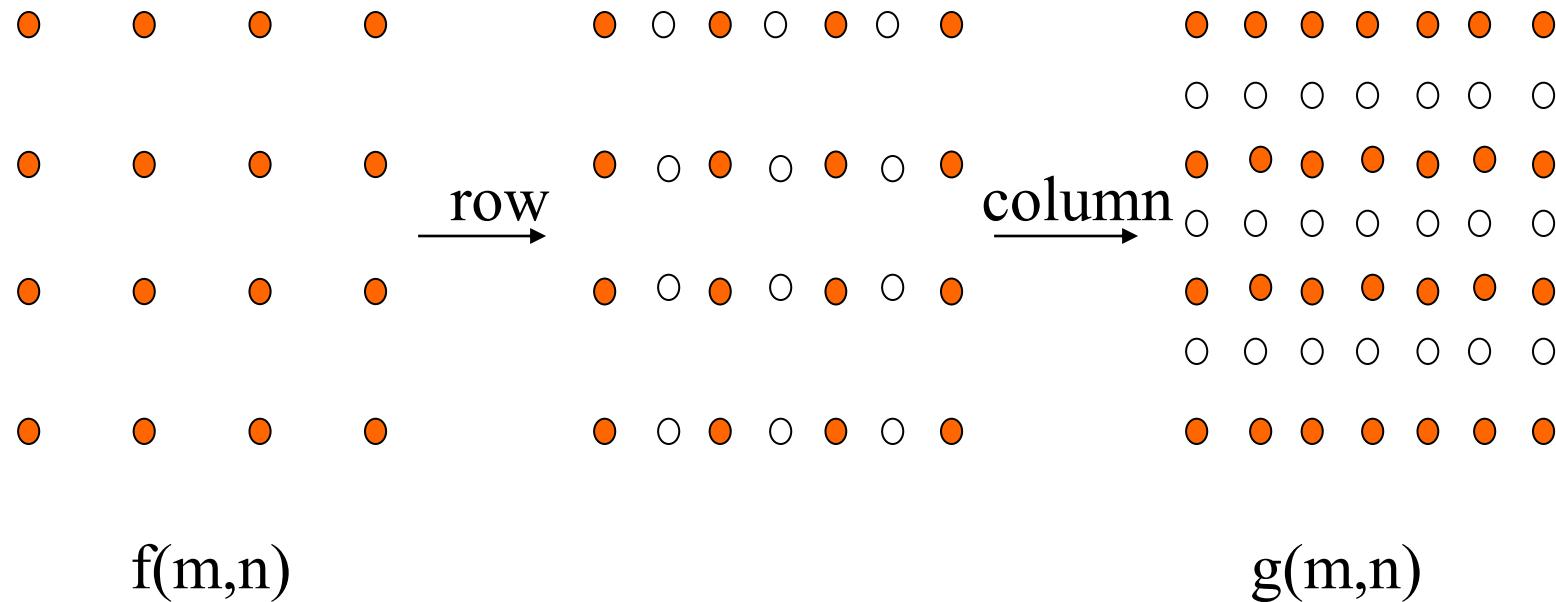
1D Linear Interpolation

Basic idea: the closer to a pixel, the higher weight is assigned



$$f(n+a) = (1-a)f(n) + a f(n+1), \quad 0 < a < 1$$

From 1D to 2D



Bicubic Interpolation

- Bicubic interpolation uses 16 nearest neighbors (4×4 neighborhood)
- Fit a series of cubic polynomials
 1. In y, fit 4 polynomials each using 4 control points
 2. Fit another cubic polynomial in x using polynomials found in step 1 and unknown pixel's address in y
 3. Substitute unknown pixel's address in x into the polynomial found in step 2 to get unknown pixel's brightness value

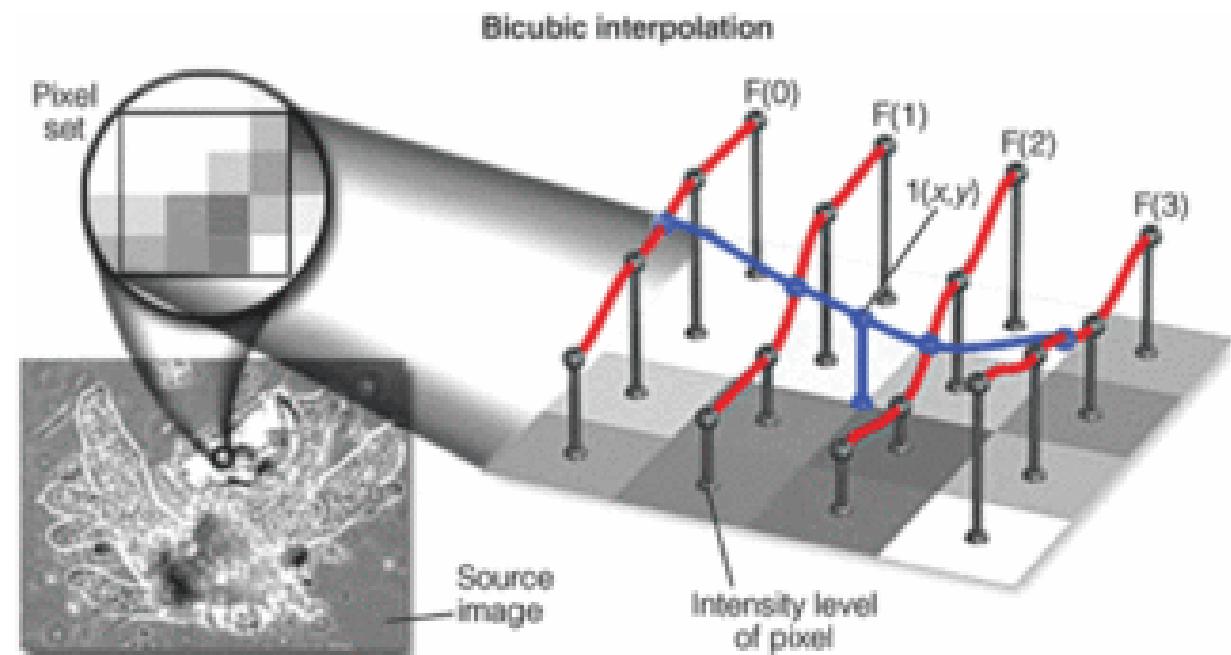
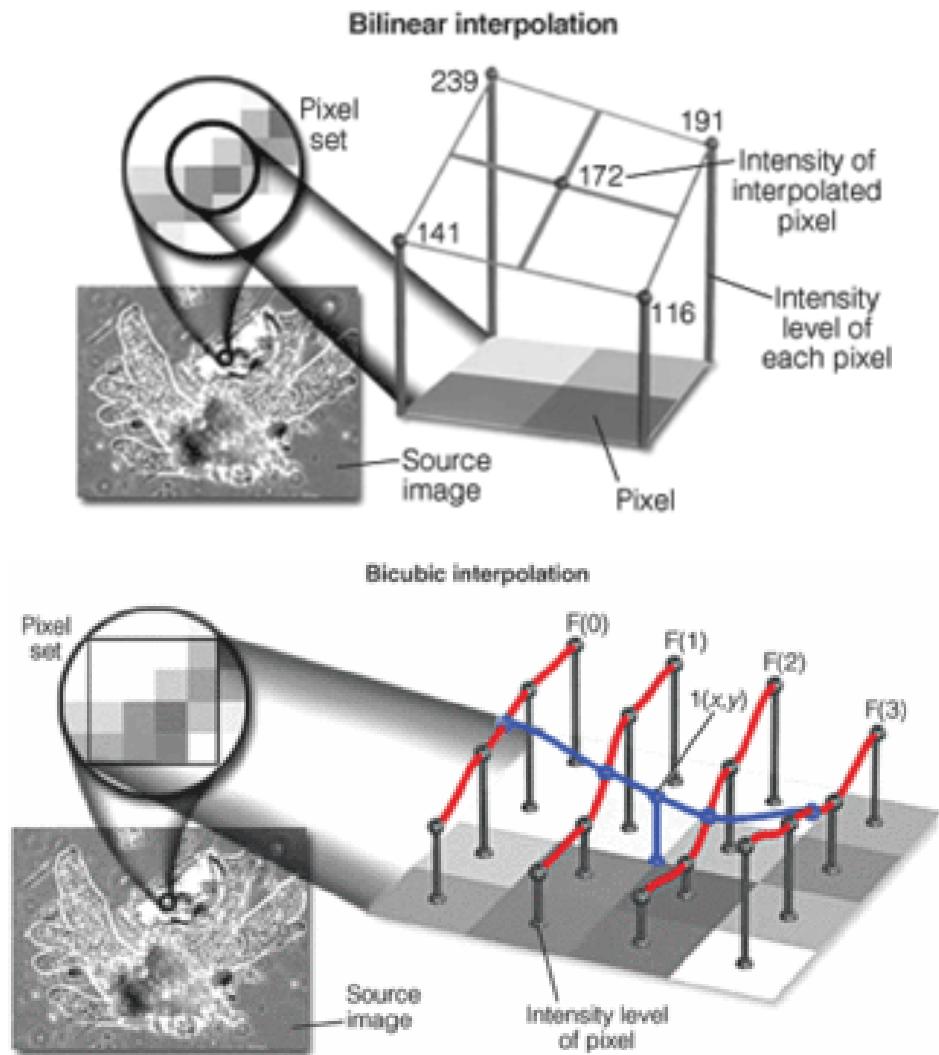


Image source: <https://www.vision-systems.com/boards-software/article/16738873/understanding-imageinterpolation-techniques>

Bilinear Vs Bicubic Interpolation

- Bilinear uses 4 nearest neighbors to determine the output, while Bicubic uses 16 nearest neighbors (4×4 neighborhood).
- Weight distribution is done differently.
- Bicubic interpolation gives better results with fewer interpolation artifacts but is more computationally demanding



Effects of Image Interpolation



- (a) Image reduced to 72 dpi and zoomed back to its original 930 dpi using **nearest neighbor interpolation**.
- (b) Image reduced to 72 dpi and zoomed using **bilinear interpolation**.
- (c) Same as (b) but using **bicubic interpolation**

Image source: Gonzalez and Woods

Outline

- Digital Image
- Image Sampling and Quantization
- Image Interpolation
- **Relationships between Pixels**
- Mathematical Tools used in DIP

Relationships Between Pixels

- Involves concepts of:
 - Neighborhood
 - Adjacency
 - Connectivity
 - Distance measures

Neighborhood

- 4-neighbors of p ($N_4(p)$)
- 4 diagonal neighbors of p ($N_D(p)$)
- 8-neighbors of p ($N_8(p)$)

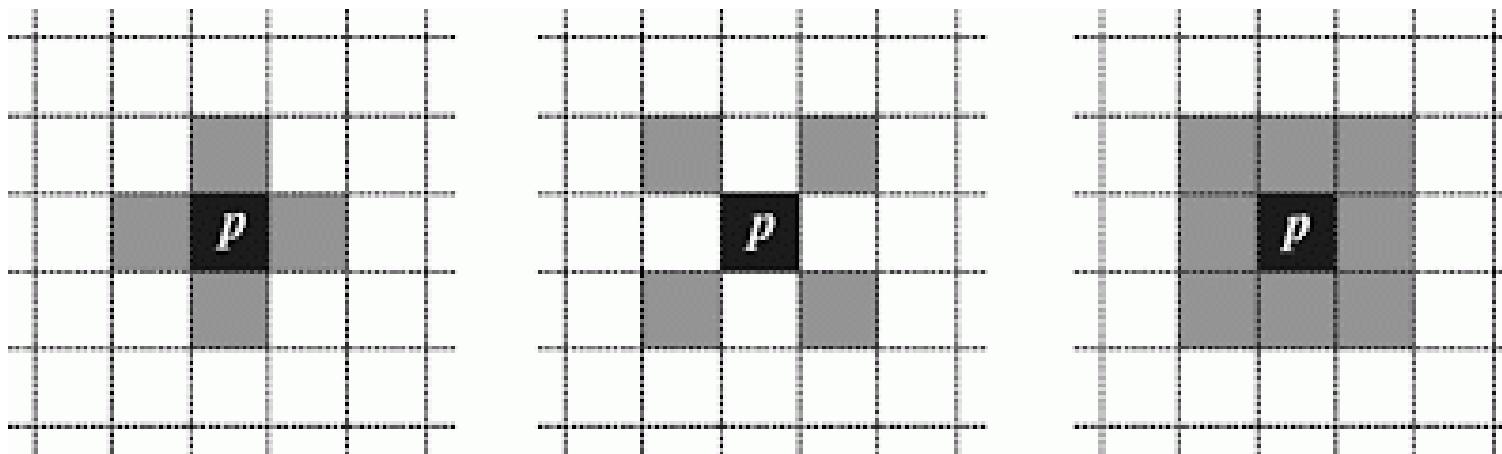


Image source: https://www.researchgate.net/figure/Fig-11-a-Pixel-p-and-its-4-neighbors-b-diagonal-neighborhood-c-8-neighborhood_fig10_301201506

Adjacency

□ 4-adjacency

- Two pixels p and q with values from V (sub-set of intensities) are 4-adjacent if q is in the set $N_4(p)$.

□ 8-adjacency

- Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.

□ m-adjacency (mixed)

- Two pixels p and q with values from V are m-adjacent if
 - q is in $N_4(p)$, or
 - q is in $N_D(p)$ and the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

0	1	1
0	1	0
0	0	1

0	1	1
0	1	0
0	0	1

0	1	1
0	1	0
0	0	1

Image source: Gonzalez and Woods

Connectivity

- Path
 - A sequence of adjacent pixels with values from V
- Closed path
 - If start point = end point
- Connected pixels
 - p and q are connected if there is a path between p and q
- Connected component
 - A set of connected pixels

Distance Measures

- Let $p(x,y)$, $q(s,t)$ and $z(v,w)$ be three points, a distance function D has 3 properties
 - $D(p,q) \geq 0$
 - $D(p,q) = D(q,p)$
 - $D(p,z) \leq D(p,q) + D(q,z)$
- Euclidean distance
 - $D(p,q) = [(x-s)^2 + (y-t)^2]^{1/2}$
- D_4 distance (city-block or Manhattan distance)
 - $D(p,q) = |x-s| + |y-t|$
- D_8 distance (chessboard or Chebyshev distance)
 - $D(p,q) = \max(|x-s|, |y-t|)$

Outline

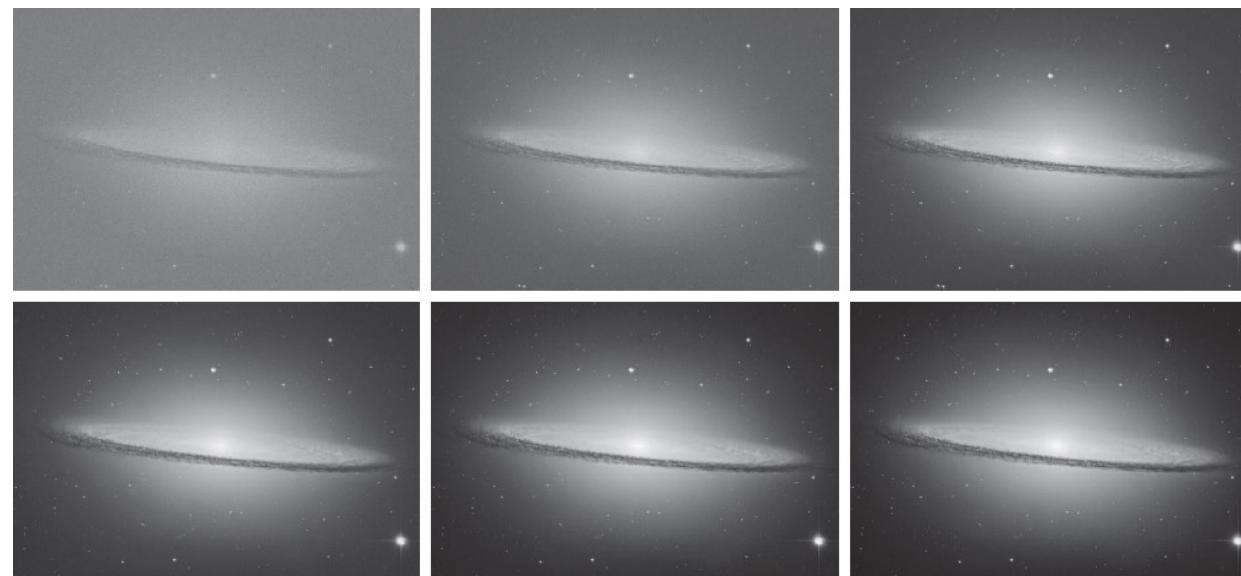
- Digital Image
- Image Sampling and Quantization
- Image Interpolation
- Relationships between Pixels
- Mathematical Tools used in DIP

Arithmetic Operations

- Arithmetic operations are +, -, * and /
- Addition
 - Noisy images can be added (averaged) to reduce noise (if noise is uncorrelated and has zero mean)
 - Images must be registered first
 - An important application area is astronomy (see images on next slide)

Arithmetic Operations...

Figure 2.29: (a) Sample noisy image of the Sombrero Galaxy. (b)-(f) Result of averaging 10, 50, 100, 500, and 1,000 noisy images, respectively. All images are of size 1548×2238 pixels, and all were scaled so that their intensities would span the full $[0, 255]$ intensity scale. (Discovered in 1767, the Sombrero Galaxy is 28 light years from Earth. Original image courtesy of NASA.)



a b c

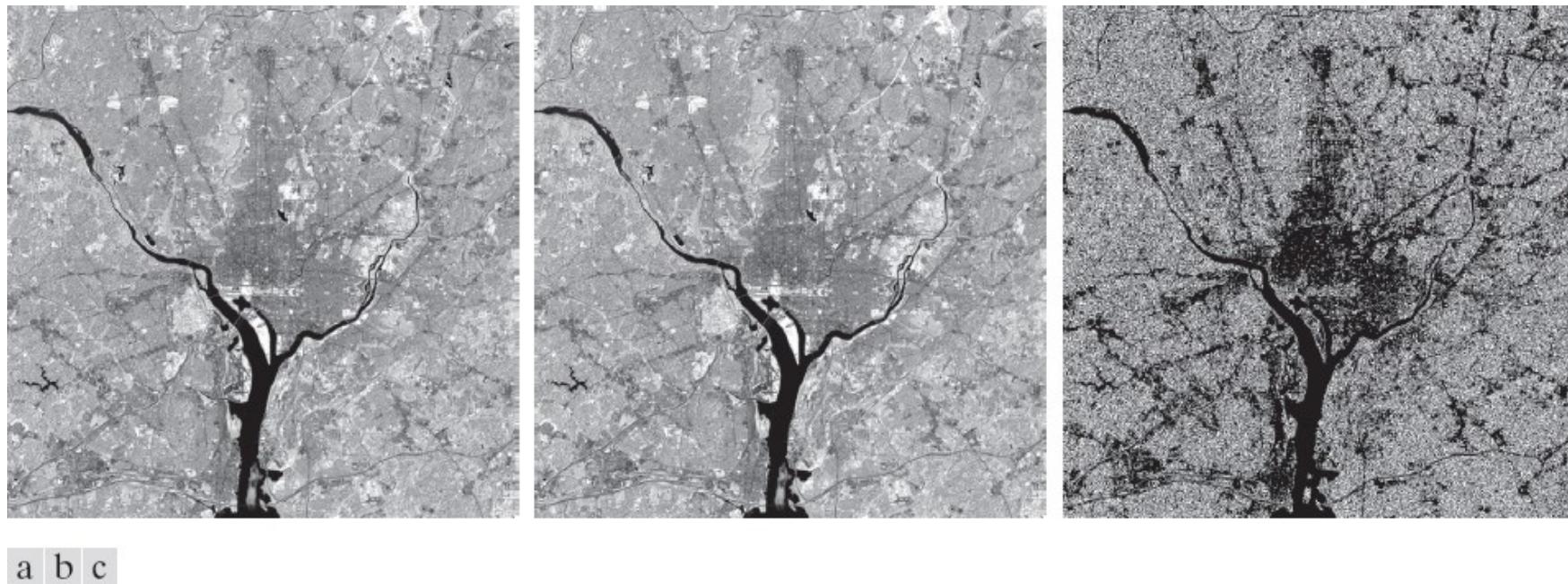
CP467: Image Processing & Pattern Recognition

Arithmetic Operations...

□ Subtraction

- A frequent application is in enhancement of differences between images

Figure 2.30: (a) Infrared image of the Washington, D.C. area. (b) Image resulting from setting to zero the least significant bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity. (Original image courtesy of NASA.)



Arithmetic Operations...

Figure 2.31: (a) Difference between the 930 dpi and 72 dpi images in Fig. 2.23. (b) Difference between the 930 dpi and 150 dpi images. (c) Difference between the 930 dpi and 300 dpi images.



a | b | c

Arithmetic Operations...

Figure 2.34: (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).



References and Credits

- Gonzalez and Woods, Ch. 2

Image Processing & Pattern Recognition

Intensity Transformations and Histogram Processing

Dr. Zia Ud Din

Outline

- Intensity Transformations (Point Operations)
- Histogram Processing

Image Enhancement

- The principal objective of **image enhancement** is to process an image so that the result is more suitable than the original image for a **specific** application.
- **Image enhancement** approaches fall into **two** broad categories:
 - Spatial domain methods
 - Frequency domain methods
- Spatial domain refers to the image plane itself
 - Approaches in this category are based on **direct manipulation** of pixels in an image.
- Frequency domain processing techniques are based on modifying the **Fourier transform** of an image.

Image Enhancement in Spatial Domain

- Spatial domain processes will be denoted by the expression

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

- $f(x, y)$ is the intensity of input image at pixel (x, y) ,
- $g(x, y)$ is the intensity of processed image at pixel (x, y) ,
- T is an operator on f , defined over some neighborhood of (x, y)

- Point operations use pixel value at (x, y) only
- Neighborhood operations/filters use square or rectangular sub-image around (x, y)
 - The center of the sub-image is moved from pixel to pixel

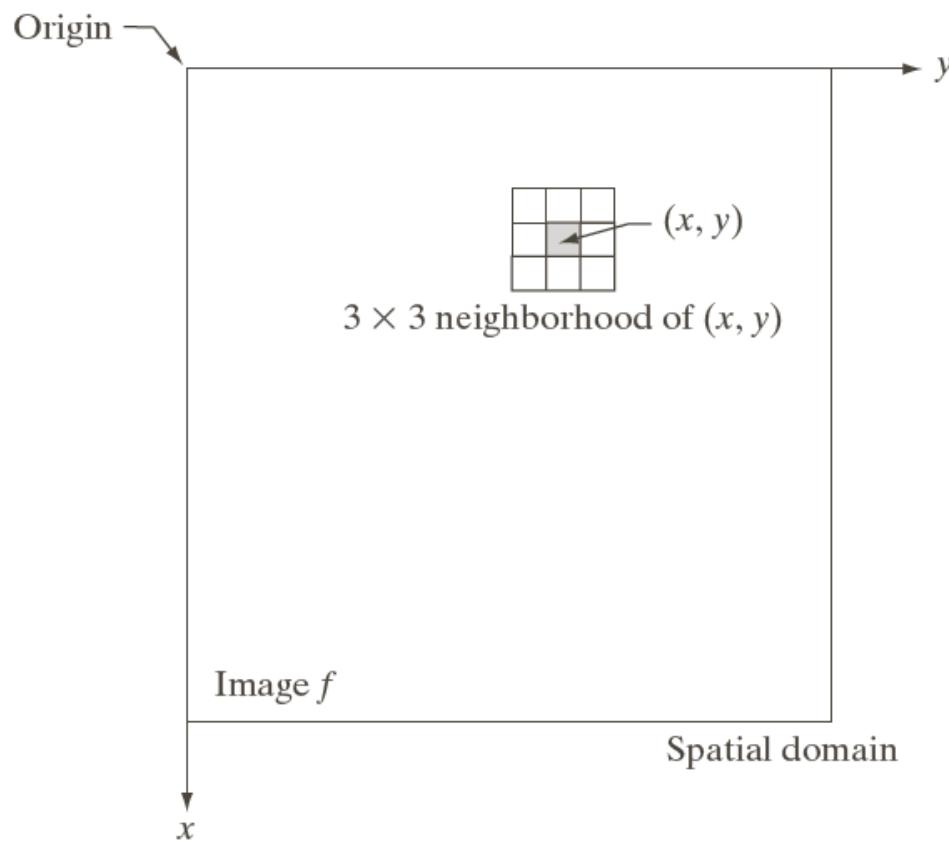


FIGURE 3.1
A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

Image source: Gonzalez and Woods

Point Operations

- These are among the **simplest** of all image enhancement techniques.
- The values of pixels, before and after processing, will be denoted by r and s , respectively.
- These values are related $s = T(r)$, where T is a transformation

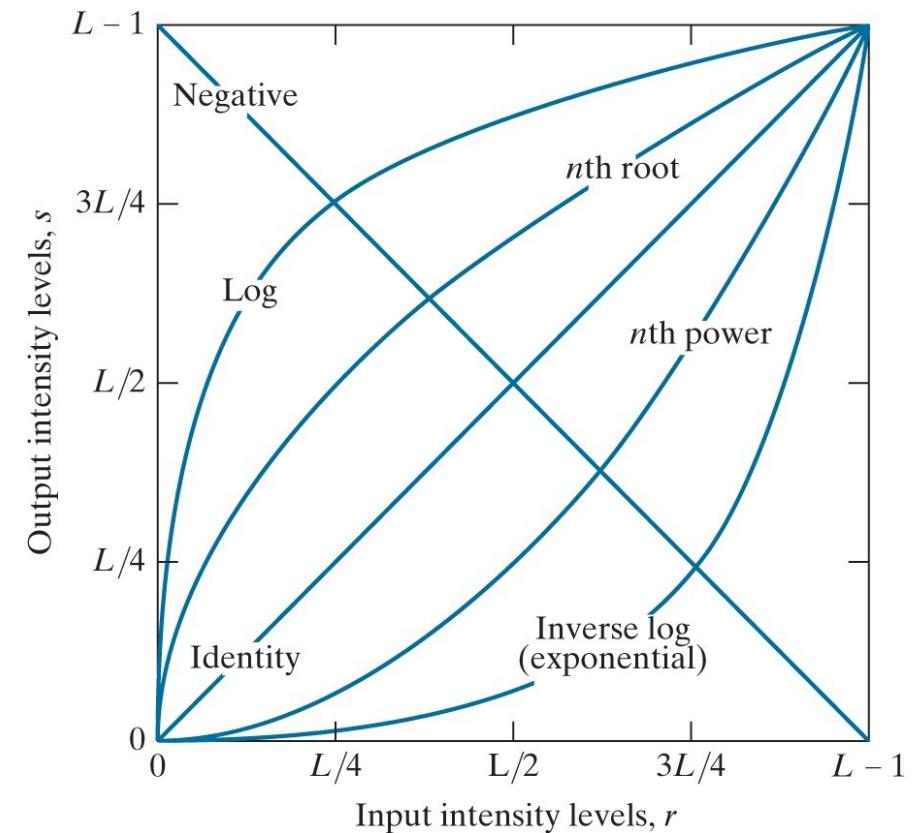


Image source: Gonzalez and Woods

Point Operations

- Image Negatives
- Log Transformations
- Power-Law Transformations
- Piecewise-Linear Transformation Functions

Point Operations: Image Negatives

- The negative of an image with gray levels in the range $[0, L-1]$ is obtained by:

$$s = L-1-r$$

- Reversing in this manner produces equivalent of a photographic negative.
- Used for enhancing bright detail embedded in dark regions
 - Especially when the dark areas are dominant in size.

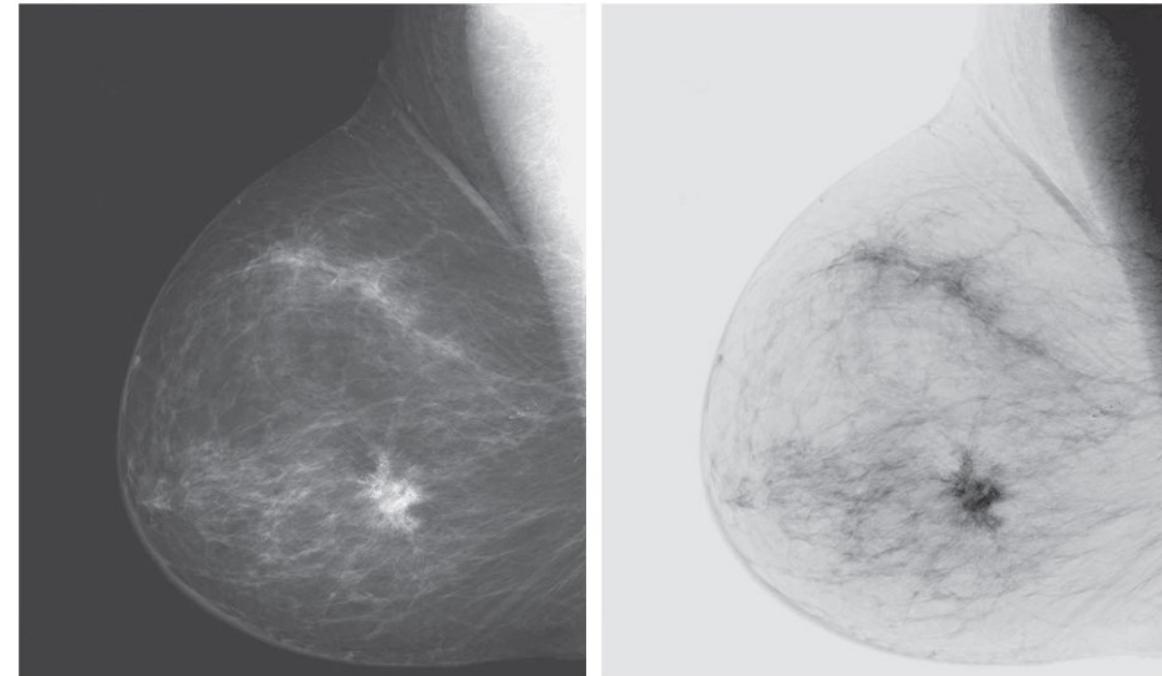


Image source: Gonzalez and Woods

Point Operations: Log Transformations

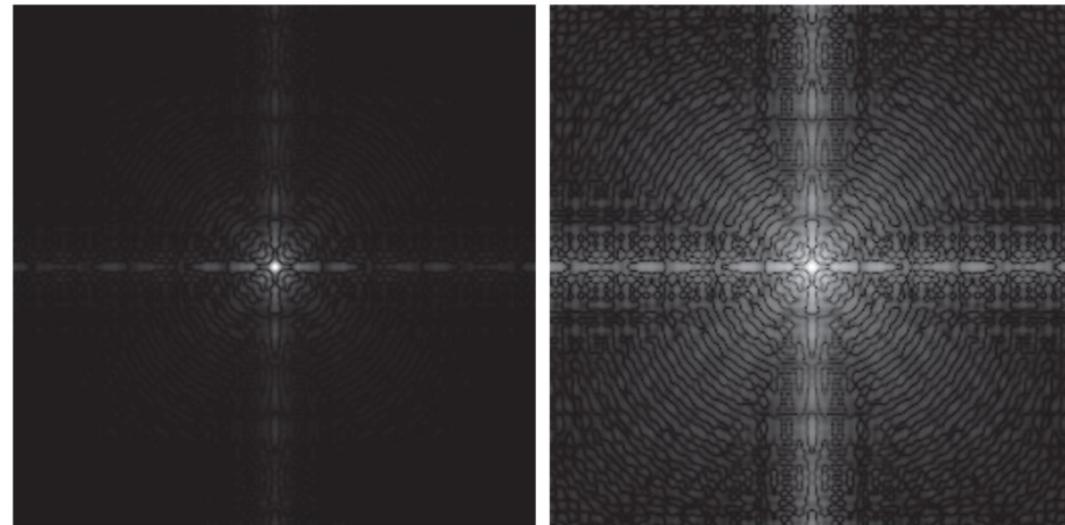
- The general form of the log transformation is:

$$s = c \log (1 + r)$$

- This transformation maps a **narrow** range of low gray-level values in the input image into a **wider** range of output levels.

FIGURE 3.5

a b



(a) Fourier spectrum displayed as a grayscale image. (b) Result of applying the log transformation in Eq. (3-4) with $c = 1$. Both images are scaled to the range [0, 255].

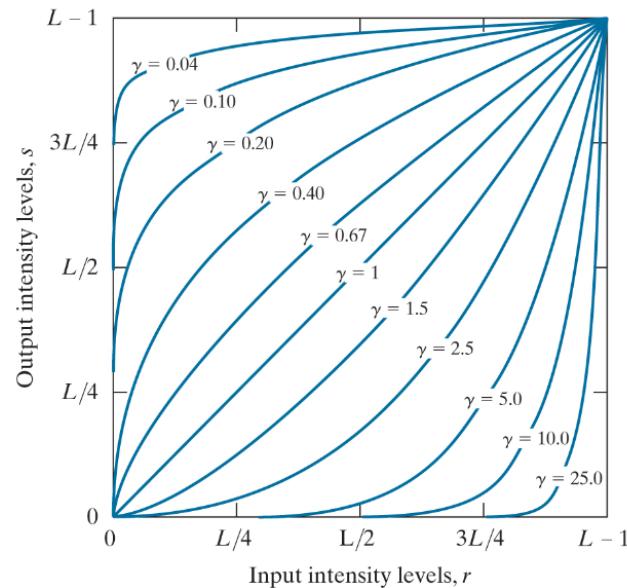
Image source: Gonzalez and Woods

Point Operations: Power-Law Transformations

- Power-law transformations have the basic form:

$$s = cr^\gamma$$

FIGURE 3.6



Plots of the gamma equation $s = cr^\gamma$ for various values of γ ($c = 1$ in all cases). Each curve was scaled independently so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.

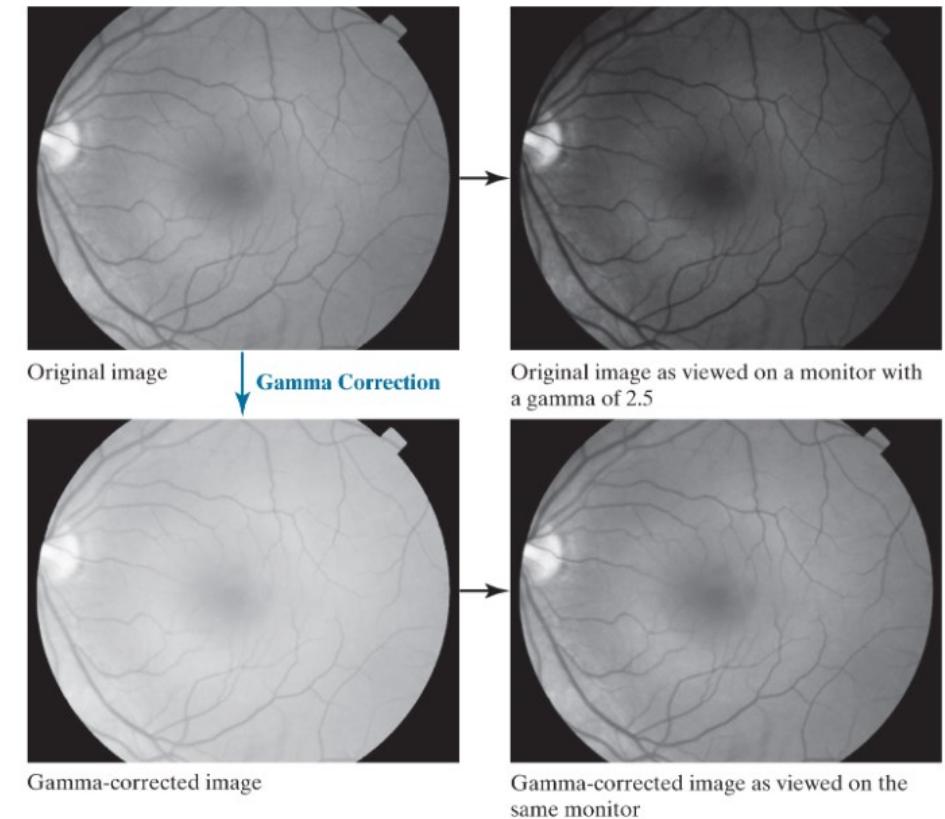
Image source: Gonzalez and Woods

Point Operations: Power-Law Transformations...

- Cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function
 - with exponents varying from approximately 1.8 to 2.5.
- Such display systems would produce images that are **darker** than intended
- The process used to correct this power-law response phenomena is called **gamma correction**.

FIGURE 3.7

a
b
c
d



(a) Image of a human retina. (b) Image as it appears on a monitor with a gamma setting of 2.5 (note the darkness). (c) Gamma-corrected image. (d) Corrected image, as it appears on the same monitor (compare with the original image). (Image (a) courtesy of the National Eye Institute, NIH)

Image source: Gonzalez and Woods

Point Operations: Power-Law Transformations

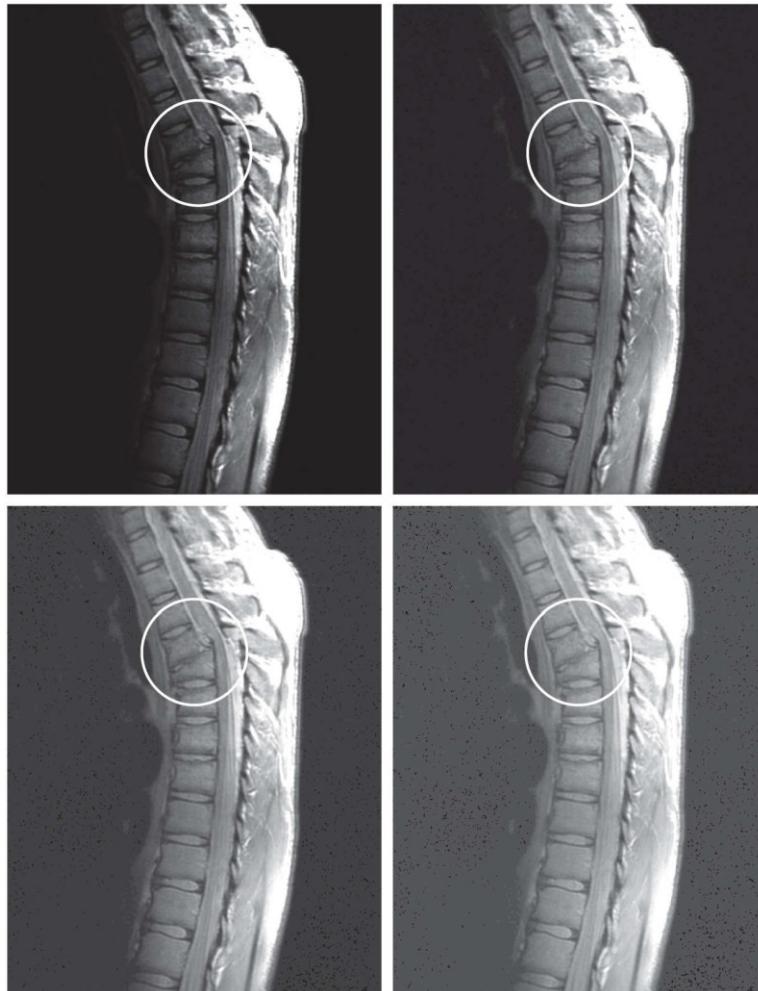


FIGURE 3.8
(a) Magnetic resonance image (MRI) of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3 , respectively. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)



FIGURE 3.9
(a) Aerial image.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0 , respectively. (Original image for this example courtesy of NASA.)

Image source: Gonzalez and Woods

Point Operations: Piecewise-Linear Transformation Functions

- The principal **advantage**
 - The form of piecewise functions can be arbitrarily complex.
- The principal **disadvantage**
 - Their specification requires considerably more user input.
- Difference with other techniques like histogram equalization is that equalization may be non-linear.
- Piecewise-linear transformation functions types:
 - Contrast Stretching
 - Gray-level Slicing
 - Bit-Plane Slicing

Contrast Stretching

- One of the simplest piecewise linear functions.
- Used to increase the **dynamic range** of the gray levels.
- Low-contrast images can result from
 - Poor illumination
 - Lack of dynamic range in the imaging sensor
 - Wrong setting of a lens aperture during image acquisition

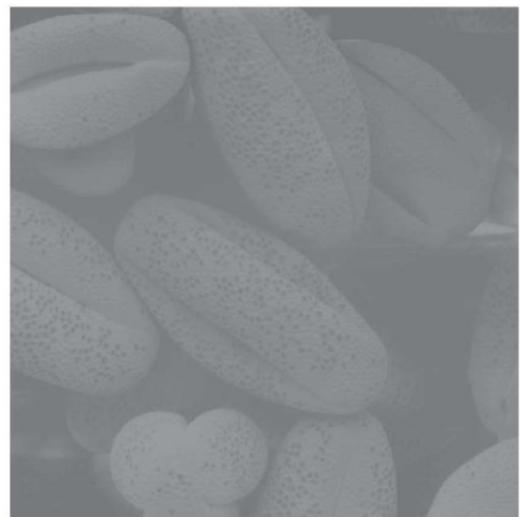
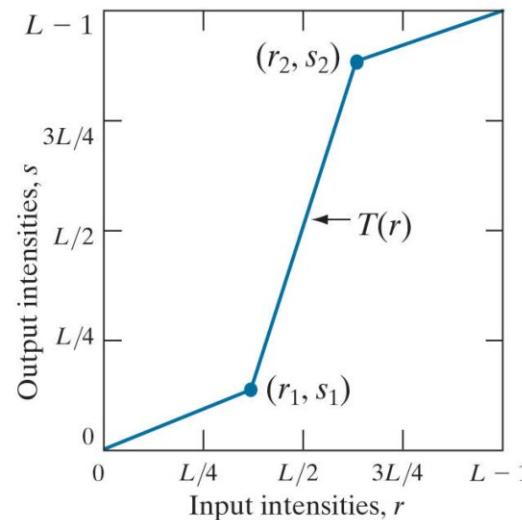


Image source: Gonzalez and Woods

Contrast Stretching

- The locations of points (r_1, s_1) and (r_2, s_2) control the shape of the transformation function.
- A common contrast stretching is obtained by setting
 - $(r_1, s_1) = (r_{\min}, 0)$ and $(r_2, s_2) = (r_{\max}, L-1)$
 - First enhanced image on the last slide uses that
- The second enhanced image on the last slide uses
 - $(r_1, s_1) = (m, 0)$ and $(r_2, s_2) = (m, L-1)$ where m is mean intensity of the image
 - This results in a **thresholding** function
- In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed so that the function is **single valued** and **monotonically increasing**

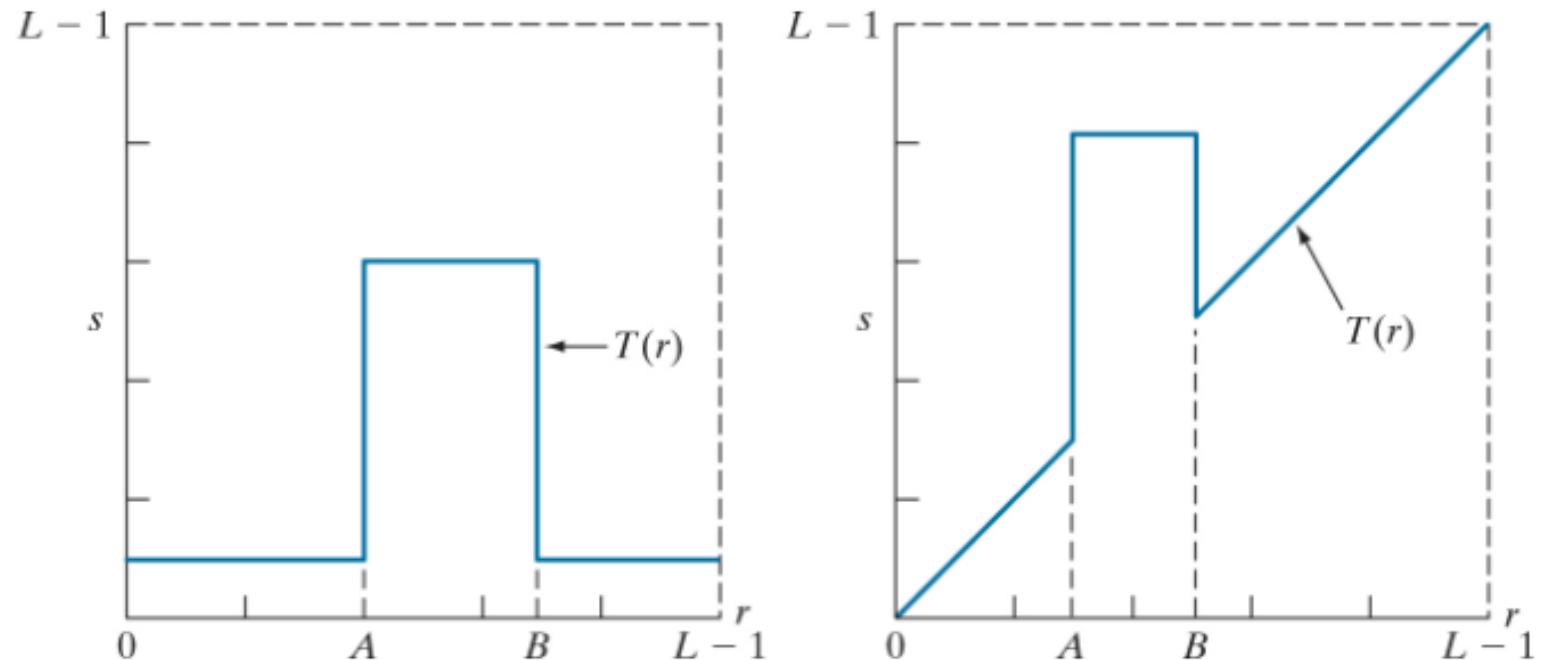
Gray-level Slicing

- Highlighting a specific range of gray levels in an image is often desired.
- Applications include enhancing features such as masses of water in satellite imagery and enhancing flaws in X-ray images.
- There are several ways of doing gray-level slicing, but most of them are variations of two basic themes (shown on next slide)

Gray-level Slicing

FIGURE 3.11

a | b

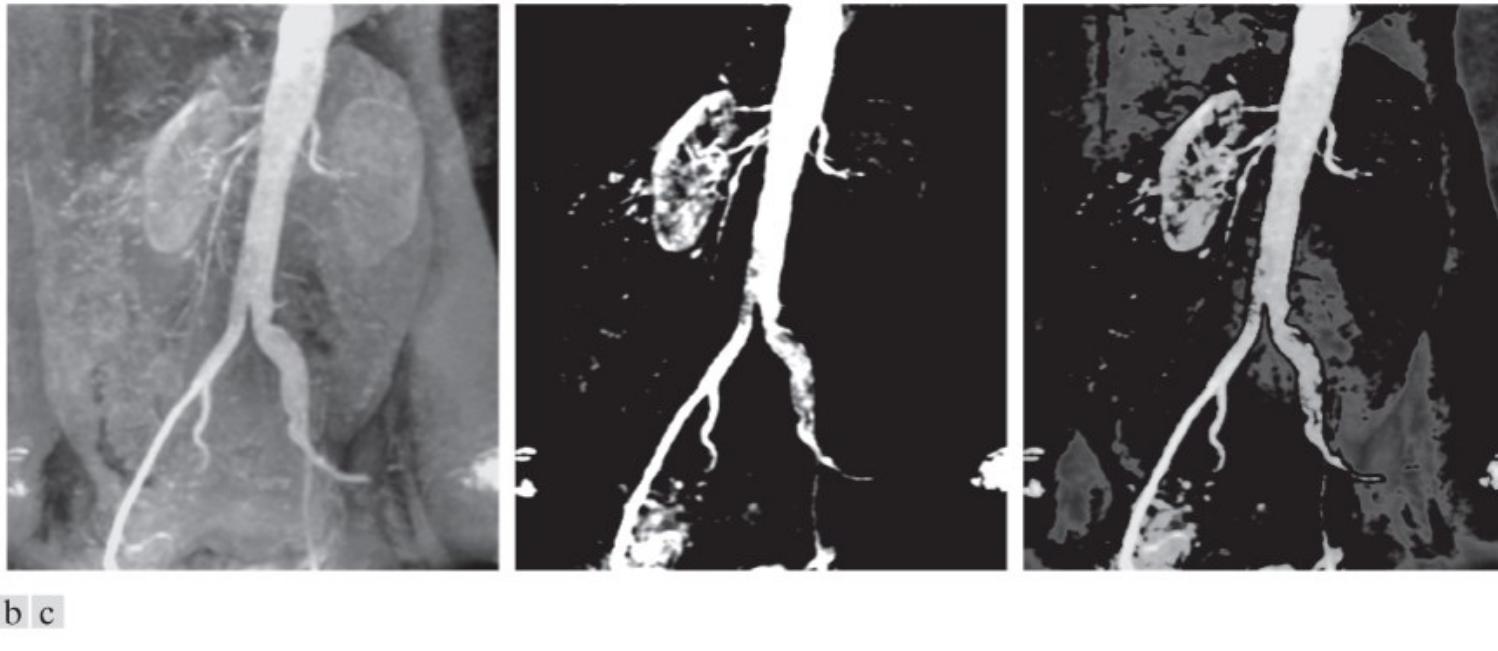


(a) This transformation function highlights range $[A, B]$ and reduces all other intensities to a lower level. (b) This function highlights range $[A, B]$ and leaves other intensities unchanged.

Image source: Gonzalez and Woods

Gray-level Slicing

FIGURE 3.12



(a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in **Fig. 3.11(a)**, with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in **Fig. 3.11(b)**, with the selected range set near black, so that the grays in the area of the blood vessels and kidneys were preserved. (Original image courtesy of Dr. Thomas R. Gest, University of Michigan Medical School.)

Image source: Gonzalez and Woods

Bit-Plane Slicing

- Highlights the contribution made to total image appearance by specific bits.
- Separation in bit planes aids in determining the adequacy of the number of bits used to **quantize** each pixel.
- Also, this type of decomposition is useful for **image compression**.

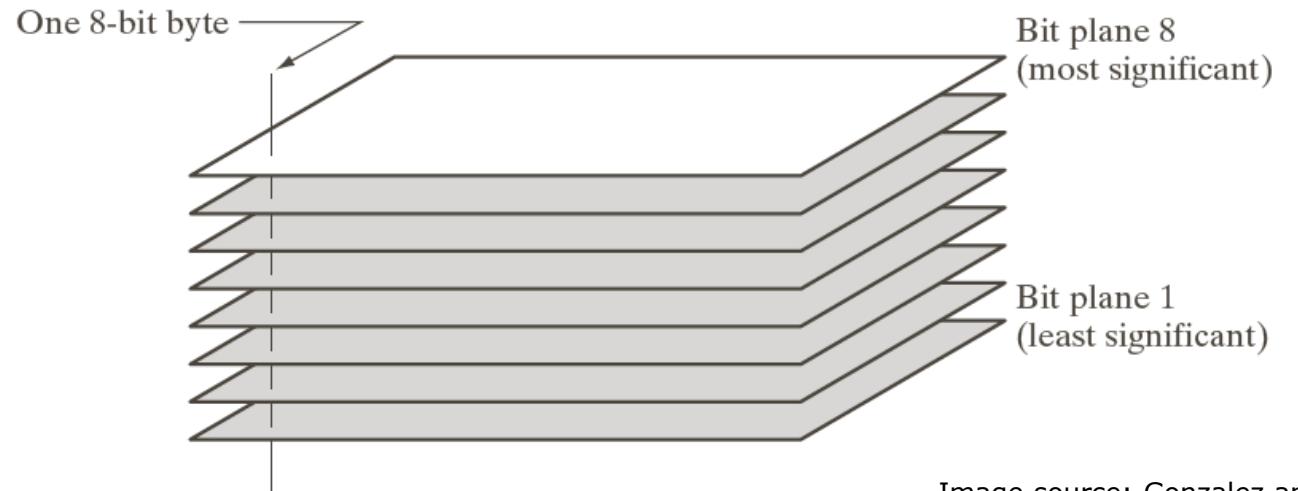
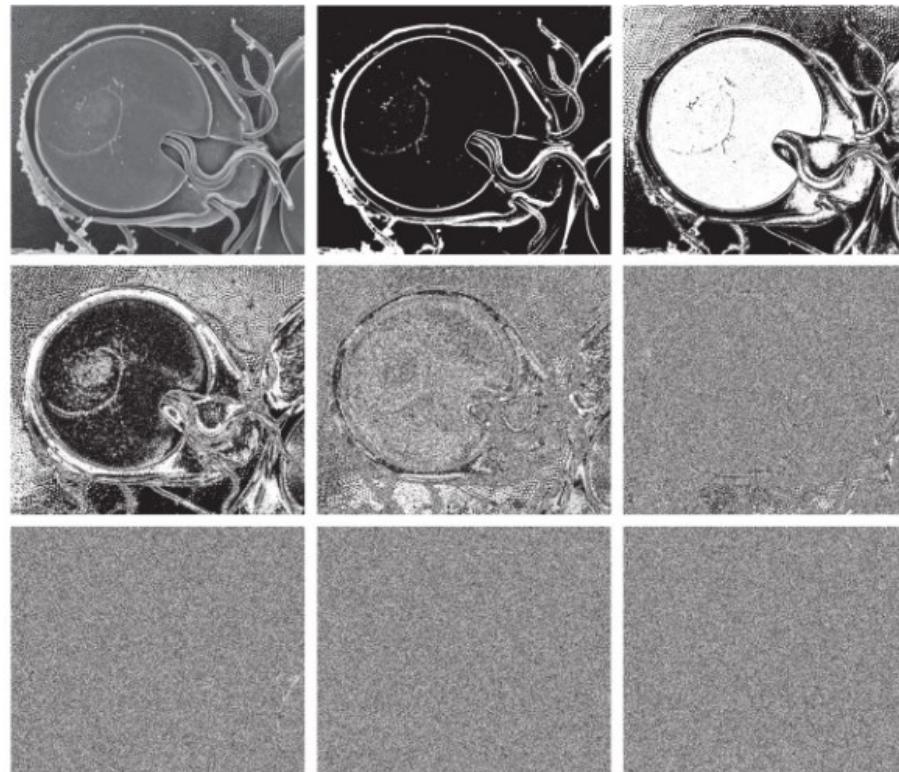


Image source: Gonzalez and Woods

Bit-Plane Slicing

FIGURE 3.14

a b c
d e f
g h i



(a) An 8-bit gray-scale image of size 837×988 pixels. (b) through (i) Bit planes 8 through 1, respectively, where plane 1 contains the least significant bit. Each bit plane is a binary image. Figure (a) is an SEM image of a trophozoite that causes a disease called *giardiasis*. (Courtesy of Dr. Stan Erlandsen, U.S. Center for Disease Control and Prevention.)

Image source: Gonzalez and Woods

Bit-Plane Slicing

FIGURE 3.15

a b c



Image reconstructed from bit planes: (a) 8 and 7; (b) 8, 7, and 6; (c) 8, 7, 6, and 5.

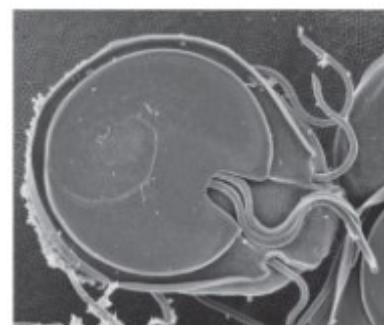


Image source: Gonzalez and Woods

Outline

- Intensity Transformations (Point Operations)
- Histogram Processing

Histogram Processing

- The unnormalized histogram of a digital image with gray levels in the range $[0, L-1]$ is a discrete function

$$h(r_k) = n_k \text{ for } k=0,1,2,\dots,L-1$$

- r_k is the k^{th} gray level
- n_k is the number of pixels in the image having gray level r_k .

- It is common practice to **normalize** a histogram

$$p(r_k) = n_k / MN$$

- MN is the number of pixels in the image (M rows & N cols)

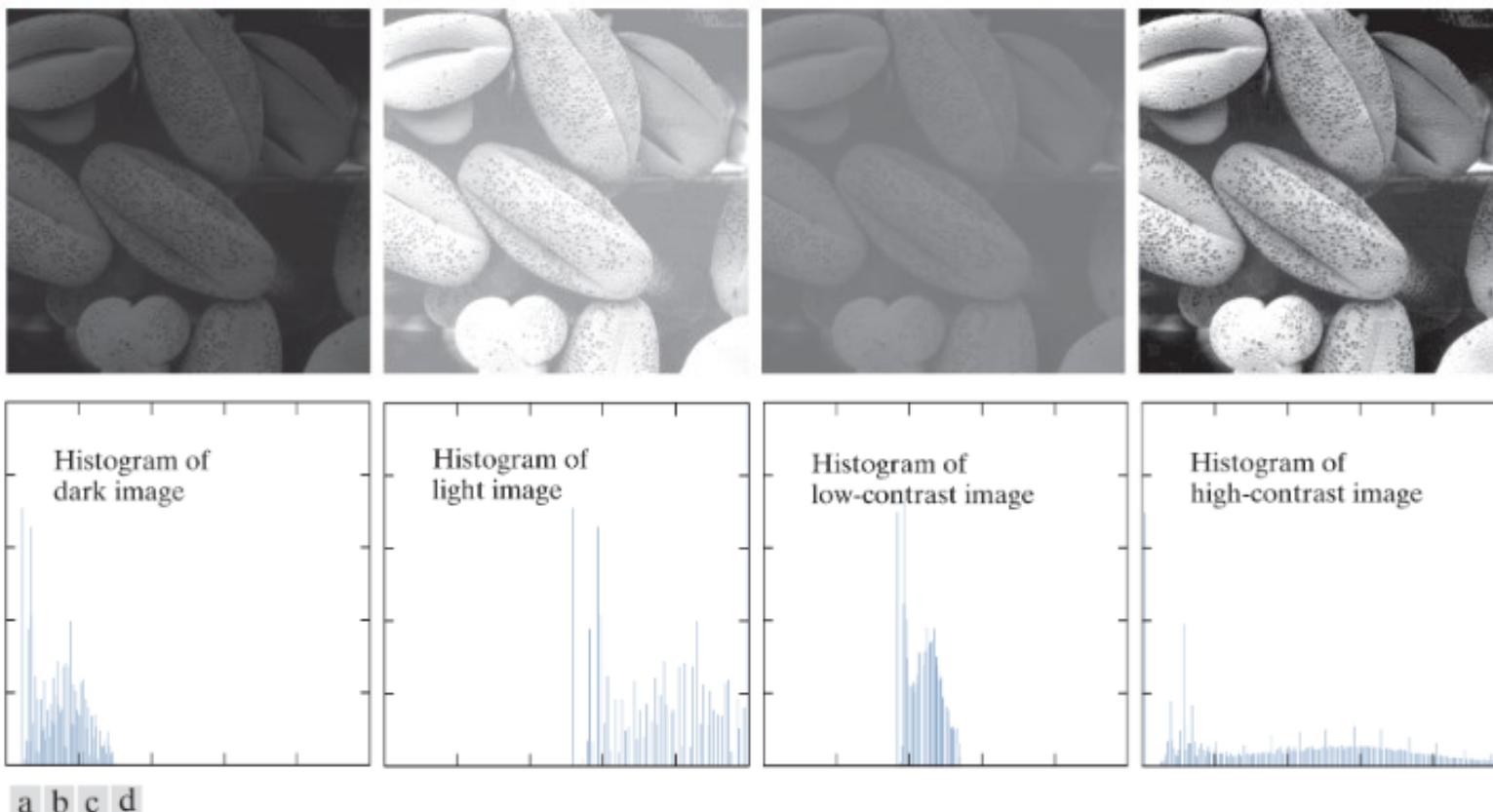
- The sum of $p(r_k)$ for all values of k is always 1

- Histograms are a **popular** tool for real-time image processing.

- Simple to implement in software and hardware requirement is economical

Histogram Processing...

FIGURE 3.16



Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; (d) high contrast. The horizontal axis of the histograms are values of r_k and the vertical axis are values of $p(r_k)$.

Histogram Processing: Histogram Equalization

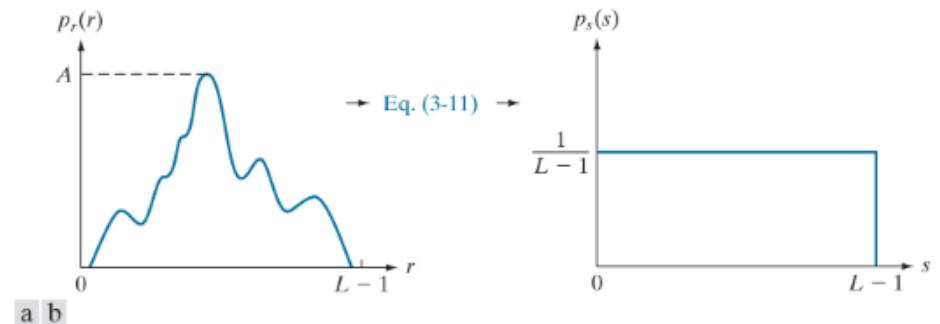
- An image whose pixels occupy entire range of possible gray levels and distributed uniformly, will have a high dynamic range.
- It is possible to develop a transformation function that can automatically achieve this effect,
 - Based only on information available in the histogram of the input image.
- Such a transformation is called **histogram equalization**.

Histogram Equalization

- The required transformation function (**cumulative distribution function**) for histogram equalization is given below:

$$s = T(r) = (L - 1) \int_0^r p_r(w) dw$$

FIGURE 3.18



(a) An arbitrary PDF. (b) Result of applying **Eq. (3-11)** to the input PDF. The resulting PDF is always uniform, independently of the shape of the input.

- Cumulative distribution function
 - Mapping of a continuous or discrete random variable with the cumulative probability of occurrence

Image source: Gonzalez and Woods

Histogram Equalization...

- For discrete values, we work with probabilities and summations instead of probability density functions and integrals

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L - 1$$

Histogram Equalization: Example

- Input image histogram (3-bit 64*64 image):

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

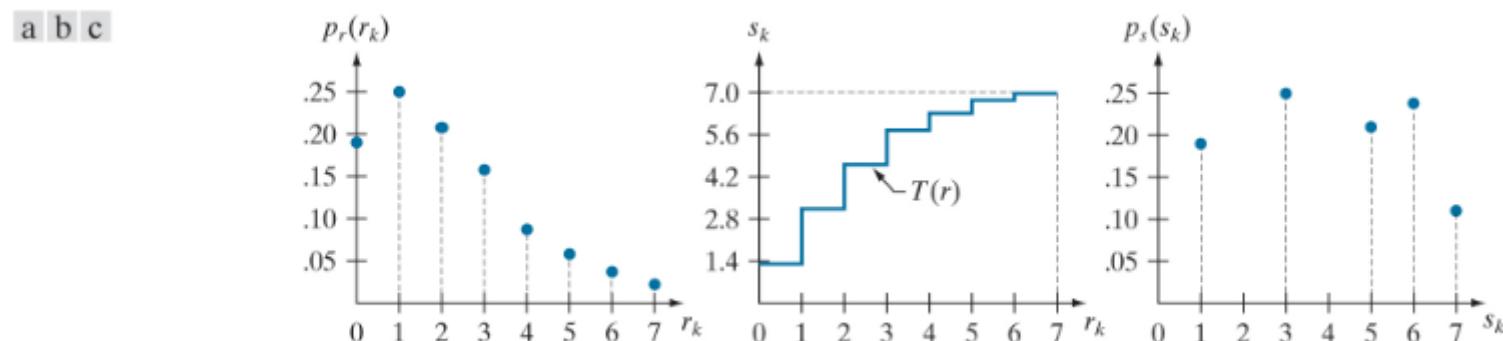
Histogram Equalization: Example...

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L - 1$$

Given $p_r(r_1) = 0.25, p_r(r_2) = 0.2, p_r(r_3) = 0.15, p_r(r_4) = 0.1, p_r(r_5) = 0.05, p_r(r_6) = 0.04, p_r(r_7) = 0.03$, we have $s_1 = T(r_1) = 3.08, s_2 = 4.55, s_3 = 5.67, s_4 = 6.23, s_5 = 6.65, s_6 = 6.86$, and $s_7 = 7.00$. This transformation function has the staircase shape shown in Fig. 3.19(b).

$$\begin{array}{llll} s_0 = 1.33 \rightarrow 1 & s_2 = 4.55 \rightarrow 5 & s_4 = 6.23 \rightarrow 6 & s_6 = 6.86 \rightarrow 7 \\ s_1 = 3.08 \rightarrow 3 & s_3 = 5.67 \rightarrow 6 & s_5 = 6.65 \rightarrow 7 & s_7 = 7.00 \rightarrow 7 \end{array}$$

FIGURE 3.19



Histogram equalization. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

Histogram Equalization

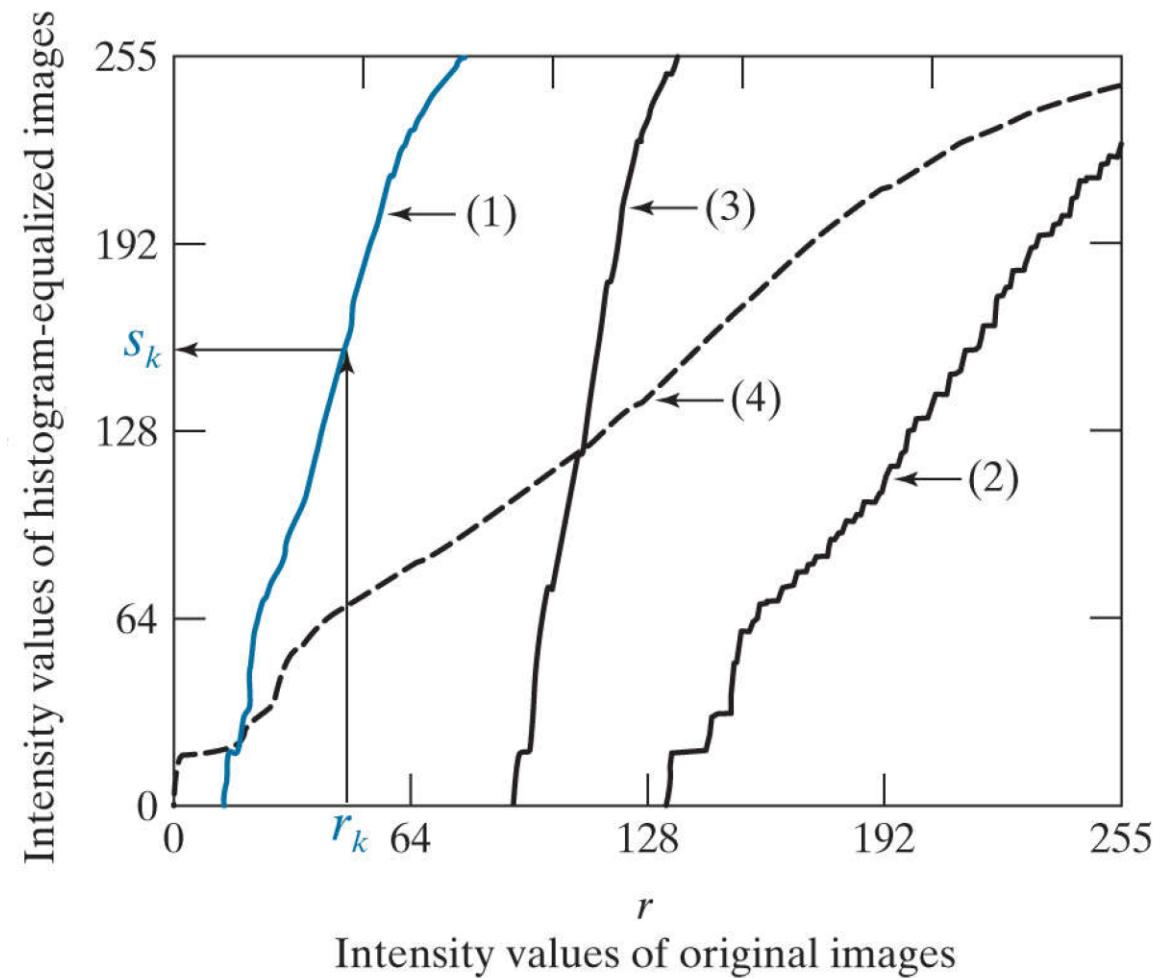
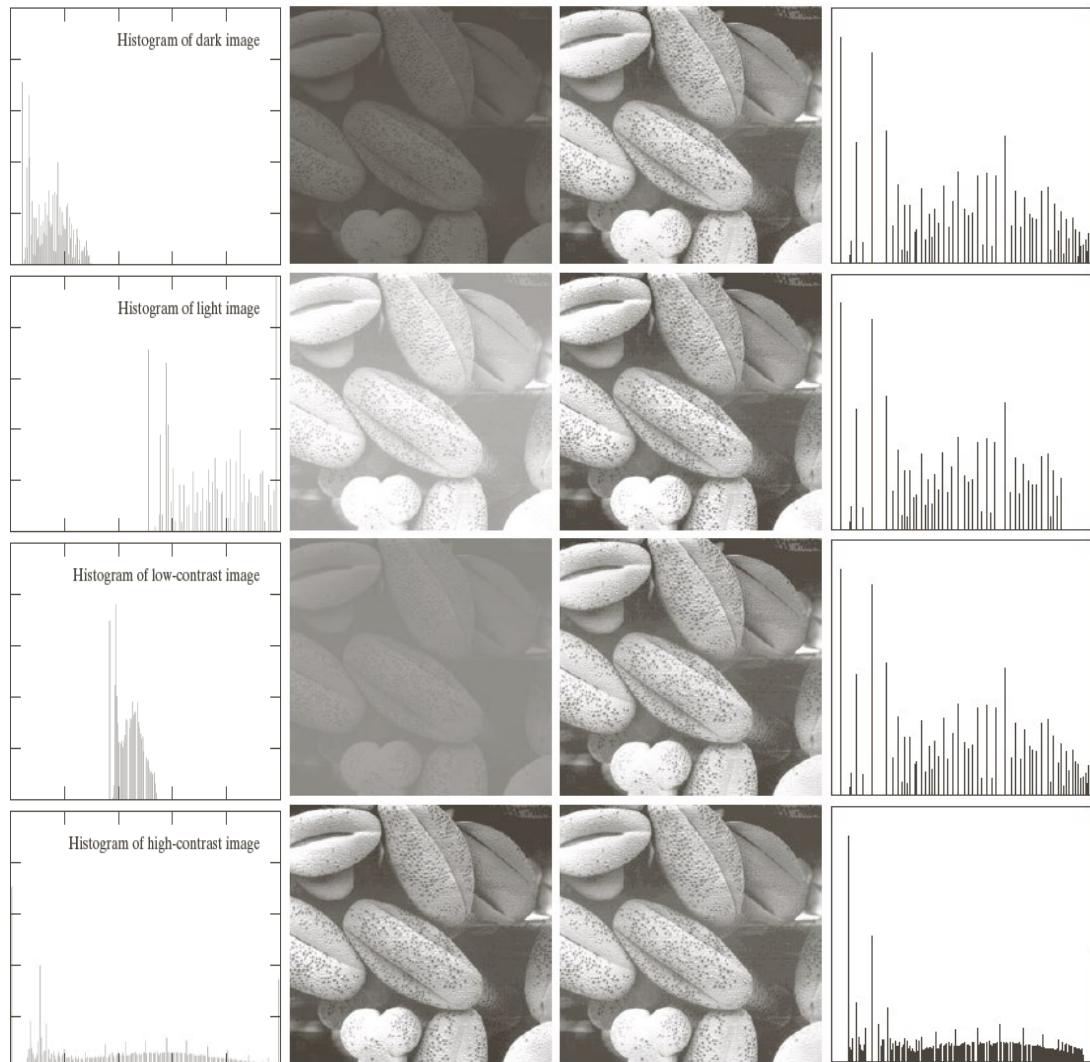


Image source: Gonzalez and Woods

Histogram Equalization...



(A) Original Image



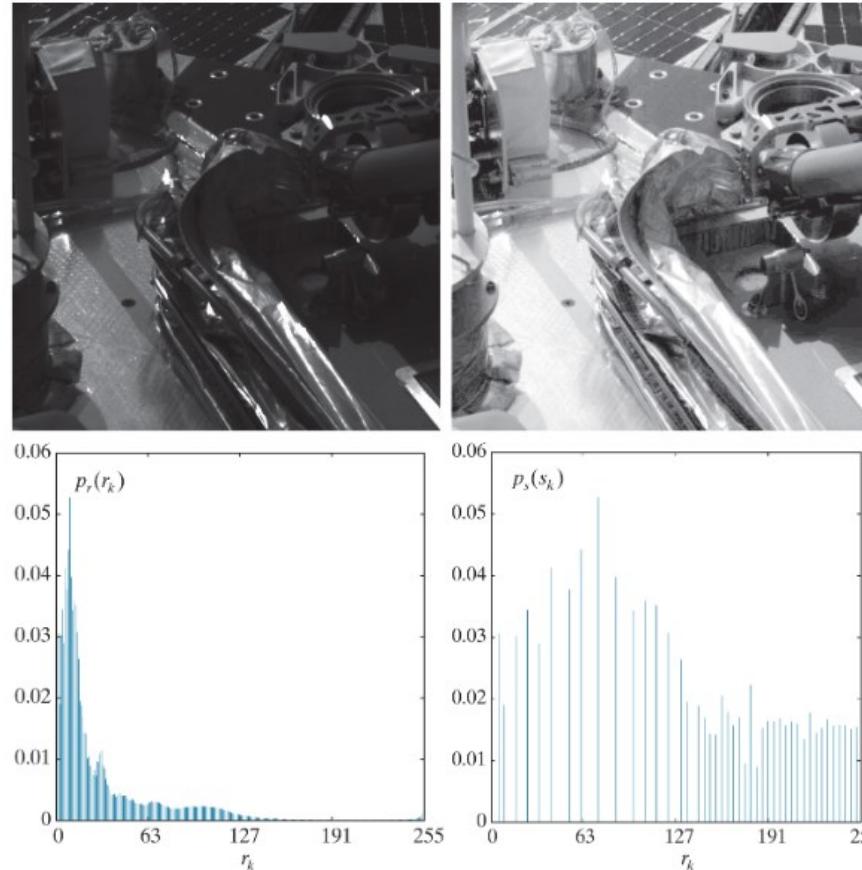
(C) Image with Equalized Histogram

Image source: Digital Image Processing by Eduardo A.B. da Silva, Gelson V. Mendonça

Histogram Equalization...

FIGURE 3.22

a b
c d



(a) Image from Phoenix Lander. (b) Result of histogram equalization. (c) Histogram of image (a). (d) Histogram of image (b). (Original image courtesy of NASA.)

Histogram Processing: Histogram Matching (Specification)

- In some applications, attempting to base enhancement on a uniform histogram is not the best approach.
- In such cases, it may be useful to specify the shape of the target histogram.
- The method used to generate a processed image that has a specified histogram is called **histogram matching** or **histogram specification**.

Histogram Matching (Specification)

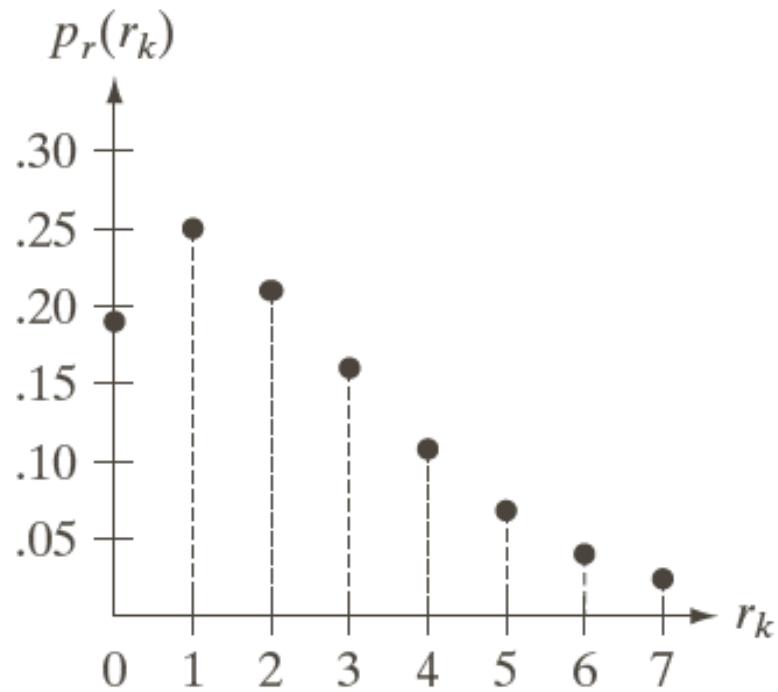
- The problem can be formalized as follows:
 - given an input image, whose pixels are distributed with probability density $p_r(r_k)$,
 - given the desired intensity distribution, $p_z(z_q)$,
 - find the transformation F , such that $z = F(r)$

$$s = T(r) = (L - 1) \int_0^r p_r(w)dw$$

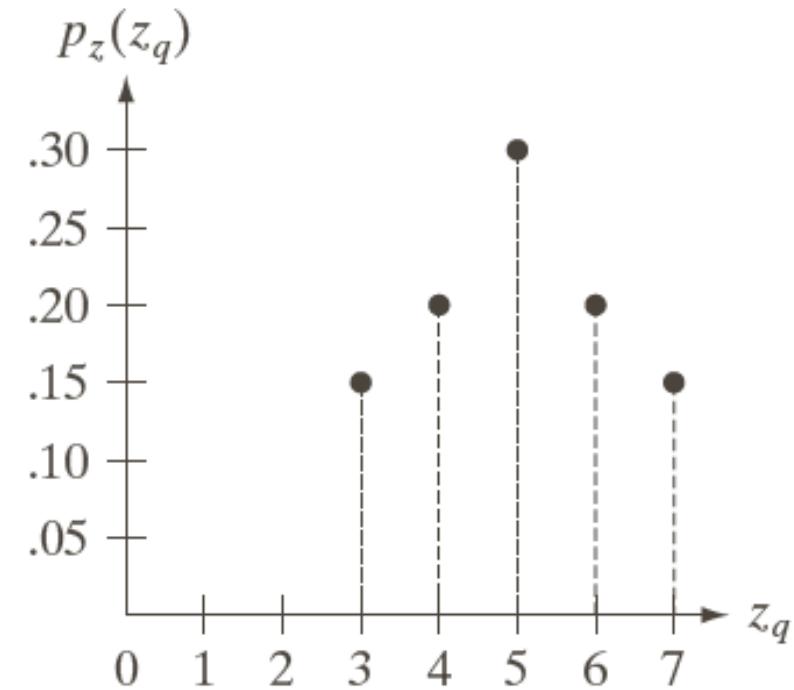
$$G(z) = (L - 1) \int_0^z p_z(t)dt = s$$

$$G(z) = s = T(r)$$
$$z = G^{-1}(T(r)), \text{ i.e., } F = G^{-1} \circ T$$

Histogram Matching: Example



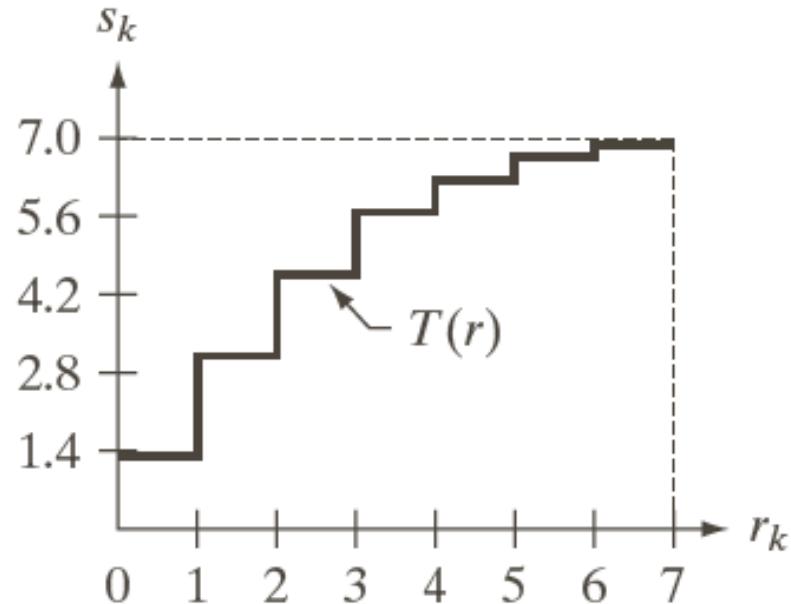
Input image histogram
(same as in last example)



Specified histogram

Histogram Matching: Example

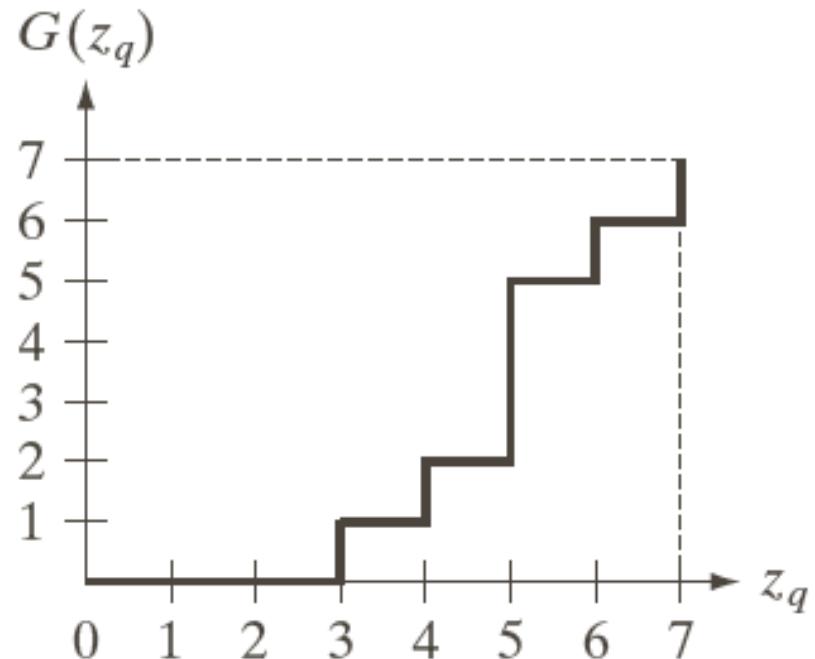
- Find $T(r)$
 - As before



$$s_0 = 1; s_1 = 3; s_2 = 5; s_3 = 6; s_4 = 6; s_5 = 7; s_6 = 7; s_7 = 7$$

Histogram Matching: Example

- Find $G(z)$ and store the mapping in a table



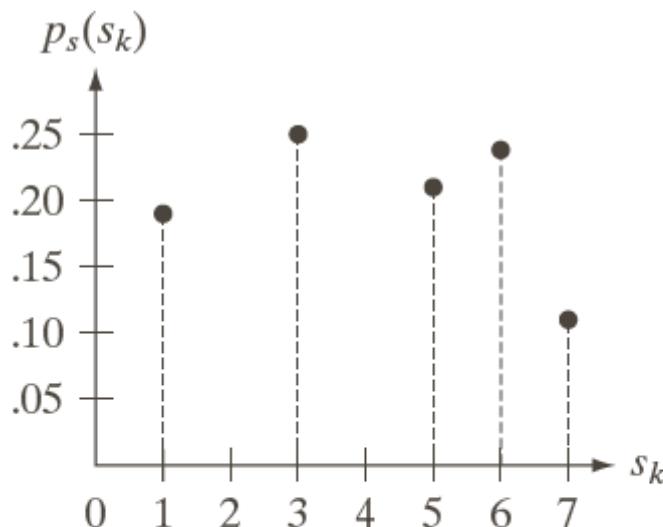
$$\begin{array}{llll} G(z_0) = 0.00 & G(z_2) = 0.00 & G(z_4) = 2.45 & G(z_6) = 5.95 \\ G(z_1) = 0.00 & G(z_3) = 1.05 & G(z_5) = 4.55 & G(z_7) = 7.00 \end{array}$$

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

Histogram Matching: Example

- For every value of s_k , use the stored values of G from last step to find the corresponding value of z_q so that $G(z_q)$ is closest to s_k .
 - When more than one value z_q of gives the same, choose the smallest value by convention.

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

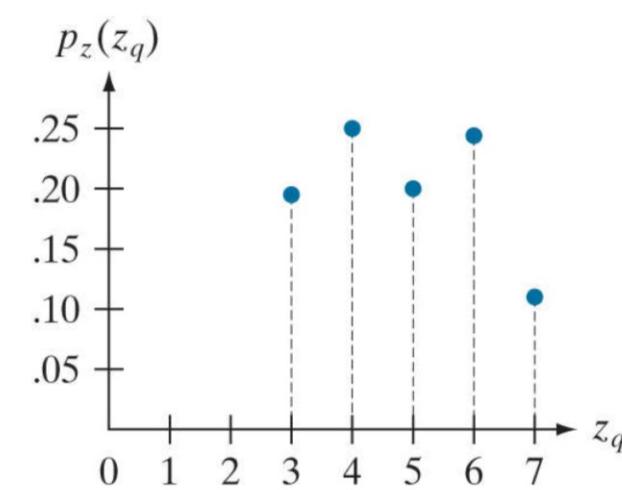
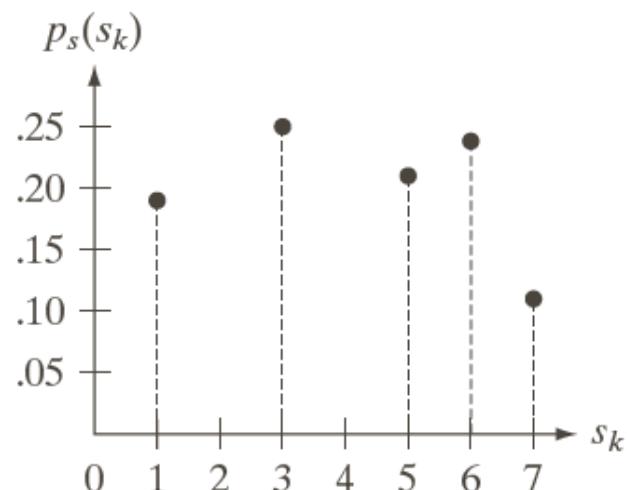


s_k	→	z_q
1	→	3
3	→	4
5	→	5
6	→	6
7	→	7

Histogram Matching (Specification)

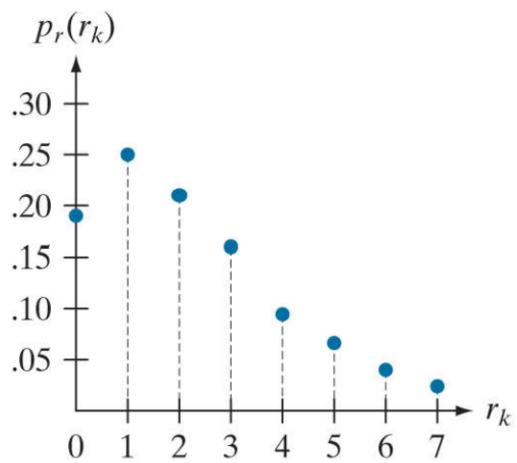
- Map each pixel in equalized image with value s_k to z_k using the mapping from the last step

s_k	\rightarrow	z_q
1	\rightarrow	3
3	\rightarrow	4
5	\rightarrow	5
6	\rightarrow	6
7	\rightarrow	7

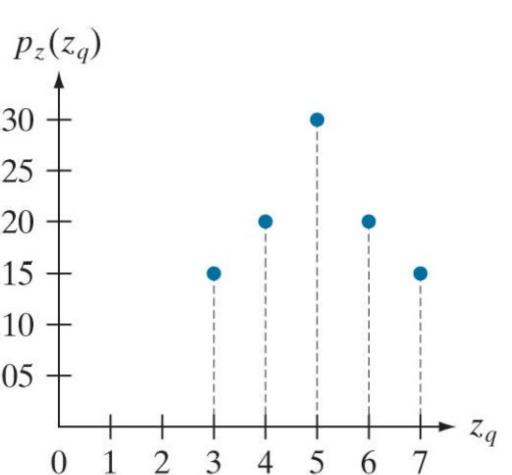


Histogram Matching (Specification)

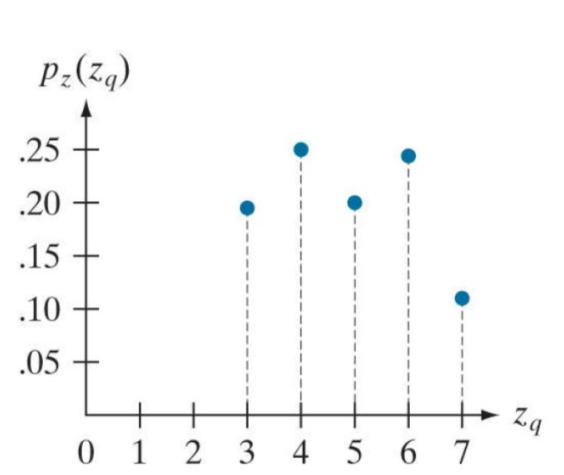
z_q	Specified $p_z(z_q)$	Actual $p_z(z_k)$
$z_0 = 0$	0.00	0.00
$z_1 = 1$	0.00	0.00
$z_2 = 2$	0.00	0.00
$z_3 = 3$	0.15	0.19
$z_4 = 4$	0.20	0.25
$z_5 = 5$	0.30	0.21
$z_6 = 6$	0.20	0.24
$z_7 = 7$	0.15	0.11



Histogram of input image

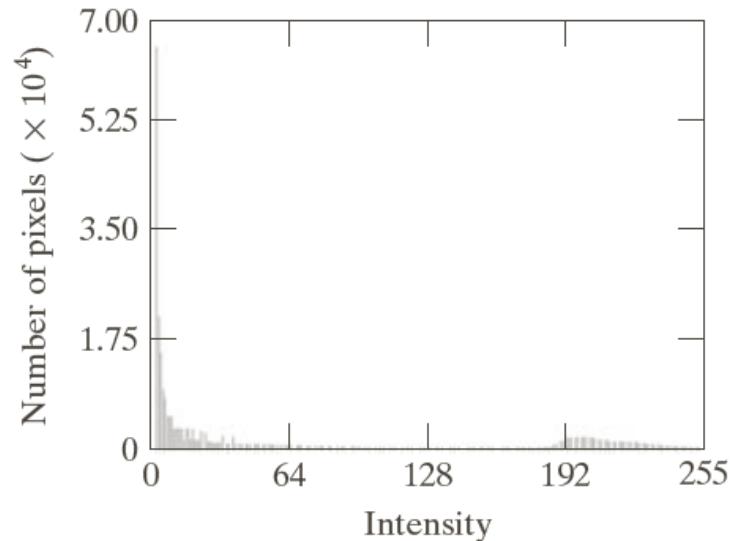


Histogram of specified image



Histogram of output image

Histogram Matching (Specification)

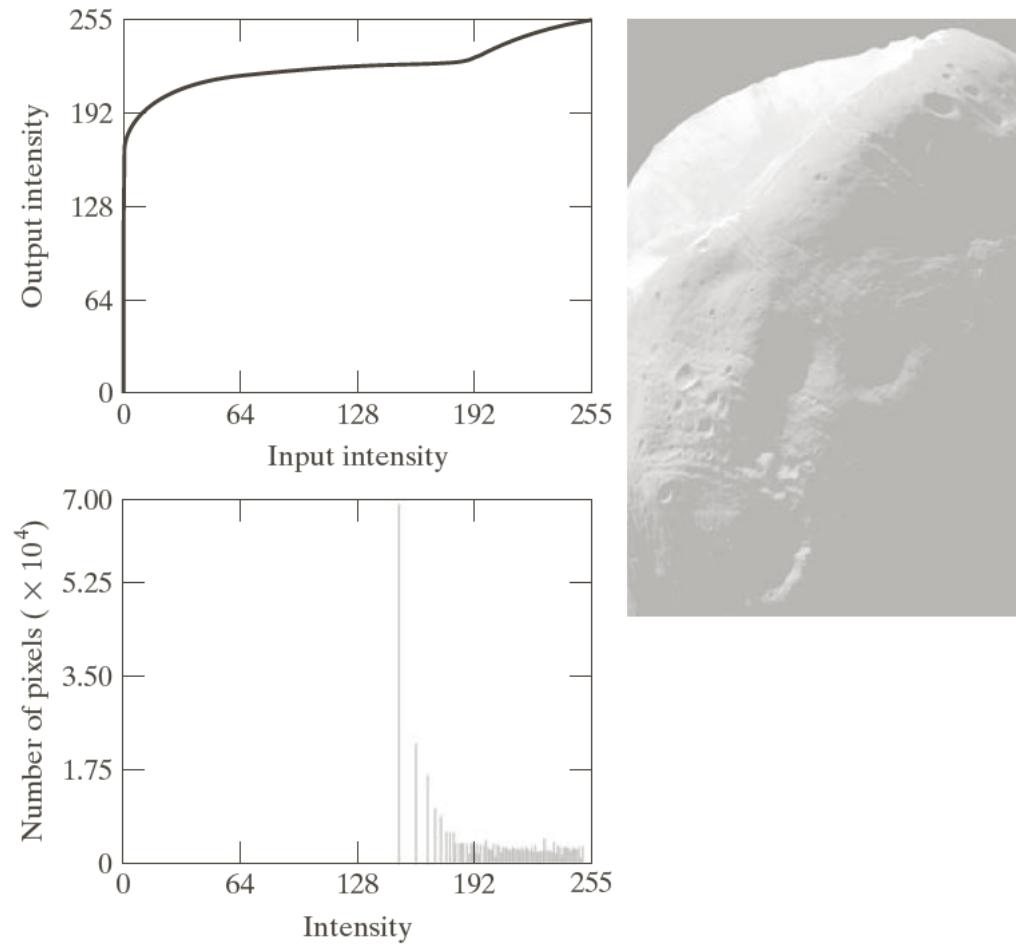


a b

FIGURE 3.23
(a) Image of the
Mars moon
Phobos taken by
NASA's *Mars
Global Surveyor*.
(b) Histogram.
(Original image
courtesy of
NASA.)

Image source: Gonzalez and Woods

Histogram Matching (Specification)

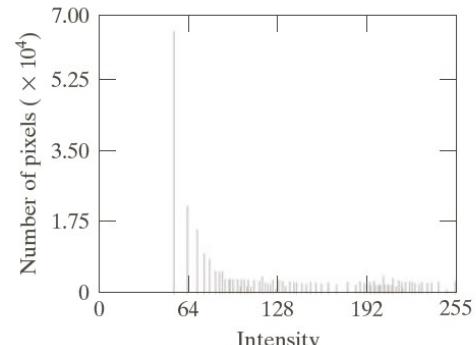
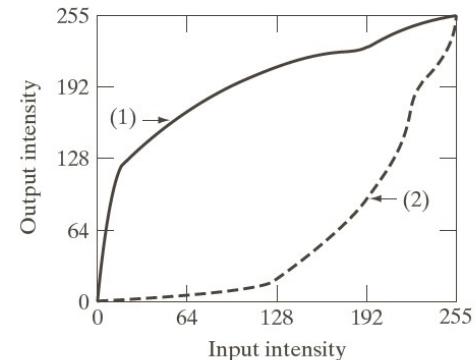
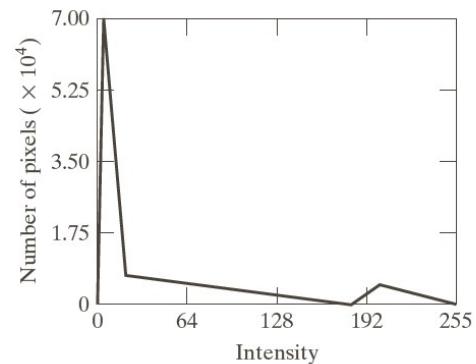


a b
c

FIGURE 3.24
(a) Transformation function for histogram equalization.
(b) Histogram-equalized image (note the washed-out appearance).
(c) Histogram of (b).

Image source: Gonzalez and Woods

Histogram Matching (Specification)



a
b
c
d

FIGURE 3.25
(a) Specified histogram.
(b) Transformations.
(c) Enhanced image using mappings from curve (2).
(d) Histogram of (c).

Image source: Gonzalez and Woods

Summary

- In this lecture, we discussed
 - Intensity Transformations (Point Operations)
 - Histogram Processing

Resources and Credits

- "Digital Image Processing (Fourth Edition)", Rafael C. Gonzalez and Richard E. Woods
 - Chapter 3

Image Processing & Pattern Recognition

Filtering

Dr. Zia Ud Din

Outline

- Basics of Spatial Filtering
- Smoothing Spatial Filters
- Sharpening Spatial Filters

Outline

- Basics of Spatial Filtering
- Smoothing Spatial Filters
- Sharpening Spatial Filters

Spatial Filtering

- The name “filter” is borrowed from frequency domain processing
 - Accepting or rejecting certain frequency components
 - E.g., a filter that passes low frequencies is called **lowpass filter**
- Filters are also called **masks, kernels, templates** and **windows**
- Spatial filters are more versatile than frequency ones
 - Can also be used for non-linear filtering

Spatial Filtering

- A spatial filter consists of
 - A neighborhood
 - A predefined operation

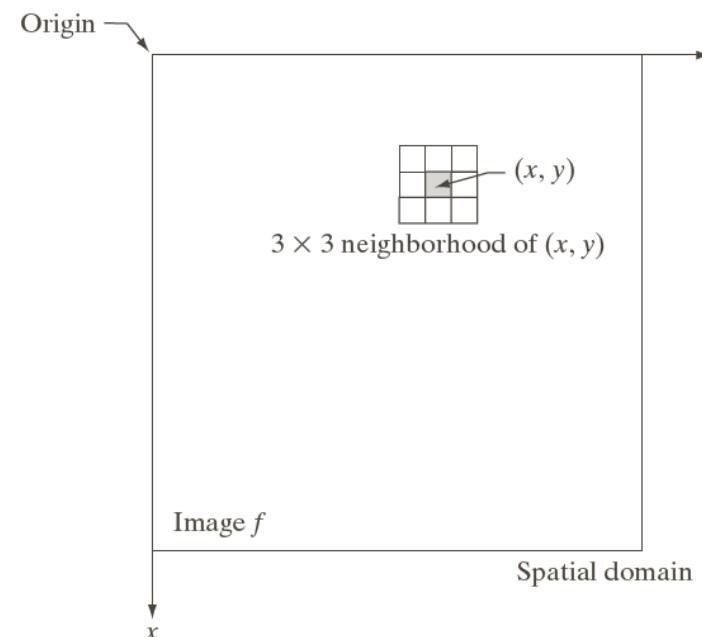


FIGURE 3.1
A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

Image source: Gonzalez and Woods

Spatial Filtering

- Filtering creates a new pixel
 - Coordinates equal to coordinates of centre of neighborhood
 - Value is the result of filtering operation
- A filtered image is generated as center of filter visits each pixel
- If the operation performed on image pixels is linear then it is called a linear filter and vice versa

Spatial Filtering

- For a mask of size $m \times n$, we assume that $m=2a+1$ and $n=2b+1$, where a and b are nonnegative integers.
- In general, linear filtering of an image f of size $M \times N$ with a filter mask w of size $m \times n$ is given by the expression:

$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

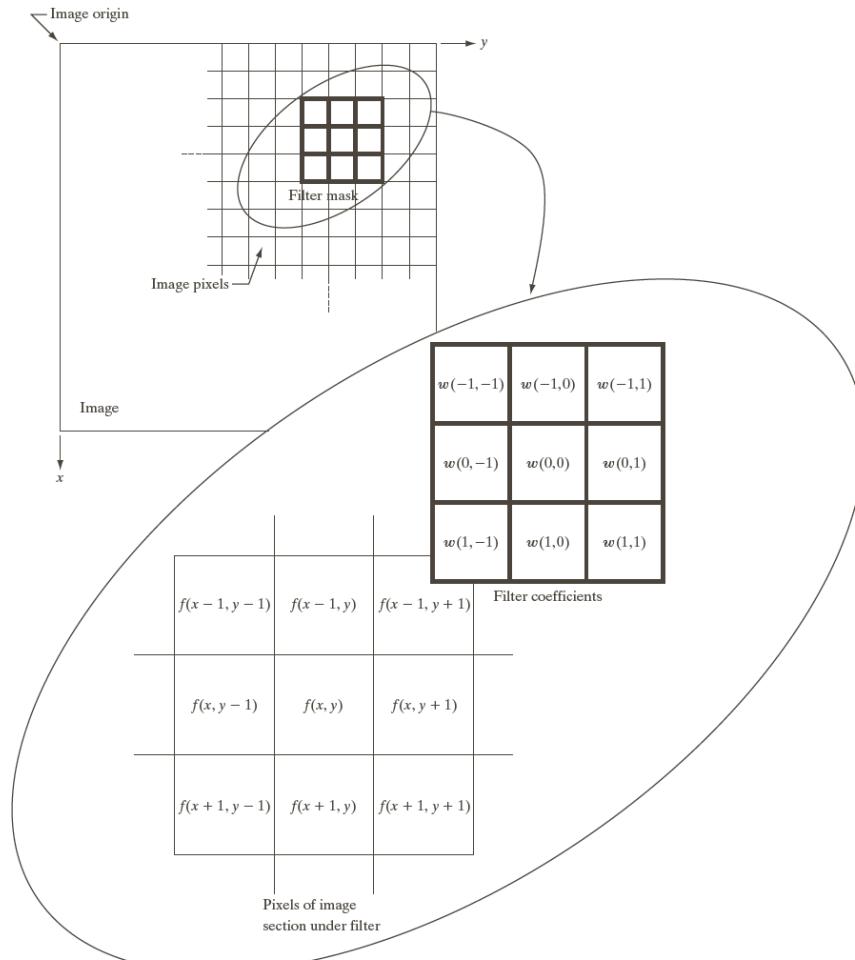


Image source: Gonzalez and Woods

Spatial Correlation and Convolution

- Correlation and convolution are closely related concepts
- Correlation is the process of moving a filter mask over the image and computing the sum of products at each location, exactly as explained earlier
- The mechanics of convolution are the same except that the filter is first rotated by 180°
- Using correlation or convolution is a matter of preference
- The terms convolution filter or convolution mask may denote either a correlation or convolution

Spatial Correlation and Convolution

		Padded f		
Origin $f(x, y)$		0 0	0 0	0 0
		$w(x, y)$	0 0 0 0 1 0	0 0
(a)		1 2 3 4 5 6 7 8 9	0 0	0 0
		(b)		
Initial position for w		1 2 3 4 5 6 7 8 9	Full correlation result	Cropped correlation result
(c)		0 0	0 0	0 0 0 0 0 0 9 8 7 0 0 6 5 4 0 0 3 2 1 0 0 0 0 0 0
		0 0 0 0 1 0	0 0 0 9 8 0 0 0 7 0 0 0 0 6 5 0 0 0 3 2 0 0 0 0 0	0 0 0 0 0 0 9 8 7 0 0 6 5 4 0 0 3 2 1 0 0 0 0 0 0
		0 0	0 0	0 0
		(d)		
Rotated w		9 8 7 6 5 4 3 2 1	Full convolution result	Cropped convolution result
(f)		0 0	0 0	0 0 0 0 0 0 1 2 3 0 0 4 5 6 0 0 7 8 9 0 0 0 0 0 0
		0 0 0 0 1 0	0 0 0 1 2 0 0 0 4 5 0 0 0 7 8 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 1 2 3 0 0 4 5 6 0 0 7 8 9 0 0 0 0 0 0
		0 0	0 0	0 0
		(g)		
		0 0	0 0	0 0
		(h)		

FIGURE 3.30
 Correlation
 (middle row) and
 convolution (last
 row) of a 2-D
 filter with a 2-D
 discrete, unit
 impulse. The 0s
 are shown in gray
 to simplify visual
 analysis.

Outline

- Basics of Spatial Filtering
- Smoothing Spatial Filters
- Sharpening Spatial Filters

Smoothing Spatial Filters

- Smoothing filters are used for blurring an image
- Blurring is used in preprocessing steps, such as
 - Removal of small details from an image prior to (large) object extraction
 - Bridging of small gaps in lines or curves.
- Blurring can also be used for **noise reduction**.

Smoothing Linear Filters

- The output (response) of a smoothing, linear spatial filter is simply the average of the pixels contained in the neighborhood.
- These filters sometimes are called **averaging filters** or **lowpass filters**.
- This averaging process results in an image with reduced "sharp" transitions in gray levels.

Smoothing Linear Filters

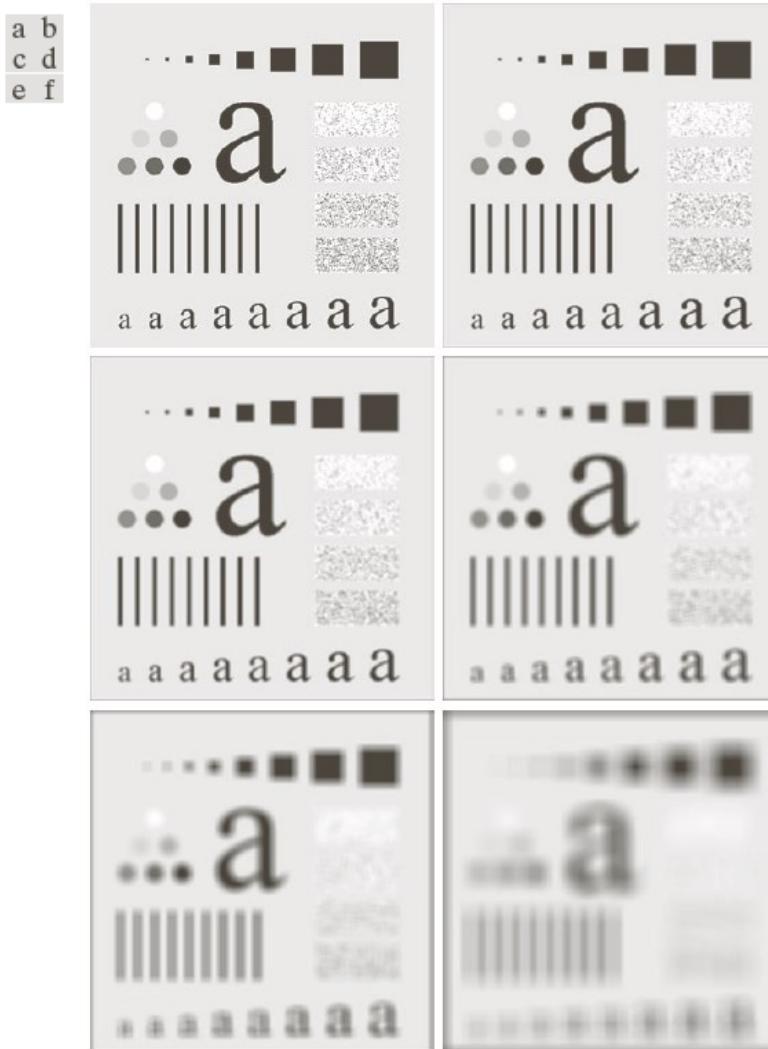
- Because random noise typically consists of sharp transitions in gray levels, the most obvious application of smoothing is noise reduction.
- On the other hand, averaging filters have the undesirable side effect that they blur edges.
- A major use of averaging filters is in the reduction of “irrelevant” detail in an image.
 - By “irrelevant” we mean pixel regions that are small with respect to the size of filter mask.

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$
$$\frac{1}{16} \times \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

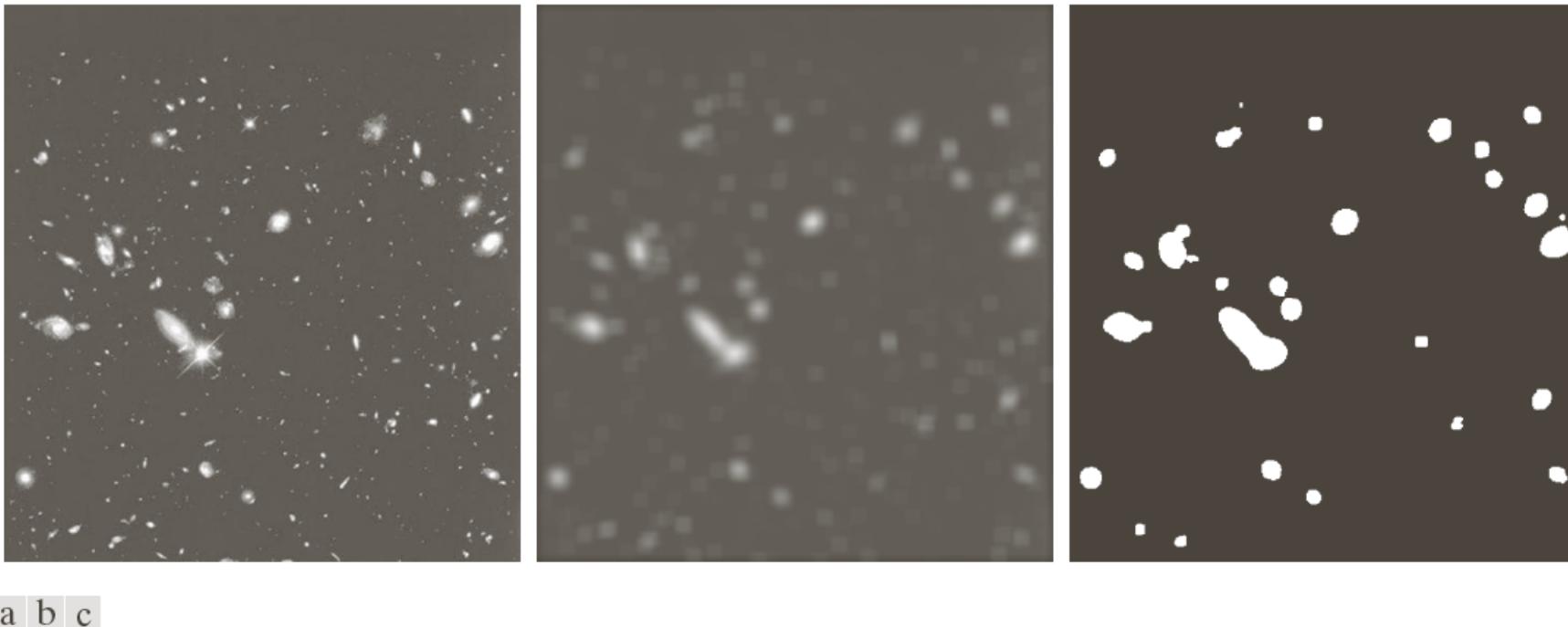
$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

Smoothing Linear Filters

FIGURE 3.33 (a) Original image, of size 500×500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35 , respectively. The black squares at the top are of sizes $3, 5, 9, 15, 25, 35, 45$, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50×120 pixels.



Smoothing Linear Filters



a b c

FIGURE 3.34 (a) Image of size 528×485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15×15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

Image source: Gonzalez and Woods

Order-Statistic (Nonlinear) Filters

- Order-statistics filters are nonlinear spatial filters
 - The response is based on ordering (ranking) the pixels contained in the sub-image
 - Then replacing the value of the center pixel with the value determined by the ranking result
- The best-known example in this category is the median filter
 - Though max filter, min filter, etc are also used
- Median filters are particularly effective in the presence of impulse noise, also called salt-and-pepper noise

Order-Statistic (Nonlinear) Filters

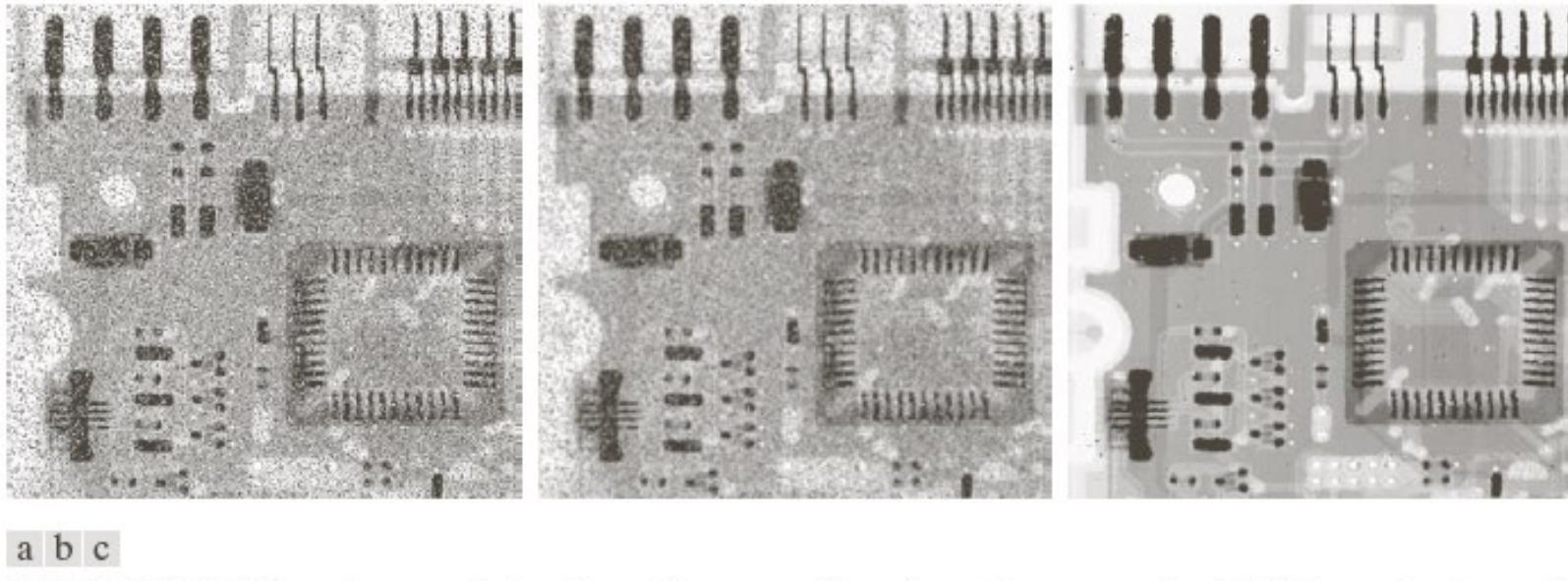


FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Image source: Gonzalez and Woods

Outline

- Basics of Spatial Filtering
- Smoothing Spatial Filters
- Sharpening Spatial Filters

Sharpening Spatial Filters

- The principal objective of sharpening is to highlight **fine detail** in an image or to enhance detail that has been blurred
- Sharpening could be accomplished by spatial **differentiation**
 - Image differentiation enhances edges and other discontinuities (such as noise)

Sharpening Spatial Filters

- Foundation
- The Gradient
- The Laplacian
- Unsharp Masking and Highboost Filtering

Foundation

- To simplify the explanation, we focus attention on one-dimensional derivatives.
- We are interested in the behavior of these derivatives in
 - Areas of constant gray level (flat segments),
 - At the onset and end of discontinuities (step and ramp discontinuities),
 - Along gray-level ramps

Foundation

- We require that any definition we use for a first derivative
 1. Must be zero in flat segments (areas of constant gray-level values);
 2. Must be nonzero at the onset of a gray-level step or ramp;
 3. Must be nonzero along ramps.
- Similarly, any definition of a second derivative
 1. Must be zero in flat areas;
 2. Must be nonzero at the onset and end of a gray-level step or ramp;
 3. Must be zero along ramps of constant slope

Foundation

- A basic definition of the first-order derivative of a one-dimensional function $f(x)$ is the difference:

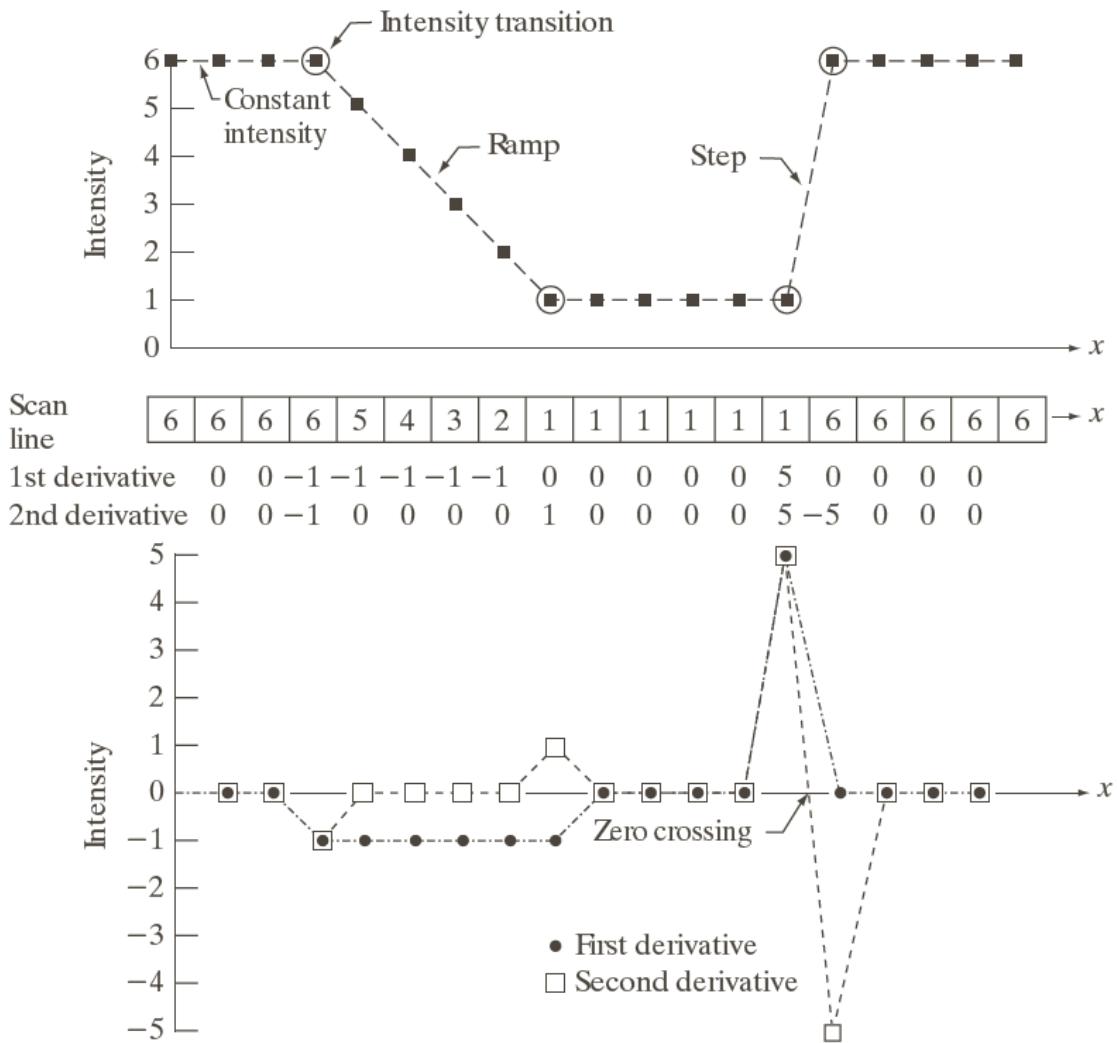
$$\frac{\partial f}{\partial x} = f(x + 1) - f(x).$$

- We define a second-order derivative as the difference:

$$\frac{\partial^2 f}{\partial x^2} = f(x + 1) + f(x - 1) - 2f(x).$$

- It is easily verified that these two definitions satisfy the conditions stated previously.

Foundation



a
b
c

FIGURE 3.36
Illustration of the first and second derivatives of a 1-D digital function representing a section of a horizontal intensity profile from an image. In (a) and (c) data points are joined by dashed lines as a visualization aid.

Image source: Gonzalez and Woods

1st Order Derivative for Image Sharpening: The Gradient

- First derivatives in image processing are implemented using the magnitude of the gradient.

$$\nabla f \equiv \text{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$M(x, y) = \| \nabla f \| = \text{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

- The approach used is defining a discrete formulation of the first-order derivative
 - And then constructing a filter mask based on that formulation.

The Gradient

- The most used gradient masks are Prewitt and Sobel.

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

Gx

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

Gy

Prewitt filter

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

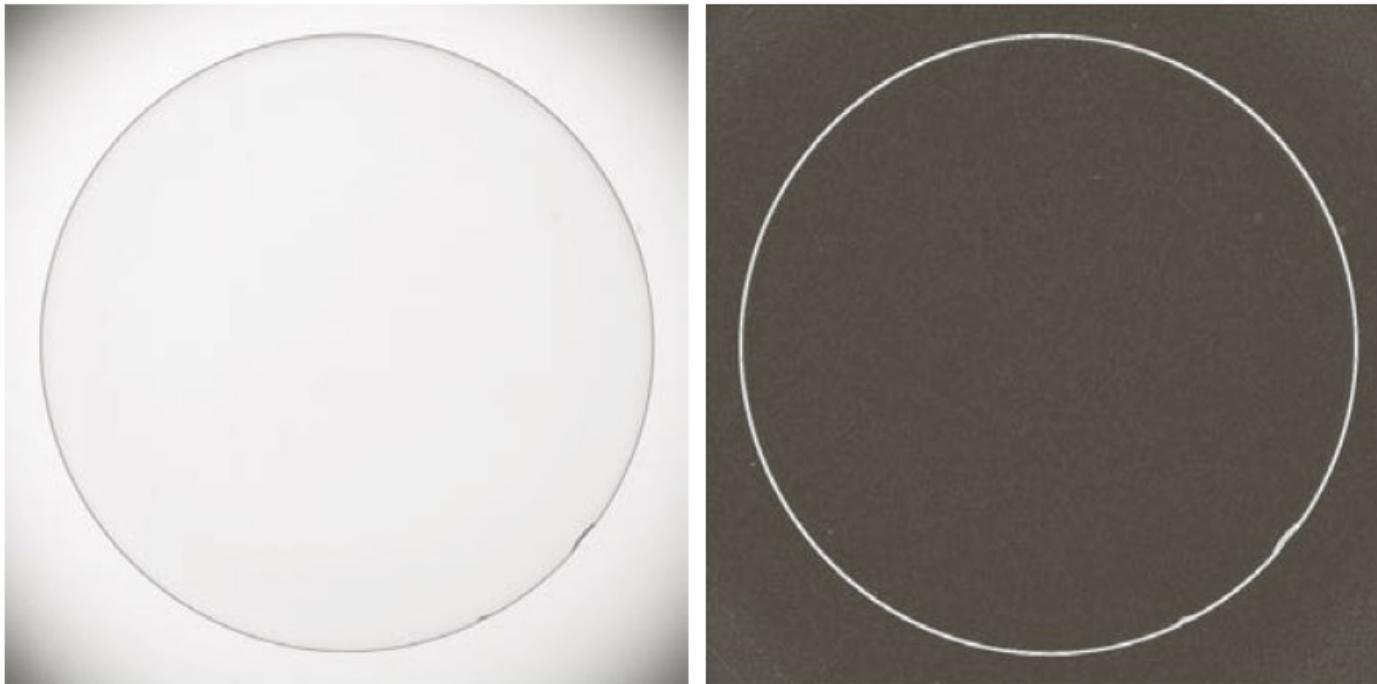
Gx

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

Gy

Sobel filter

The Gradient



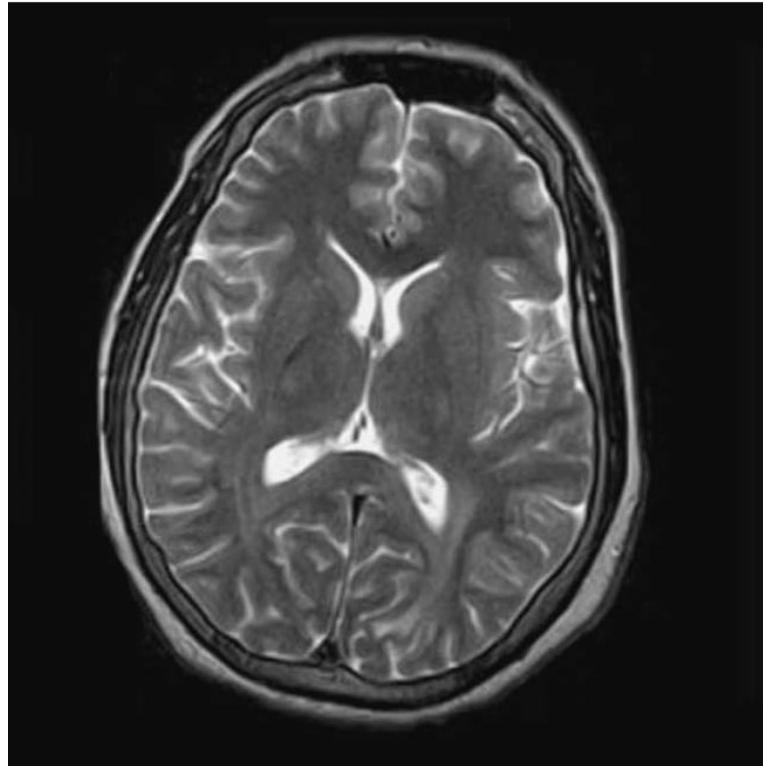
a b

FIGURE 3.42

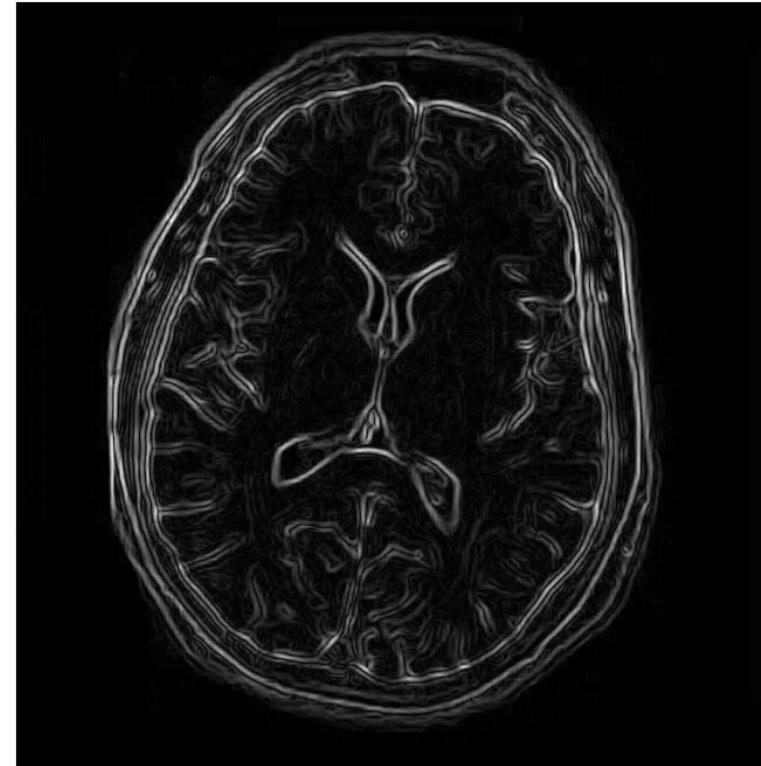
(a) Optical image of contact lens (note defects on the boundary at 4 and 5 o'clock).
(b) Sobel gradient.
(Original image courtesy of Pete Sites, Perceptics Corporation.)

Image source: Gonzalez and Woods

The Gradient



T2-weighted spin echo image of the head



Gradient magnitude image using Prewitt filter

Image source: Digital Image Processing for Medical Applications by Geoff Dougherty

2nd Order Derivative for Image Sharpening: The Laplacian

- Edges in digital images often are ramp-like transitions in intensity
 - The first derivative of the image would result in thick edges
 - The second derivative would produce a double edge one pixel thick, separated by zeros.

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

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

The Laplacian

□ Most common Laplacian filters

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

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

The Laplacian

- Laplacian filters will produce images with grayish edge lines superimposed on a dark background.
- Background features can be "recovered" by adding the Laplacian image to the original.

$$g(x, y) = f(x, y) + c[\nabla^2 f(x, y)]$$

- Use $c=-1$ when using filter with negative center value and $c=1$ when using filter with positive center value

The Laplacian



a b
c d

- (a) Blurred image of the North Pole of the moon.
- (b) Laplacian image obtained using the kernel with center -4 .
- (c) Image sharpened using equation on last slide
- (d) Image sharpened using the same procedure, but with the kernel with center -8 (Original image courtesy of NASA.)

Image source: Gonzalez and Woods

Unsharp Masking and Highboost Filtering

- The idea of **unsharp masking** is get a sharpened image by subtracting an unsharp (smoothed) version of the image from original one.
- It consists of the following steps
 1. Blur the original image.
 2. Subtract the blurred image from the original (the result is called mask)
 3. Add the mask to the original

Unsharp Masking and Highboost Filtering

$$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$$

$$g(x, y) = f(x, y) + kg_{\text{mask}}(x, y)$$

- k is non-negative weight used for generality
 - When k=1, we have **unsharp masking**
 - When k>1, the process is called **highboost filtering**
 - Choosing k<1 de-emphasizes the contribution of mask

Unsharp Masking and Highboost Filtering

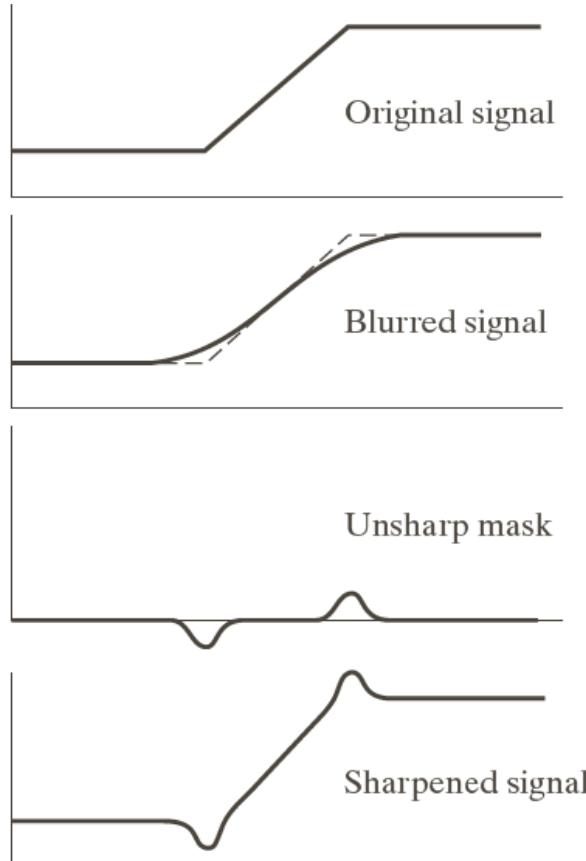


FIGURE 3.39 1-D illustration of the mechanics of unsharp masking.
(a) Original signal.
(b) Blurred signal with original shown dashed for reference.
(c) Unsharp mask.
(d) Sharpened signal, obtained by adding (c) to (a).

Image source: Gonzalez and Woods



a
b
c
d
e

FIGURE 3.40
(a) Original image.
(b) Result of blurring with a Gaussian filter.
(c) Unsharp mask.
(d) Result of using unsharp masking.
(e) Result of using highboost filtering.

Summary

- In this lecture, we learned
 - Basics of Spatial Filtering
 - Smoothing Spatial Filters
 - Sharpening Spatial Filters

Resources and Credits

- "Digital Image Processing (Fourth Edition)", Rafael C. Gonzalez and Richard E. Woods, Prentice Hall, 2017
 - Chapter 3

Image Processing & Pattern Recognition

Edge Detection

Dr. Zia Ud Din

Outline

- Introduction
- Gradient Operators
- Effect of Noise
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Outline

- **Introduction**
- Gradient Operators
- Effect of Noise
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Introduction

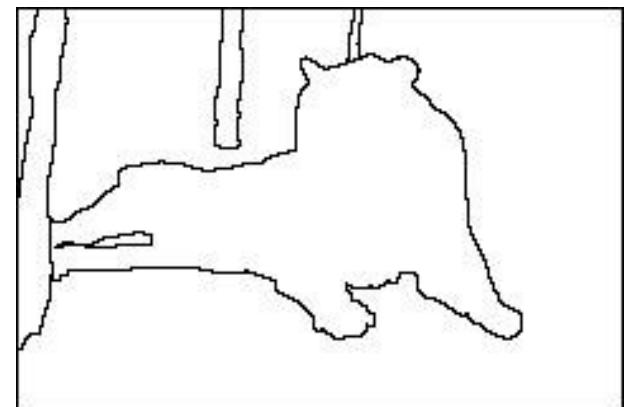
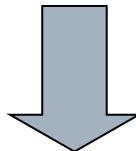
- **Goal:** Identify sudden changes (discontinuities) in an image
 - Intuitively, most semantic and shape information from the image can be encoded in the edges
 - More compact than pixels
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)



Source: D. Lowe

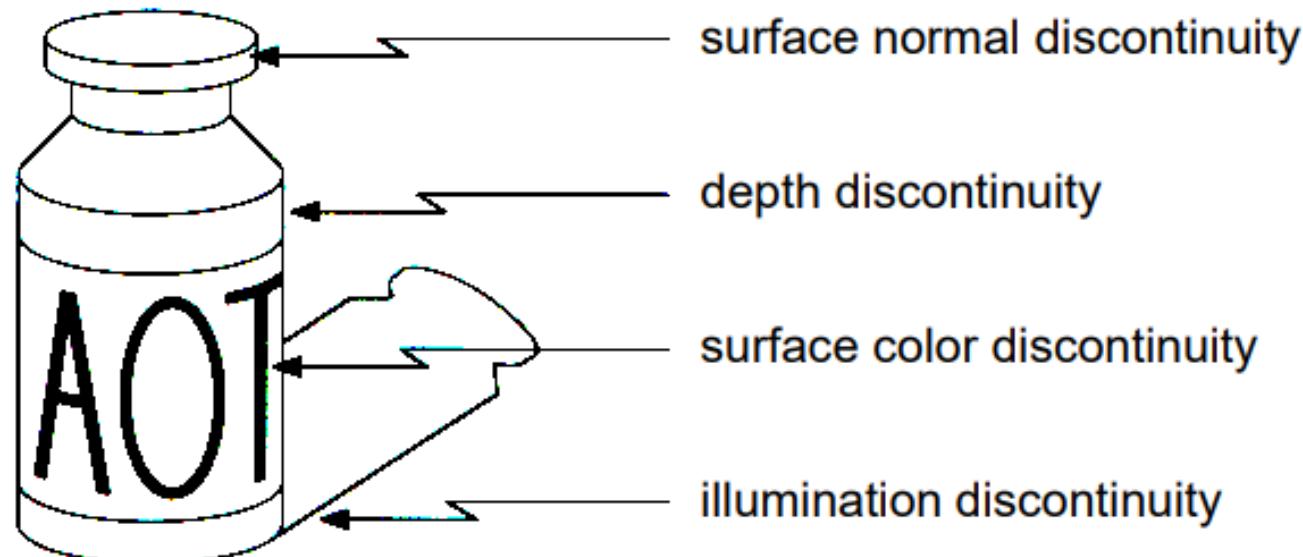
Why Edge Detection?

- Reduce dimensionality of data
- Preserve content information
- Useful in applications such as:
 - object detection
 - structure from motion
 - tracking



Introduction

- Edges are caused by a variety of factors



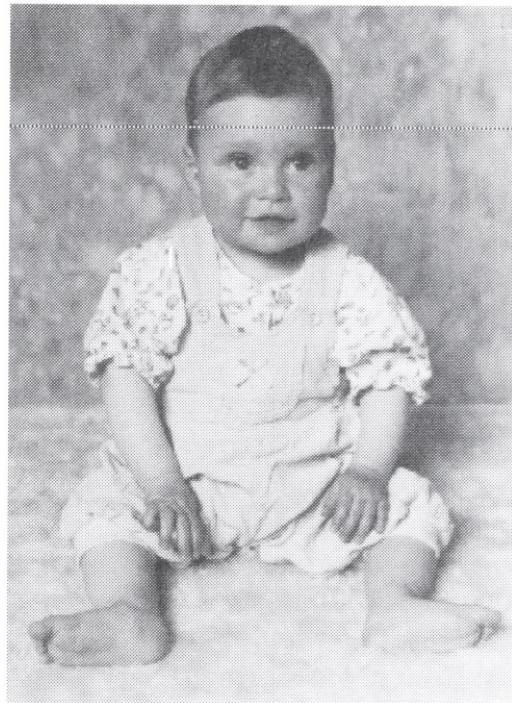
Source: Steve Seitz

Introduction...

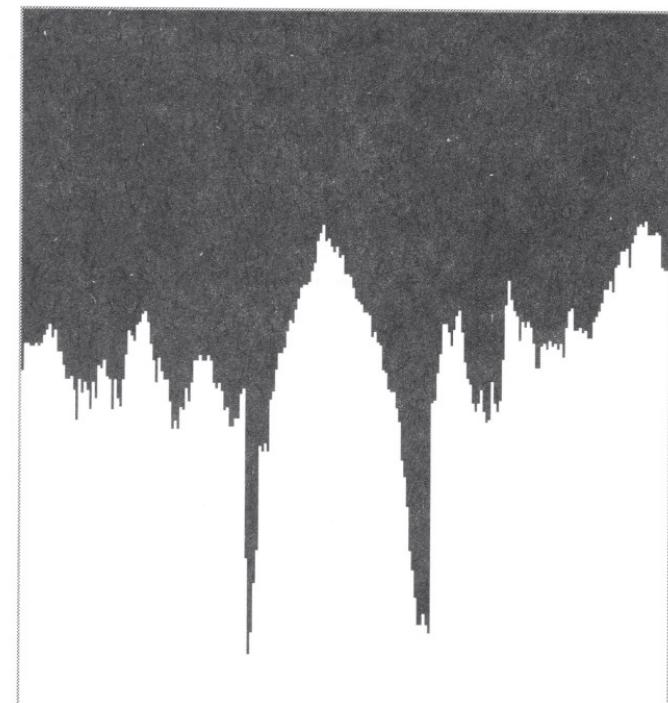
- What are features of
 - A car
 - A cell phone
- What are the commonalities & differences in features between
 - A living room vs an office room
 - Walls
 - Chairs / Sofas
 - Tables
 - TV / Projector
 - The setup is different
- Features are
 - Element(s) or group of elements that are related to an object in a distinct fashion.

Introduction...

- Mostly composition of distinct features is based on high contrasts regions



(a)



(b)

Figure 4.1 (a) A 325×237 -pixel image, with scanline $i = 56$ highlighted. (b) The intensity profile along the highlighted scanline. Notice how the main intensity variations indicate the borders of the hair region along the scanline.

Introduction...

- The most important feature extraction methods are **edge** detection, and **corner** detection.
 - Sharpening filters identify regions of high contrast in an image.
 - These regions can be single pixels, or larger neighborhoods of adjacent pixels.
 - In edge detection, we wish to group these adjacent pixels into continuous curves that represent the boundaries of objects.



Outline

- Introduction
- Gradient Operators
- Effect of Noise
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Edge Detectors

- Gradient Operators
 - Prewitt
 - Sobel
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Gradient Operators

□ General Procedure

- Smooth image I to reduce noise
- Convolve I with orthogonal masks (horizontal G_1 and vertical G_2) to obtain I_1 and I_2
- Estimate the gradient magnitude image as:

$$G(i, j) = \sqrt{I_1^2(i, j) + I_2^2(i, j)}$$

- Thresholding: Mark all pixels $I(i, j)$ as edges if $G(i, j) > \tau$

Gradient Operators...

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

Prewitt

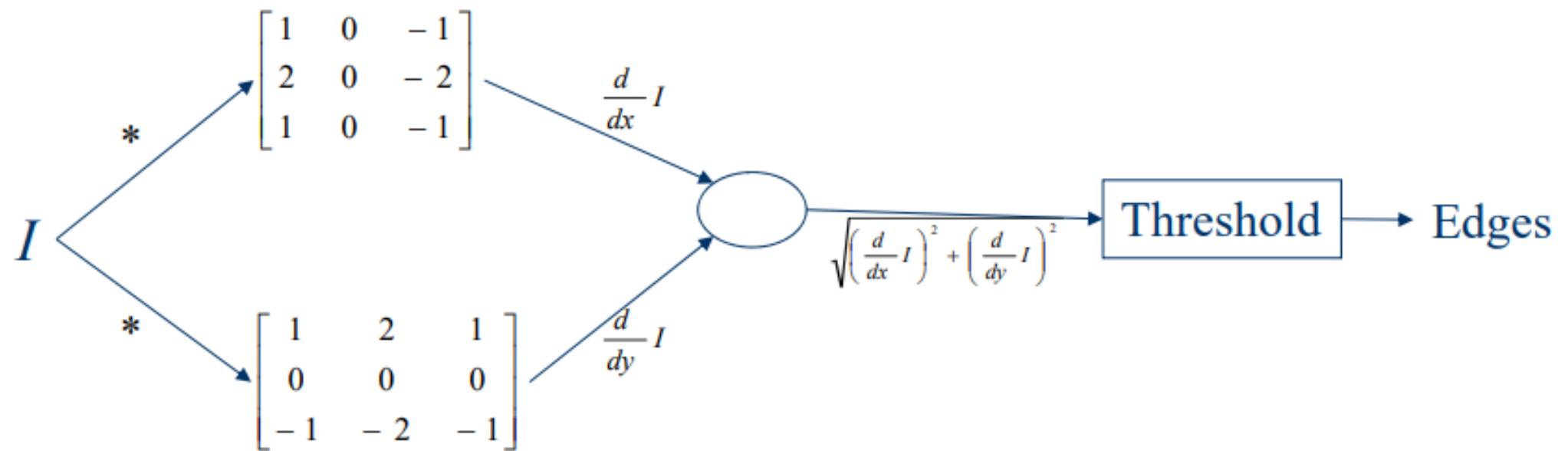
-1	0	1
-1	0	1
-1	0	1

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

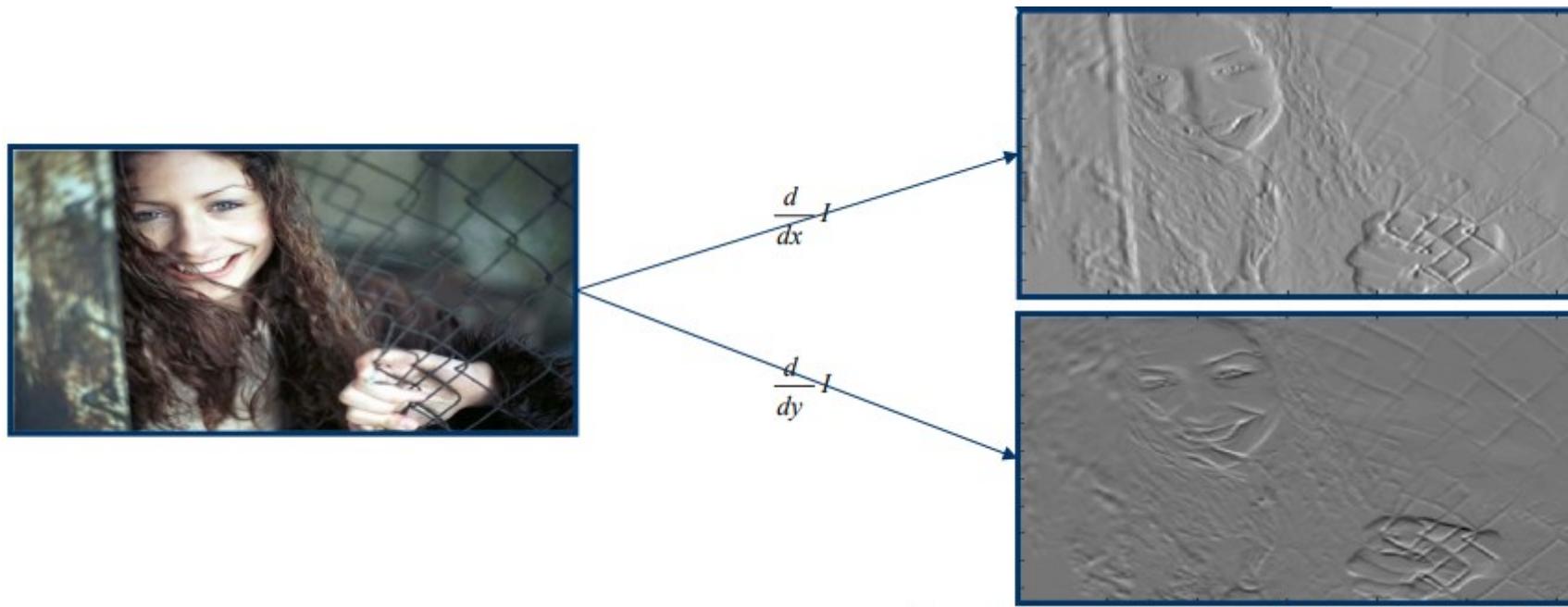
Sobel

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

Sobel Edge Detector



Sobel Edge Detector



Source: Alper Yilmaz, Mubarak Shah

Sobel Edge Detector



$$\Delta = \sqrt{\left(\frac{d}{dx} I\right)^2 + \left(\frac{d}{dy} I\right)^2}$$

$$\Delta \geq \text{Threshold} = 100$$



Source: Alper Yilmaz, Mubarak Shah

Sobel Edge Detector

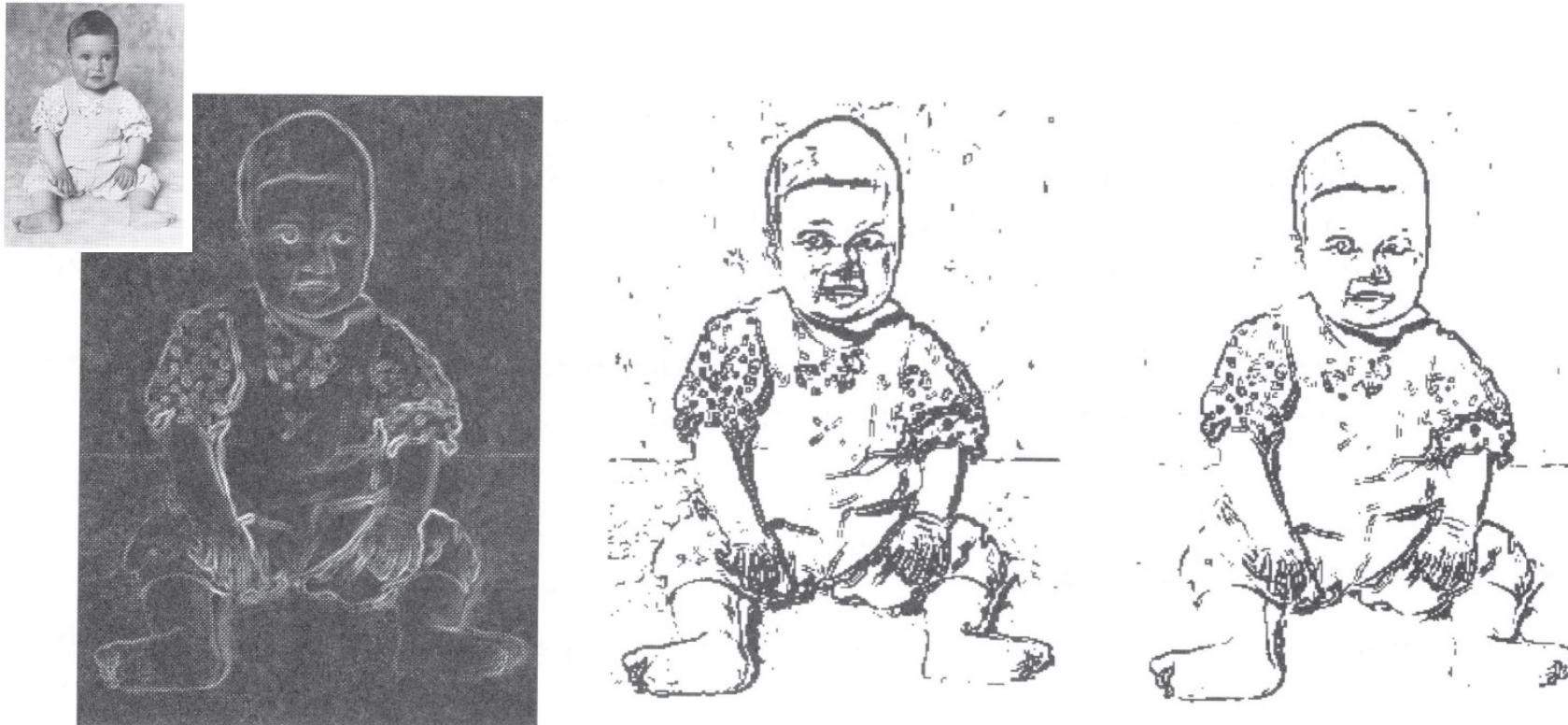
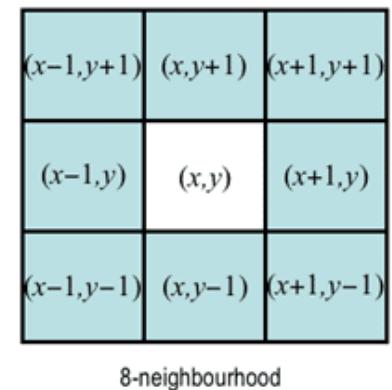
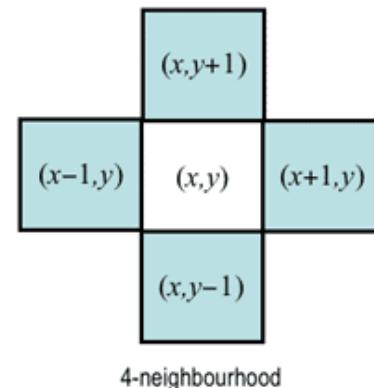


Figure 4.7 Left: output of Sobel edge enhancer run on Figure 4.1. Middle: edges detected by thresholding the enhanced image at 35. Right: same, thresholding at 50. Notice that some contours are thicker than one pixel (compare with Figure 4.5).

Group Adjacent Pixels

- Thresholding gives binary images
- Pixels on the edges represent 1, otherwise 0.
- How to find individual edge features?
 - Group adjacent "ON" pixels

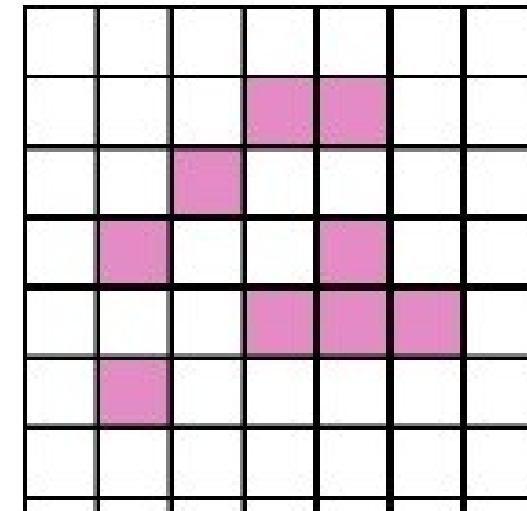
- Connected Components Algorithm
 - 4 connected
 - 8 connected



Group Adjacent Pixels

□ The algorithm is:

1. Start from the first pixel in the image. Set "curlab" (short for "current label") = 1.
2. If this pixel is a foreground pixel and it is not already labelled, then give it the label "curlab" and add it as the first element in a queue, then go to (3). If it is a background pixel, then repeat (2) for the next pixel in the image.
3. Pop out an element from the queue, and look at its neighbours (based on any type of connectivity). If a neighbour is a foreground pixel and is not already labelled, give it the "curlab" label and add it to the queue. Repeat (3) until there are no more elements in the queue.
4. Go to (2) for the next pixel in the image and increment "curlab" by 1.

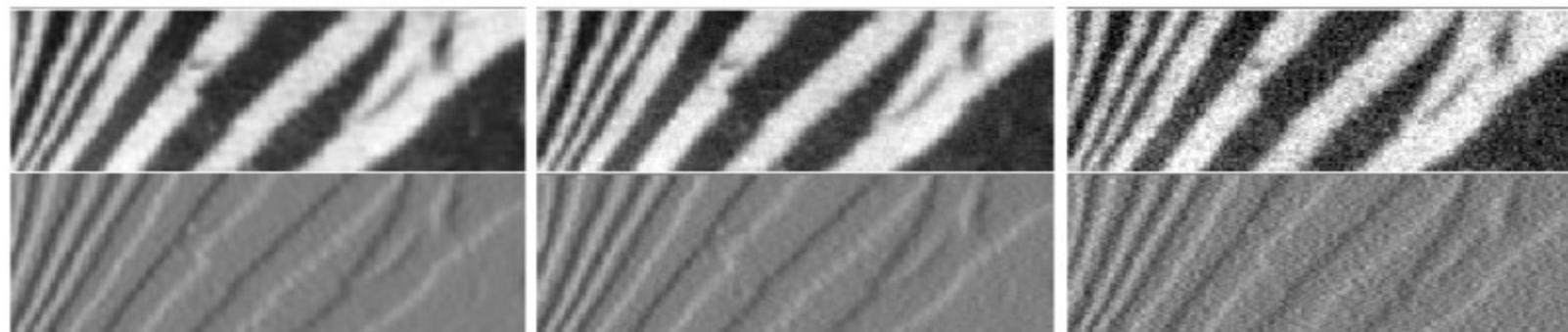


Outline

- Introduction
- Gradient Operators
- Effect of Noise
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Effect of Noise

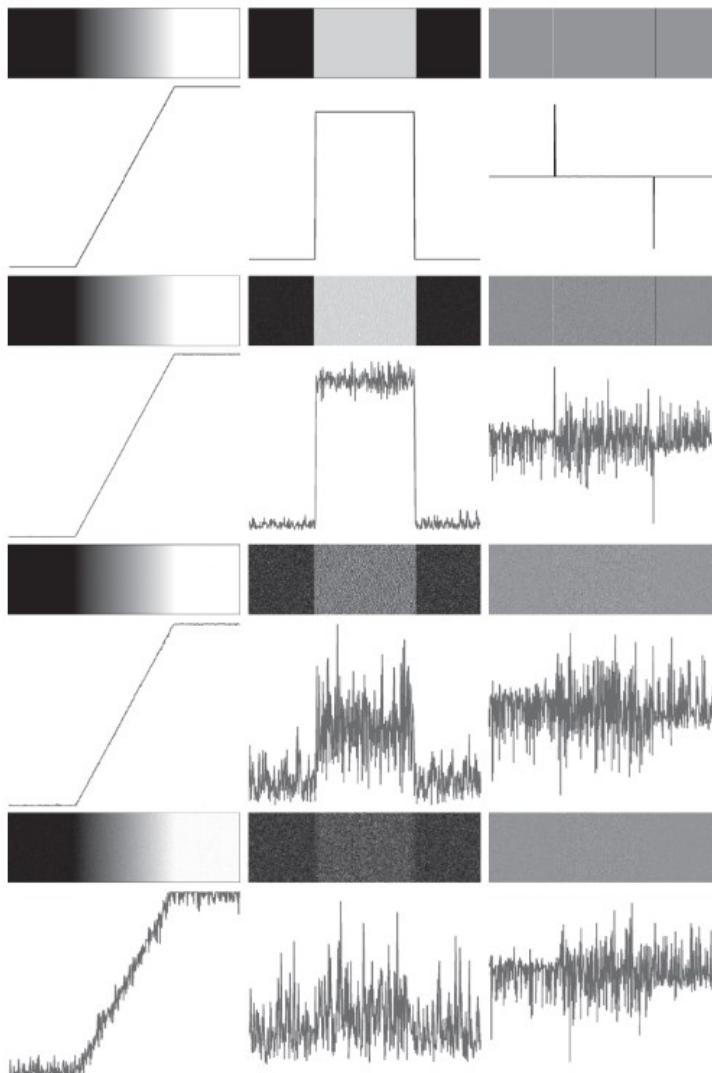
- Derivatives are strongly affected by noise
 - Obvious reason: image noise results in pixels that look very different from their neighbors
 - The larger the noise is the stronger the response



Zero mean additive Gaussian noise

Source: Alper Yilmaz, Mubarak Shah

Effect of Noise...



First column: 8-bit images with values in the range [0, 255], and intensity profiles of a ramp edge corrupted by Gaussian noise of zero mean and standard deviations of 0.0, 0.1, 1.0, and 10.0 intensity levels, respectively. Second column: First-derivative images and intensity profiles. Third column: Second-derivative images and intensity profiles.

Gaussian Smoothing

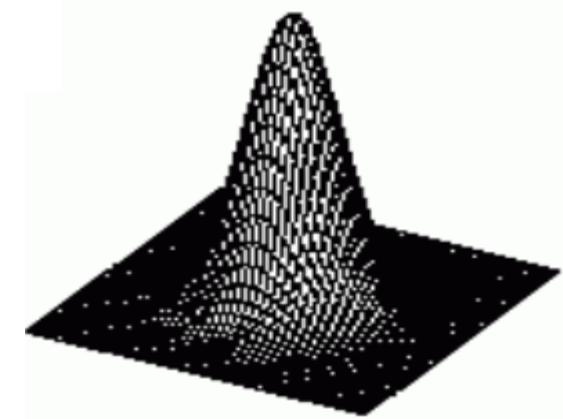
- 1D Gaussian filters

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

$$G(y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{y^2}{2\sigma^2}}$$

- 2D Gaussian filter

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



- It's a common practice to ignore the constant term

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian Smoothing...

- A sample 1D Gaussian filters

0.0044	0.054	0.242	0.3991	0.242	0.054	0.0044
--------	-------	-------	--------	-------	-------	--------

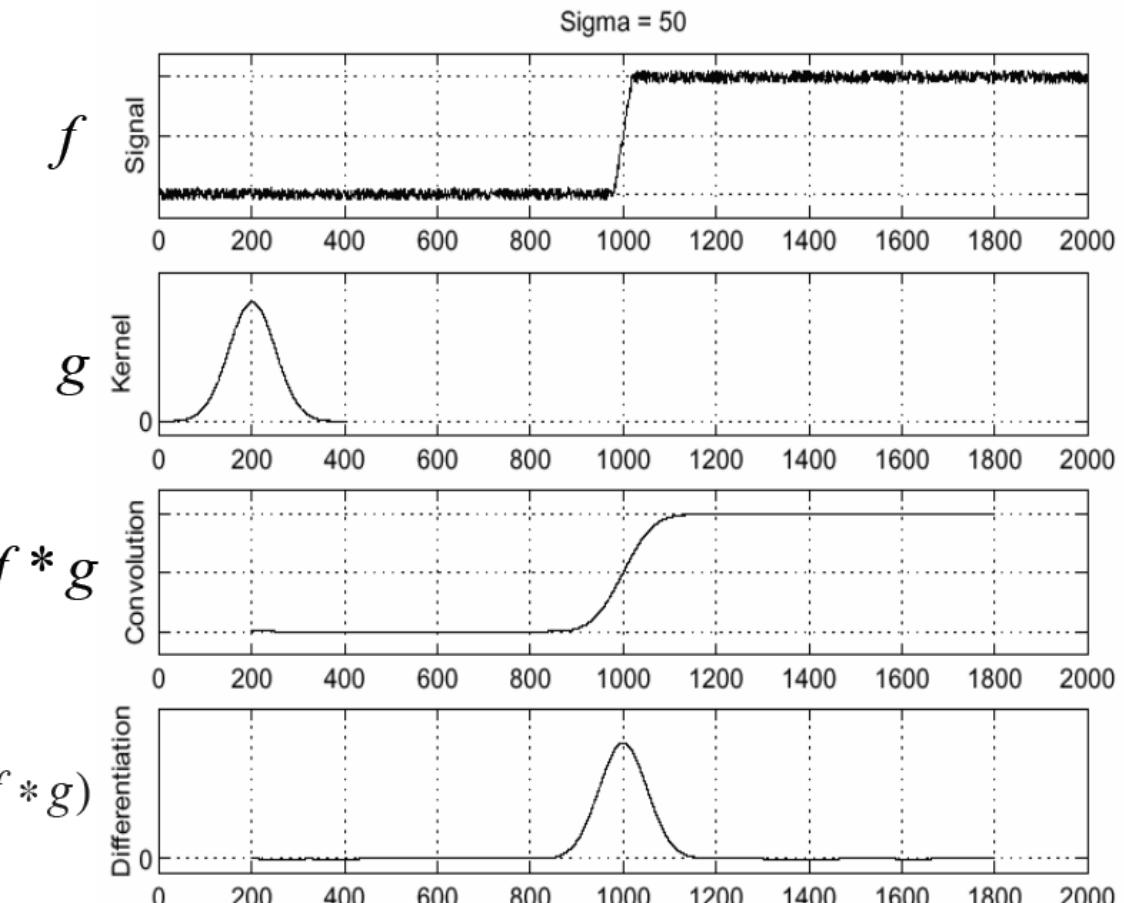
- A sample 2D Gaussian filter

0.0044	0.054	0.242	0.3991	0.242	0.054	0.0044
0.054	0.0656	0.2958	0.4877	0.2958	0.0656	0.054
0.242	0.2958	0.1332	0.2195	0.1332	0.2958	0.242
0.3991	0.4877	0.2195	0.3617	0.2195	0.4877	0.3991
0.242	0.2958	0.1332	0.2195	0.1332	0.2958	0.242
0.054	0.0656	0.2958	0.4877	0.2958	0.0656	0.054
0.0044	0.054	0.242	0.3991	0.242	0.054	0.0044

- The filters are typically normalized so that they sum to 1

Gaussian Smoothing...

- What is to be done?
 - Smooth the image first
 - Neighboring pixels look alike
 - Pixel along an edge look alike
 - Image smoothing should help
 - Force pixels different to their neighbors (possibly noise) to look like
- To find edges, look for the peaks in $\frac{d}{dx} (f * g)$

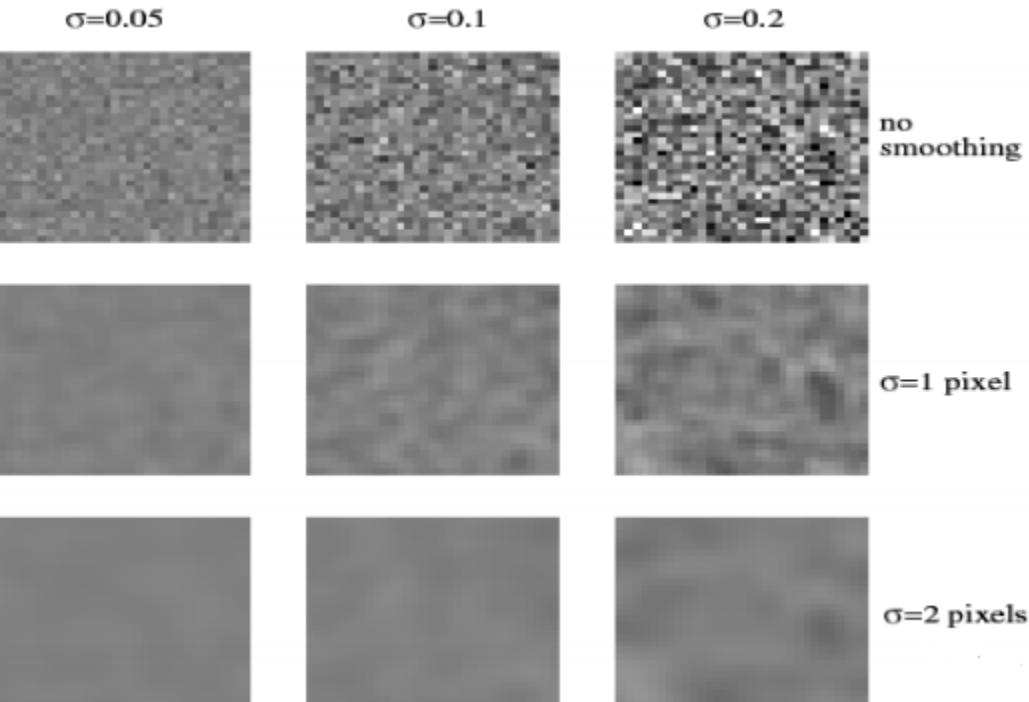


Source: Alper Yilmaz, Mubarak Shah

Gaussian Smoothing...



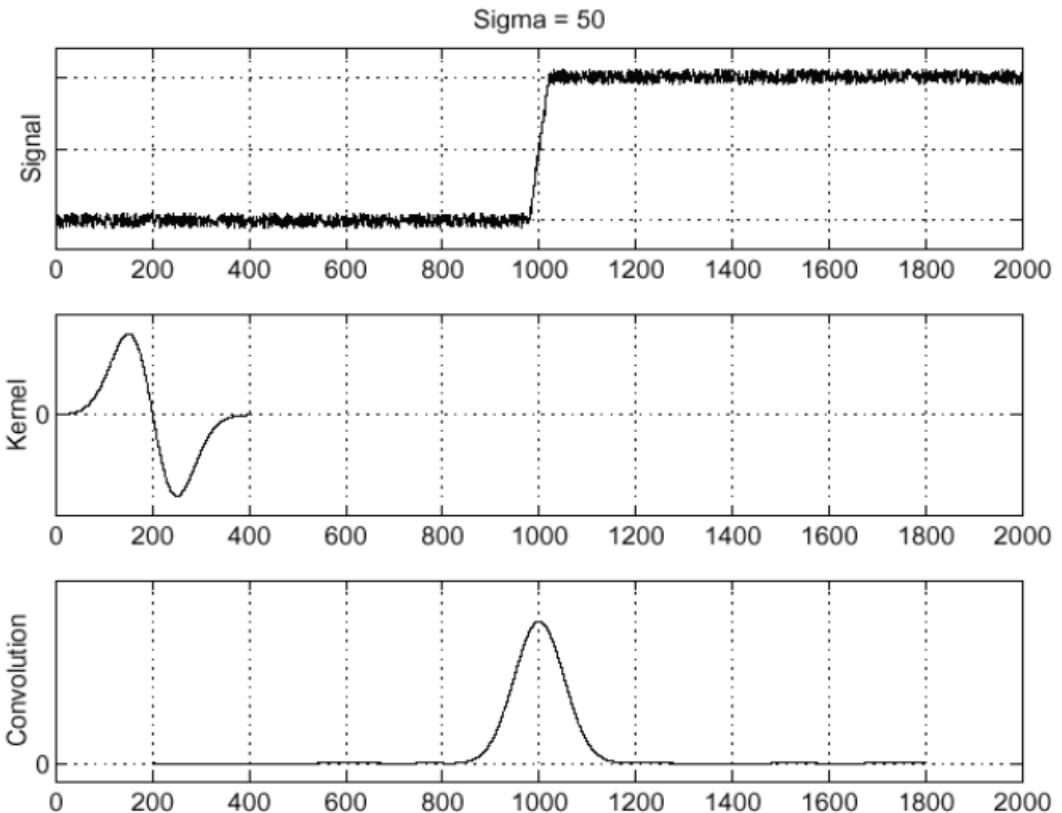
- Scale of Gaussian σ
 - As σ increases, more pixels are involved in average
 - As σ increases, image is more blurred
 - As σ increases, noise is more effectively suppressed



Gaussian Smoothing...

- Differentiate the filter first.
 - Because convolution is associative

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



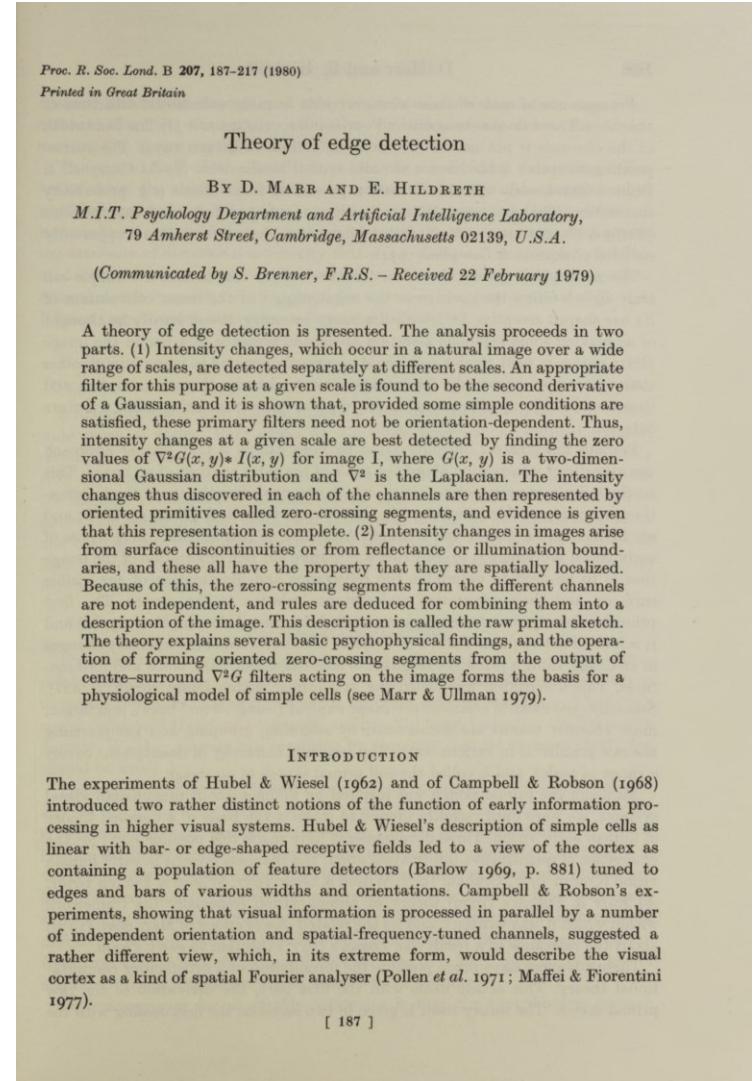
Source: Alper Yilmaz, Mubarak Shah

Outline

- Introduction
- Gradient Operators
- Effect of Noise
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Marr Hildreth Edge Detector

- D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207 (1167), pp.187-217, 1980.



Marr Hildreth Edge Detector

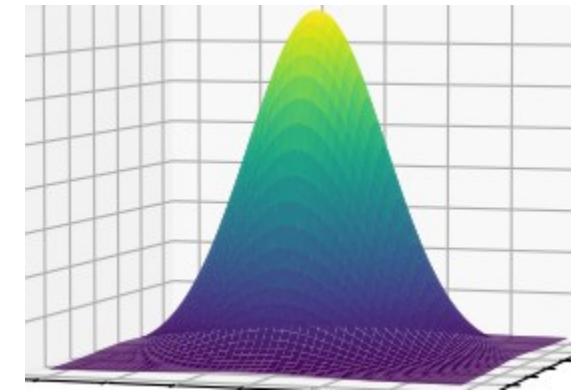
□ Algorithm

1. Smooth image by Gaussian filter
 2. Apply Laplacian to smoothed image
3. Find zero crossings
- Scan along each row, record an edge point at the location of zero-crossing.
 - Repeat above step along each column
- Combine in a single step
Laplacian of Gaussian (LoG)
- 

Marr Hildreth Edge Detector: LoG

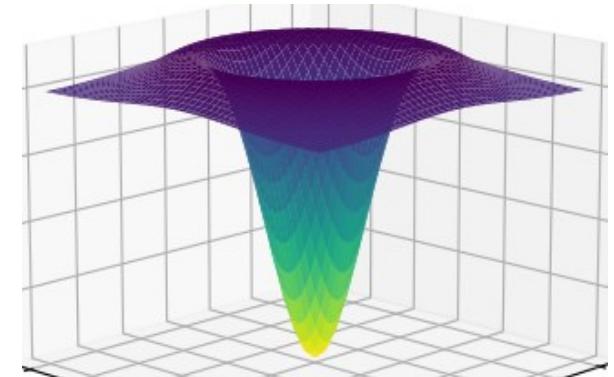
□ Gaussian

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$



□ Laplacian of Gaussian (LoG)

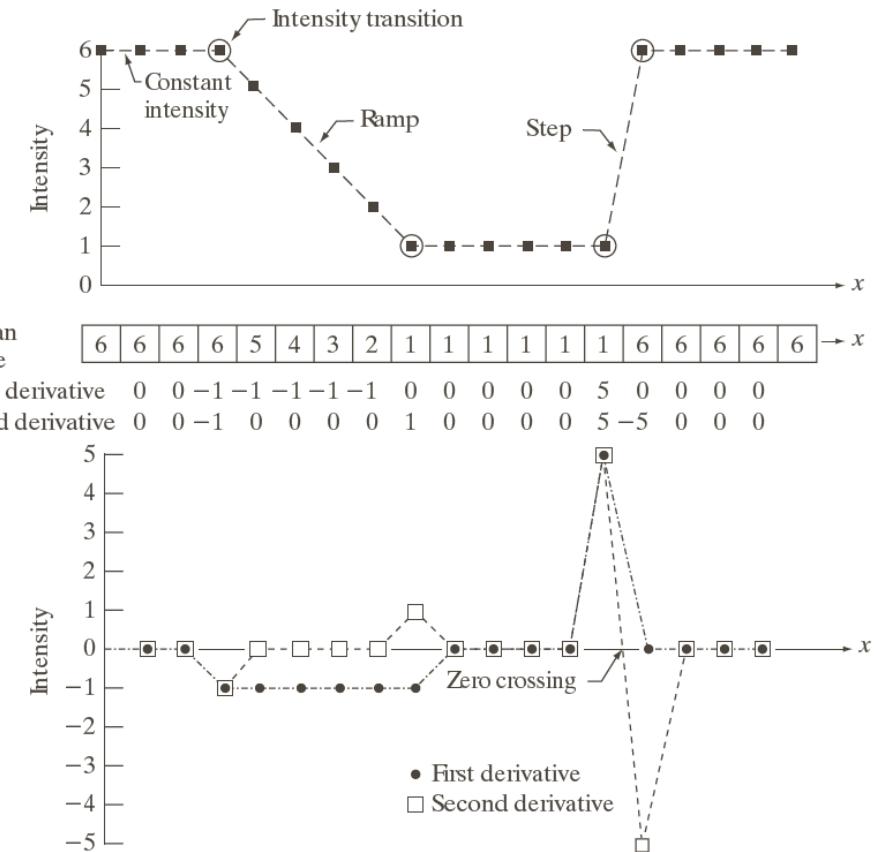
$$\begin{aligned}\nabla^2 G(x, y) &= \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \\ &= \frac{\partial}{\partial x} \left(\frac{-x}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right) + \frac{\partial}{\partial y} \left(\frac{-y}{\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \right) \\ &= \left(\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}} + \left(\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}\end{aligned}$$



$$\nabla^2 G(x, y) = \left(\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} \right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$

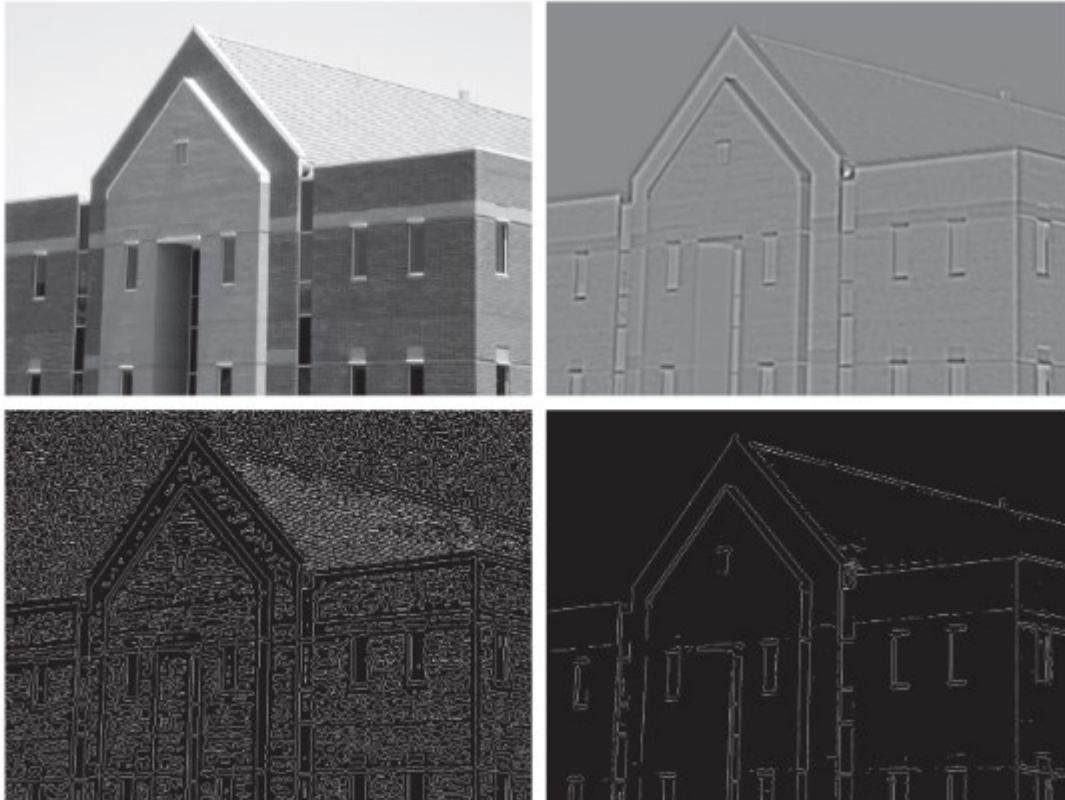
Marr Hildreth Edge Detector: Zero Crossings

- Zero crossings are the points where the Laplacian crosses the zero line (i.e., the sign changes)
- Strength of zero-crossing { $a, -b$ } is $|a+b|$
- To mark an edge
 - Find zero-crossings
 - Compute strength of zero-crossing
 - Apply a threshold to strength



Marr Hildreth Edge Detector...

a
b
c
d



- (a) Image of size 834*1114 pixels, with intensity values scaled to the range [0, 1].
- (b) Result of Steps 1 and 2 of the Marr-Hildreth algorithm using $\sigma = 4$ and $n=25$
- (c) Zero crossings of (b) using a threshold of 0 (note the closed-loop edges).
- (d) Zero crossings found using a threshold equal to 4% of the maximum value of the image in (b). Note the thin edges.

Outline

- Introduction
- Gradient Operators
- Effect of Noise
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

Canny Edge Detector

IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. PAMI-8, NO. 6, NOVEMBER 1986

679

- J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679--714, 1986.

A Computational Approach to Edge Detection

JOHN CANNY, MEMBER, IEEE

Abstract—This paper describes a computational approach to edge detection. The success of the approach depends on the definition of a comprehensive set of goals for the computation of edge points. These goals must be precise enough to delimit the desired behavior of the detector while making minimal assumptions about the form of the solution. We define detection and localization criteria for a class of edges, and present mathematical forms for these criteria as functionals on the operator impulse response. A third criterion is then added to ensure that the detector has only one response to a single edge. We use the criteria in numerical optimization to derive detectors for several common image features, including step edges. On specializing the analysis to step edges, we find that there is a natural uncertainty principle between detection and localization performance, which are the two main goals. With this principle we derive a single operator shape which is optimal at any scale. The optimal detector has a simple approximate implementation in which edges are marked at maximum gradient magnitude of a Gaussian-smoothed image. We extend this simple detector using operators of several widths to cope with different signal-to-noise ratios in the image. We present a general method, called feature synthesis, for the fine-to-coarse integration of information from operators at different scales. Finally, we show that step edge detector performance improves considerably as the operator point spread function is extended along the edge. This detection scheme uses several elongated operators at each point, and the directional operator outputs are integrated with the gradient maximum detector.

Index Terms—Edge detection, feature extraction, image processing, machine vision, multiscale image analysis.

I. INTRODUCTION

EDGE detectors of some kind, particularly step edge detectors, have been an essential part of many computer vision systems. The edge detection process serves to simplify the analysis of images by drastically reducing the amount of data to be processed, while at the same time preserving useful structural information about object boundaries. There is certainly a great deal of diversity in the applications of edge detection, but it is felt that many applications share a common set of requirements. These requirements yield an abstract edge detection problem, the solution of which can be applied in any of the original problem domains.

We should mention some specific applications here. The Binford-Horn line finder [14] used the output of an edge

Manuscript received December 10, 1984; revised November 27, 1985. Recommended for acceptance by S. L. Tanimoto. This work was supported in part by the System Development Foundation, in part by the Office of Naval Research under Contract N00014-81-K-0494, and in part by the Advanced Research Projects Agency under Office of Naval Research Contracts N00014-80-C-0505 and N00014-82-K-0334.

The author is with the Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.
IEEE Log Number 8610412.

detector as input to a program which could isolate simple geometric solids. More recently the model-based vision system ACRONYM [3] used an edge detector as the front end to a sophisticated recognition program. Shape from motion [29], [13] can be used to infer the structure of three-dimensional objects from the motion of edge contours or edge points in the image plane. Several modern theories of stereopsis assume that images are preprocessed by an edge detector before matching is done [19], [20]. Beattie [1] describes an edge-based labeling scheme for low-level image understanding. Finally, some novel methods have been suggested for the extraction of three-dimensional information from image contours, namely shape from contour [27] and shape from texture [31].

In all of these examples there are common criteria relevant to edge detector performance. The first and most obvious is low error rate. It is important that edges that occur in the image should not be missed and that there be no spurious responses. In all the above cases, system performance will be hampered by edge detector errors. The second criterion is that the edge points be well localized. That is, the distance between the points marked by the detector and the "center" of the true edge should be minimized. This is particularly true of stereo and shape from motion, where small disparities are measured between left and right images or between images produced at slightly different times.

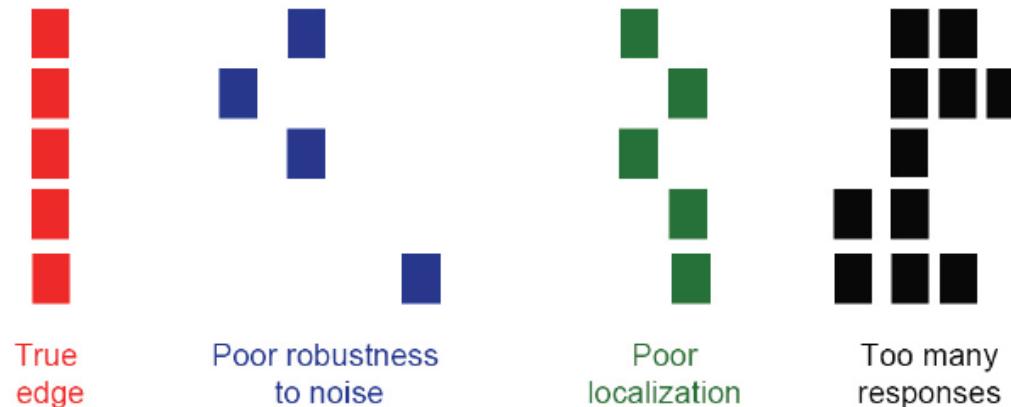
In this paper we will develop a mathematical form for these two criteria which can be used to design detectors for arbitrary edges. We will also discover that the first two criteria are not "tight" enough, and that it is necessary to add a third criterion to circumvent the possibility of multiple responses to a single edge. Using numerical optimization, we derive optimal operators for ridge and roof edges. We will then specialize the criteria for step edges and give a parametric closed form for the solution. In the process we will discover that there is an uncertainty principle relating detection and localization of noisy step edges, and that there is a direct tradeoff between the two. One consequence of this relationship is that there is a single unique "shape" of impulse response for an optimal step edge detector, and that the tradeoff between detection and localization can be varied by changing the spatial width of the detector. Several examples of the detector performance on real images will be given.

II. ONE-DIMENSIONAL FORMULATION

To facilitate the analysis we first consider one-dimensional edge profiles. That is, we will assume that two-

Designing an edge detector

- Three objectives
 - Good Detection
 - There should be a low probability of both false negatives and false positives.
 - Good Localization
 - The detected edge points should be close to the true edge.
 - Single Response
 - Each image edge should generate only a single output edge.



Canny Edge Detector...

- The most widely used edge detector in image processing and computer vision
- Idea:
 - Step-edges are corrupted by additive Gaussian noise.
 - The first derivative of the Gaussian closely approximates the operator that optimizes the product of signal to noise ratio and localization.

Canny Edge Detector...

□ Algorithm

1. Smoothing
2. Differentiation
3. Non-Maximum Suppression
4. Hysteresis Thresholding



Combine in a single step:
Derivative of Gaussian (DoG)

Canny Edge Detector: DoG

1. Convolve the image with a Gaussian mask

$$S = G * I$$

2. Differentiation: compute the gradient

$$S = \nabla(G * I)$$

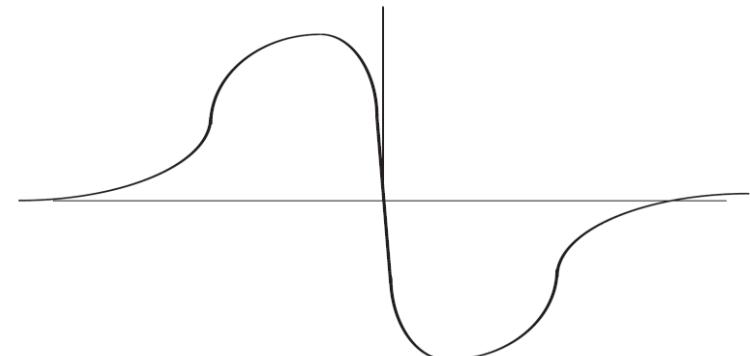
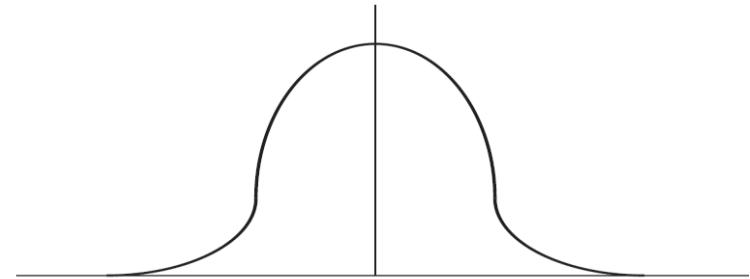
$$\nabla(G * I) = (\nabla G) * I$$

- Magnitude or Edge strength

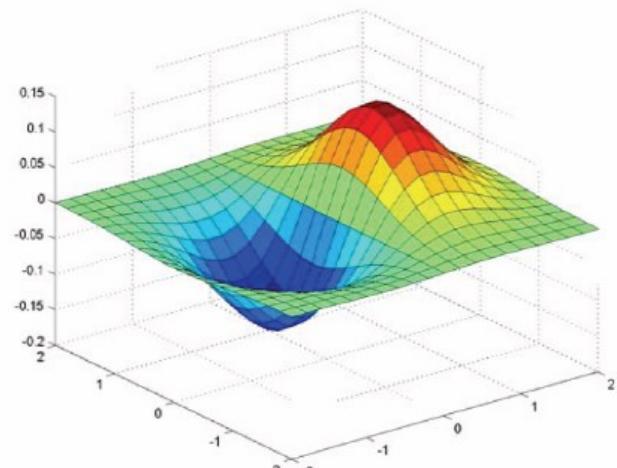
$$e_s = \sqrt{S_x^2(i, j) + S_y^2(i, j)}$$

- Direction of edge e_θ

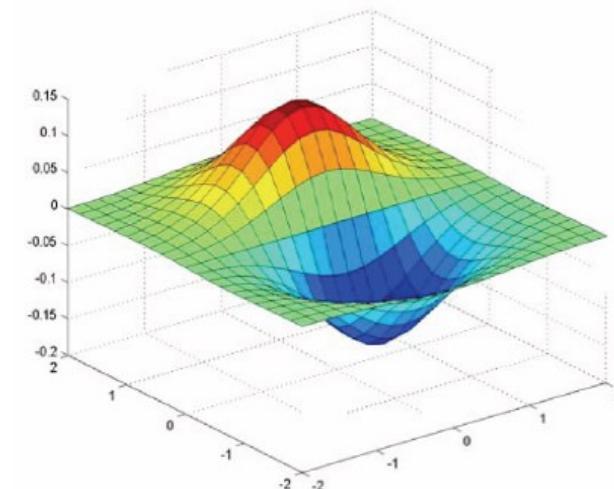
$$e_\theta = \text{atan} \left(\frac{S_y}{S_x} \right)$$



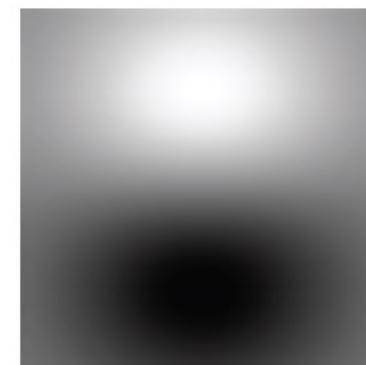
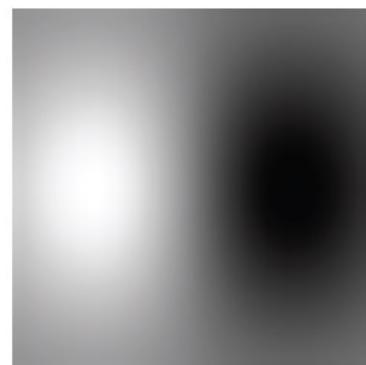
Derivative of Gaussian: DoG...



x-direction



y-direction



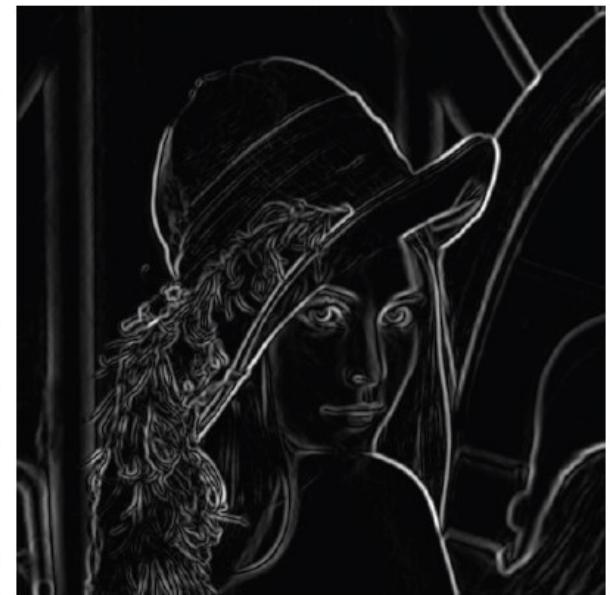
DoG Magnitude



X-Derivative of Gaussian



Y-Derivative of Gaussian

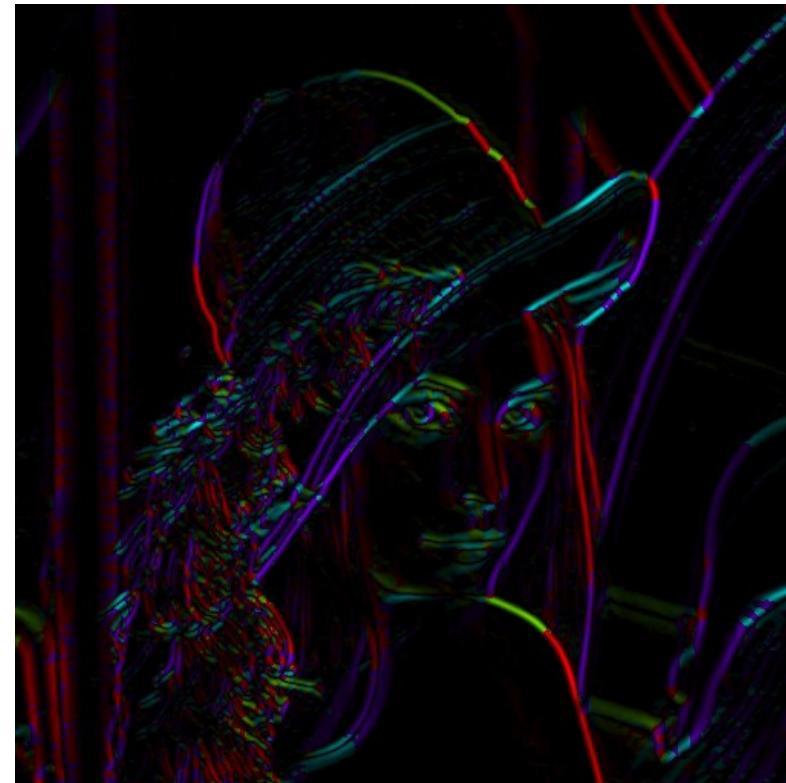


Gradient Magnitude

DoG Orientation

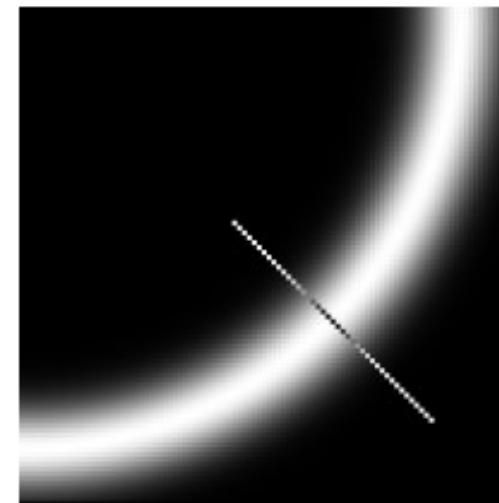
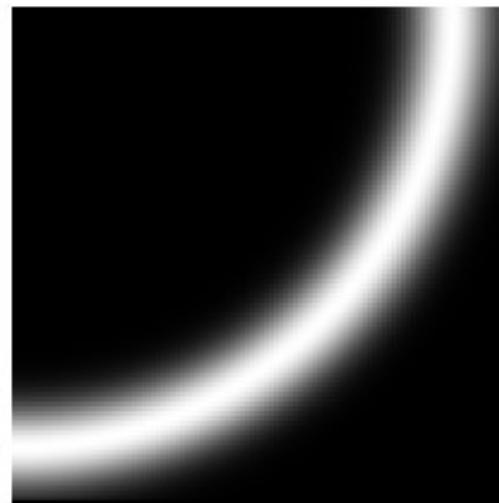
Direction of edge e_θ

$$e_\theta = \text{atan} \left(\frac{S_y}{S_x} \right)$$



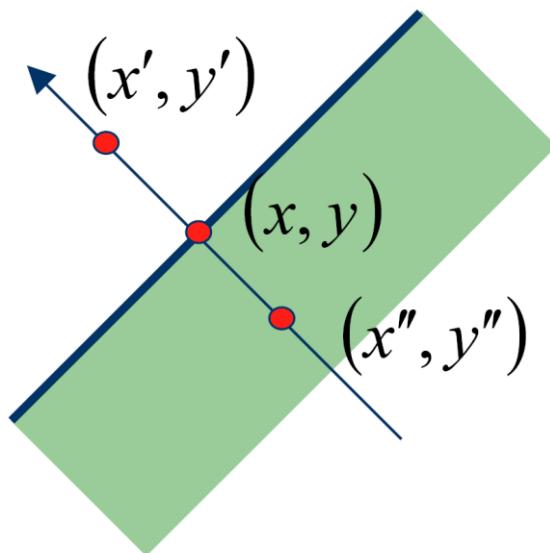
Canny Edge Detector: Non-Maximum Suppression

- Mark points along the edge where the magnitude is biggest.
- We can do this by looking for a maximum along a slice normal to the edge.



Canny Edge Detector: Non-Maximum Suppression...

- Suppress the pixels in $|\Delta S|$ which are not local maximum



$$M(x, y) = \begin{cases} |\nabla S|(x, y) & \text{if } |\Delta S|(x, y) > |\Delta S|(x', y') \\ & \& |\Delta S|(x, y) > |\Delta S|(x'', y'') \\ 0 & \text{otherwise} \end{cases}$$

- x' and x'' are the neighbors of x along normal direction to an edge

Canny Edge Detector: Non-Maximum Suppression...

Quantization of Normal Directions

$$\tan \theta = \frac{S_y}{S_x}$$

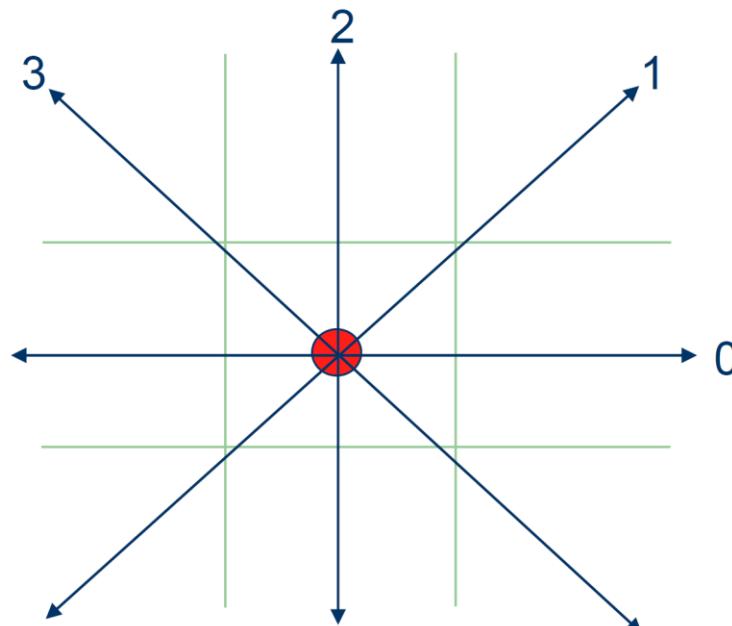
Quantizations:

0: $-0.4142 < \tan \theta \leq 0.4142$

1: $0.4142 < \tan \theta < 2.4142$

2: $|\tan \theta| \geq 2.4142$

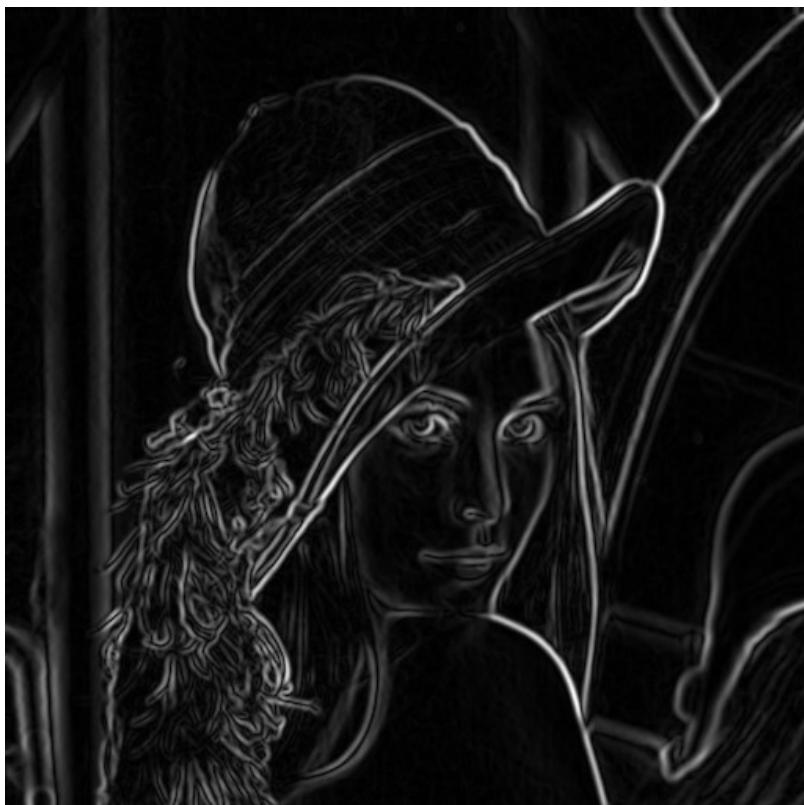
3: $-2.4142 < \tan \theta \leq -0.4142$



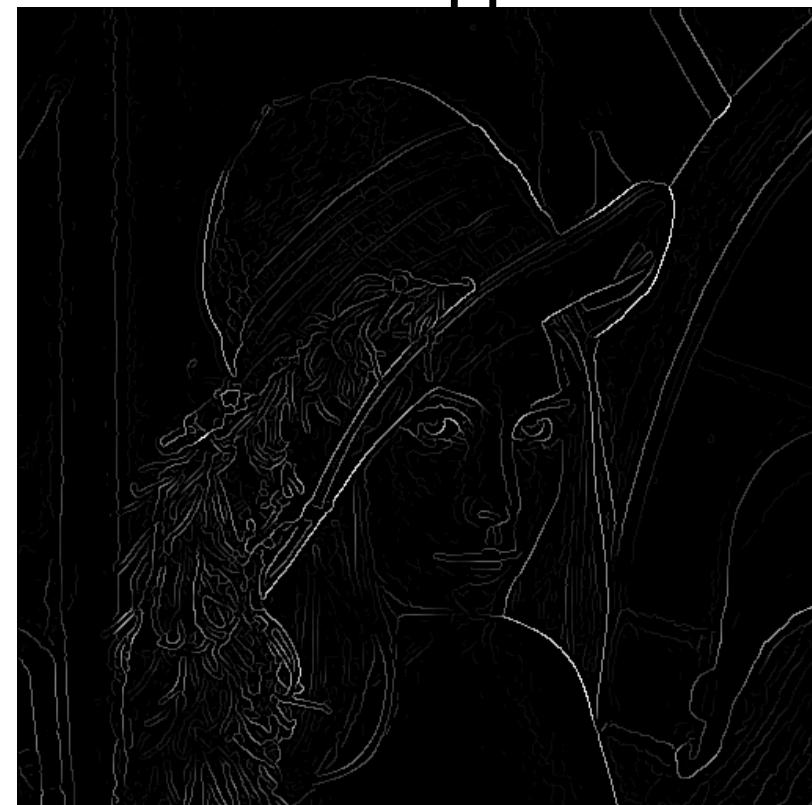
Source: Alper Yilmaz, Mubarak Shah

Canny Edge Detector: Non-Maximum Suppression...

Before
non-max suppression



After
non-max suppression

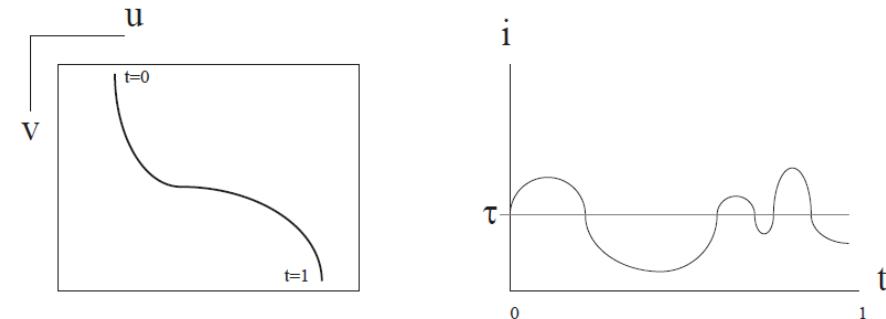


Canny Edge Detector: Hysteresis Thresholding

- Thresholding
 - Set all pixels to zero whose values are less than some threshold τ

What are the disadvantages ?

- Difficult to set the value of τ .
- Streaking can occur with an edge oscillates above and below τ .



Canny Edge Detector: Hysteresis Thresholding...

- Use two thresholds
- If the gradient at a pixel is
 - above "High", declare it an „edge pixel“
 - below "Low", declare it a "non-edge-pixel"
 - between "low" and "high"
 - consider its neighbors iteratively then declare it an "edge pixel" if it is connected to an "edge pixel" directly or via pixels between "low" and "high".

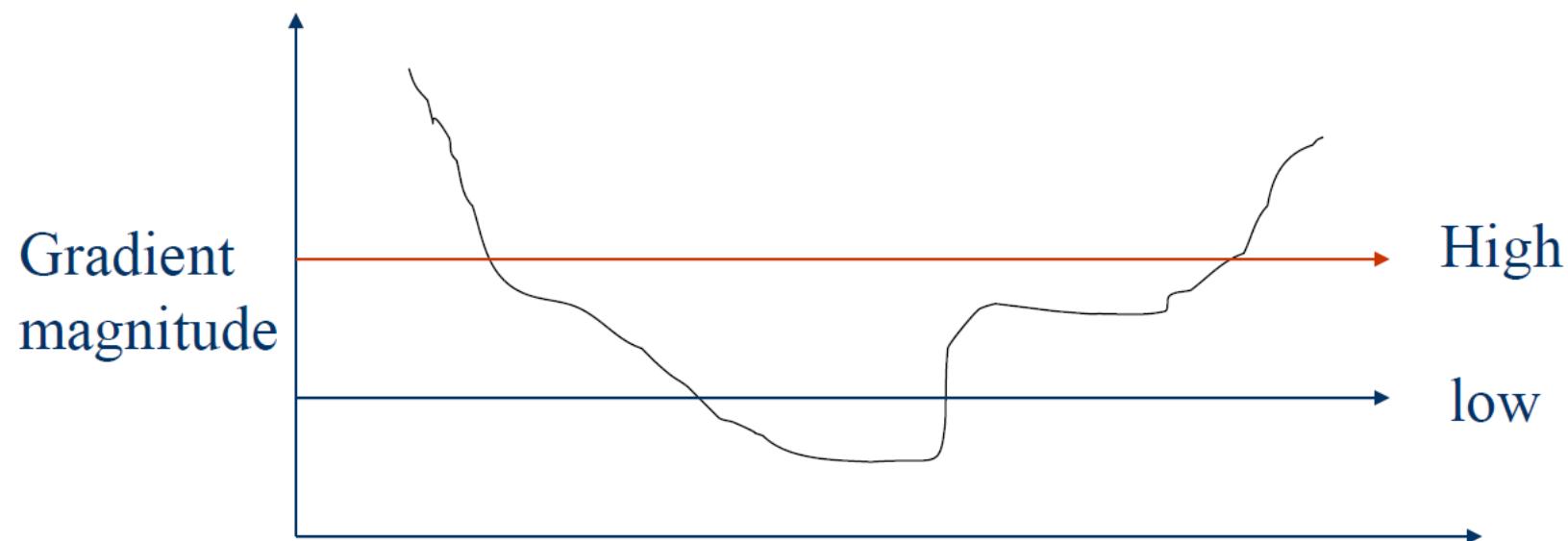
Canny Edge Detector: Hysteresis Thresholding...

■ Algorithm

1. The filtered edge strength image is scanned until a pixel value is found greater than a high threshold value τ_H .
2. This pixel is set to "on" as an edge pixel. The edge is tracked outward from this pixel.
3. All connected pixels with values greater than a low threshold τ_L are also set to "on".
4. Repeat until all pixels are accessed.

□ Ratio of $\frac{\tau_H}{\tau_L} \approx 2$ or 3

Canny Edge Detector: Hysteresis Thresholding...



Canny Edge Detector

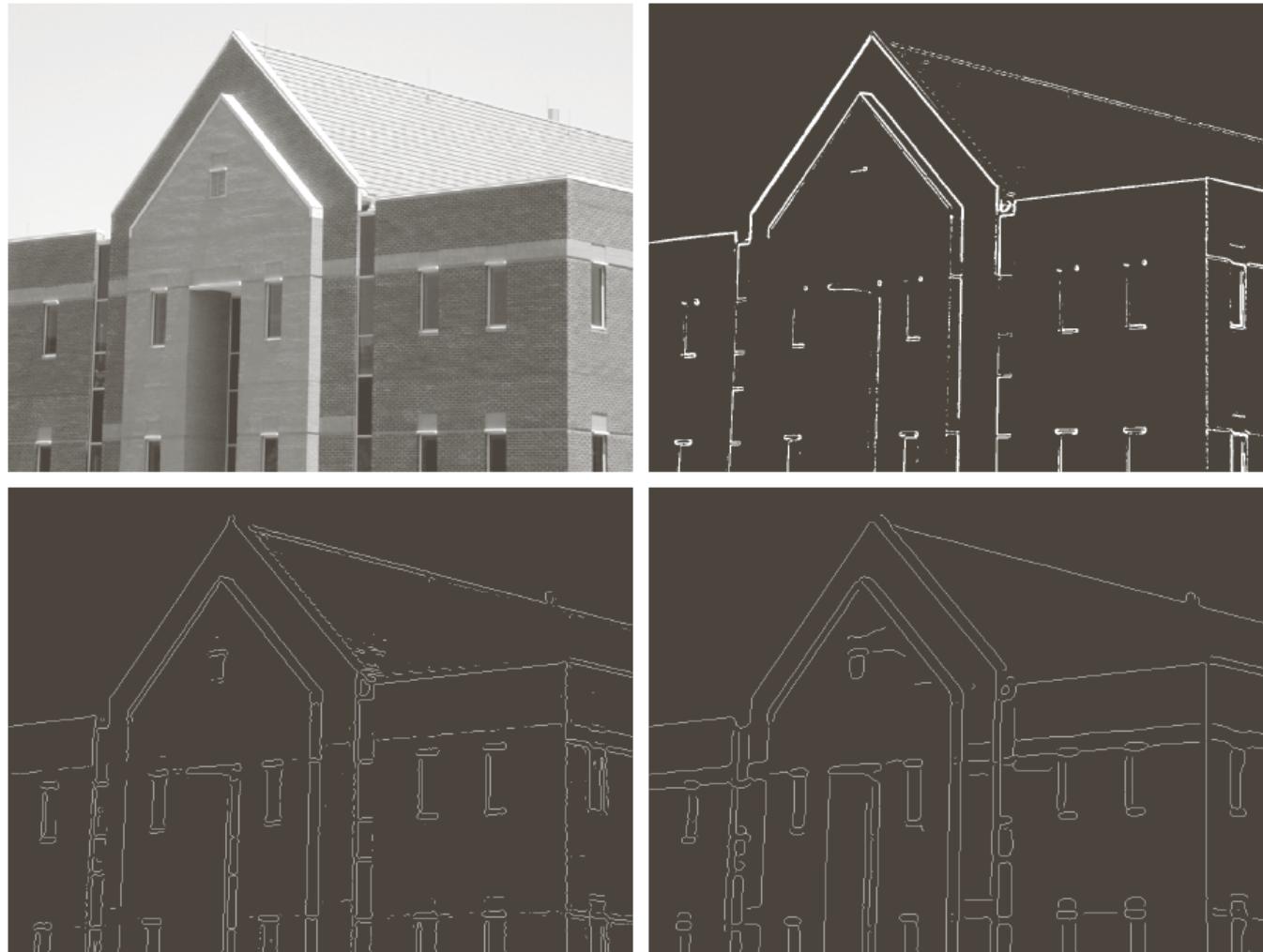
Original



Final Canny Edges



Results Comparison

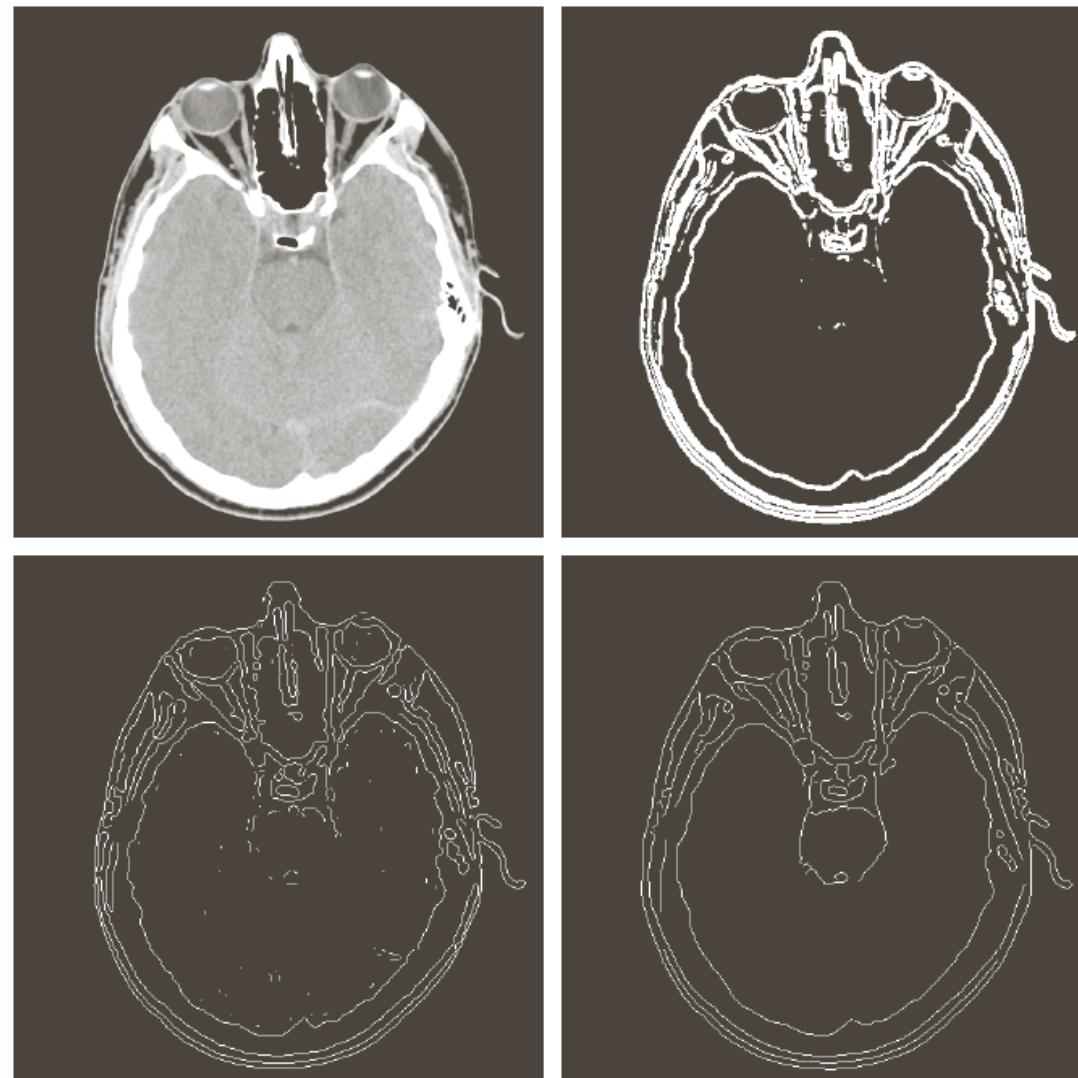


a b
c d

FIGURE 10.25
(a) Original image of size 834×1114 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. Note the significant improvement of the Canny image compared to the other two.

Source: Gonzalez and Woods

Results Comparison...



a b
c d

FIGURE 10.26

(a) Original head CT image of size 512×512 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.)

Source: Gonzalez and Woods

Results Comparison...

- In general: Canny gives the best results, followed by Marr-Hildreth, followed by Sobel, followed by Prewitt.
- Canny is a bit slower than others but still has acceptable speed.
- Always use Canny as default.

Summary

- In this lecture, we learned
 - Introduction
 - Gradient Operators
 - Effect of Noise
 - Laplacian of Gaussian (Marr-Hildreth)
 - Gradient of Gaussian (Canny)

Readings and Credits

□ Readings

- J. Canny, A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679--714, 1986.
- D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 207(1167), pp.187-217, 1980.
- Chapter 10, "Digital Image Processing (Fourth Edition)", Rafael C. Gonzalez and Richard E. Woods, Prentice Hall, 2017

□ Slides sources

- The textbook (Gonzalez and Woods)
- Lecture notes by Mubarak Shah, Univ. of Central Florida

Image Processing & Pattern Recognition

Geometric Primitive Extraction

Dr. Zia Ud Din

Outline

- Model fitting methods
 - Introduction
 - RANSAC
 - Hough Transform

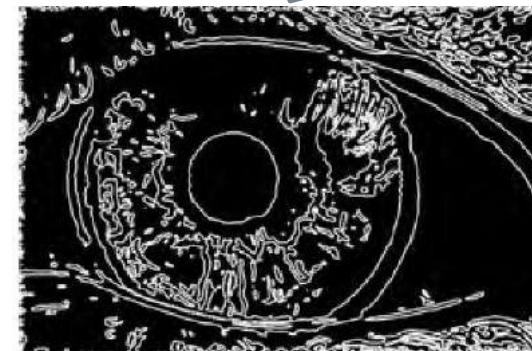
Introduction

- So far
 - Image Enhancement
 - Histogram Processing
 - Image Filters
 - Edges
 - Edge detection
 - Contours extraction
- This lecture
 - Associate more information (properties) to the edges

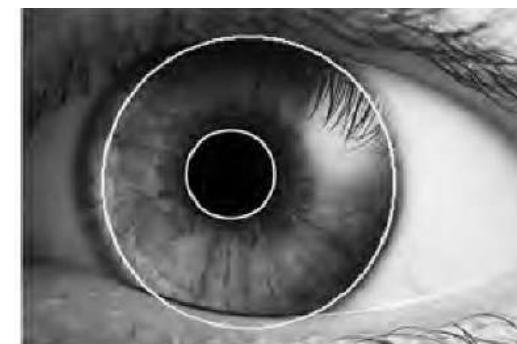
Introduction



filtering, edge
detection, and
thresholding



Circle Detection



Assignment 3

Iris Segmentation

- Find iris and pupil circles from an iris recognition dataset
- Will be uploaded later but start thinking about the problem and potential solutions

Introduction

- Choose a parametric model to represent a set of features
- Membership criterion is not local
 - No prior knowledge whether an edge pixel belongs to a given model
- Three main questions
 - What model represents this set of edge pixels best?
 - How many model instances are there?
 - Which of several model instances gets which edge pixel?

Introduction

- Objective: Aggregate edges (or contour) pixels into higher level geometric primitives, such as:
 - Lines
 - Curves
 - Circles
 - Ellipses
 - Arbitrary shapes
- Two algorithms
 - RANSAC
 - Hough Transform

Line Extraction

- A line is a higher order representative than an edge
 - Implies some interpretation of image content
- Can be used to recover geometry and viewpoint



Line Extraction

□ Challenges

■ Clutter

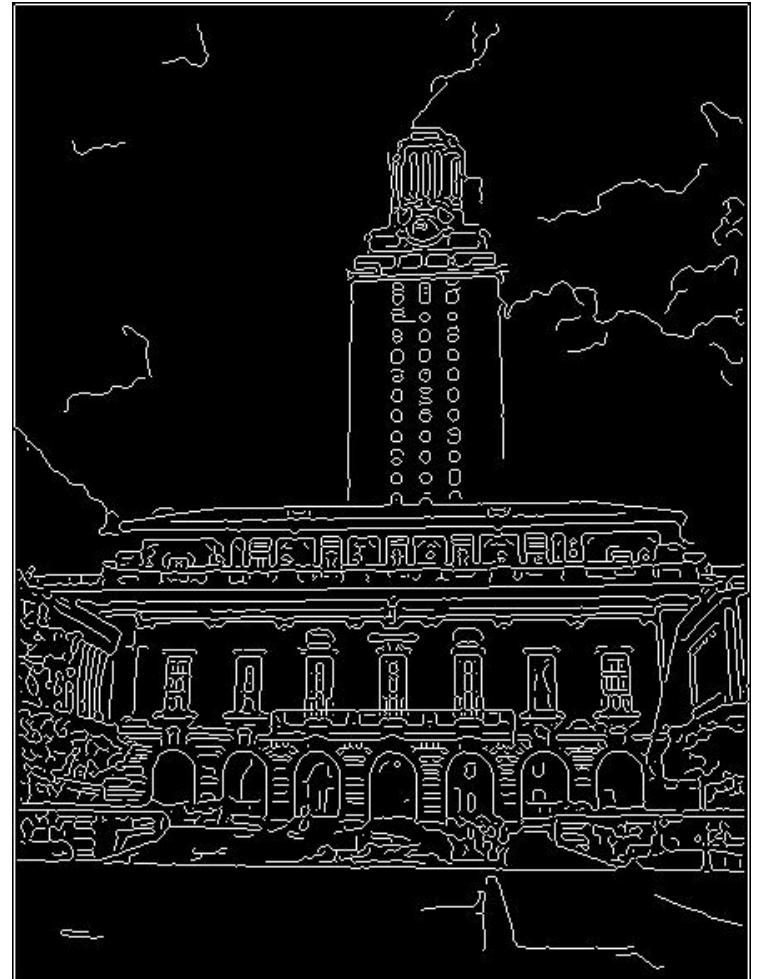
- Extra edge points, thus, multiple models
- Which points go with which line, if any?

■ Missing part of lines

- How to find a line that bridges missing evidence?

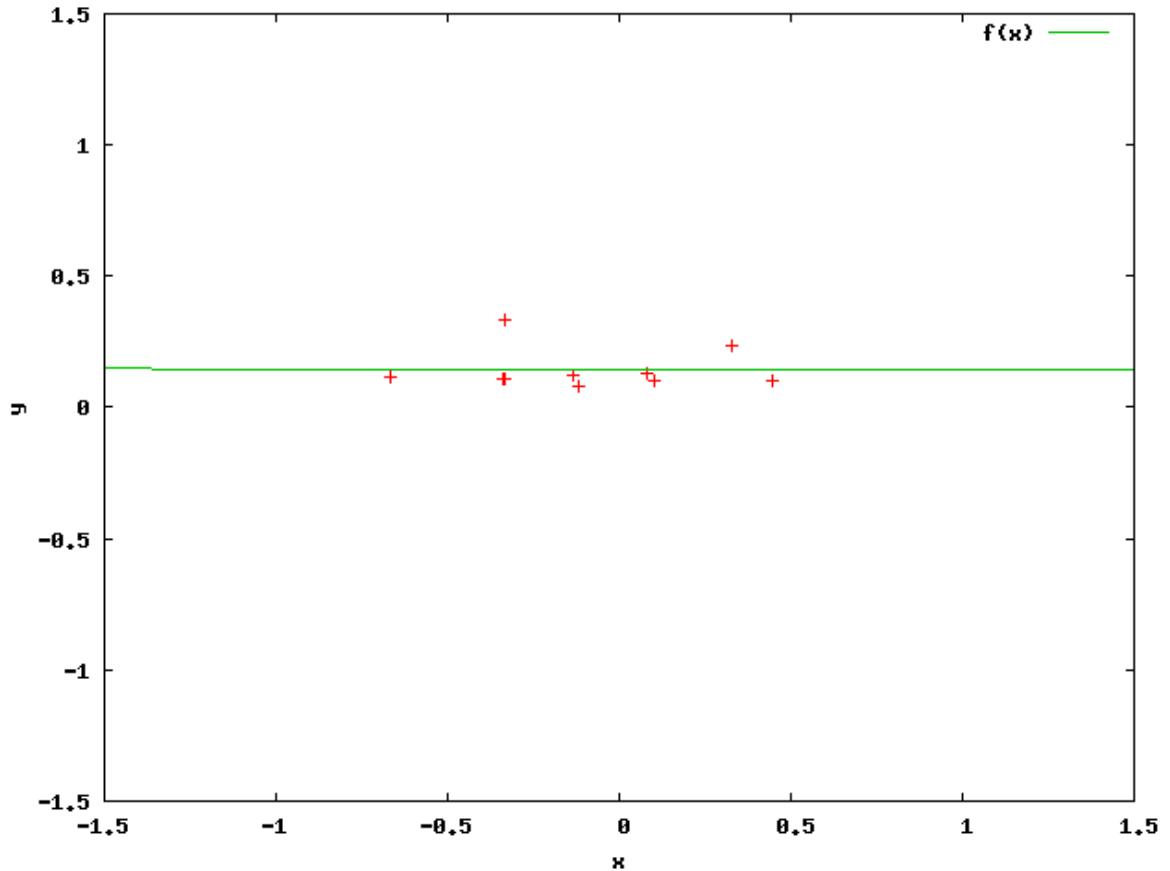
■ Noise in measured edges, orientations

- How to detect true underlying parameters?



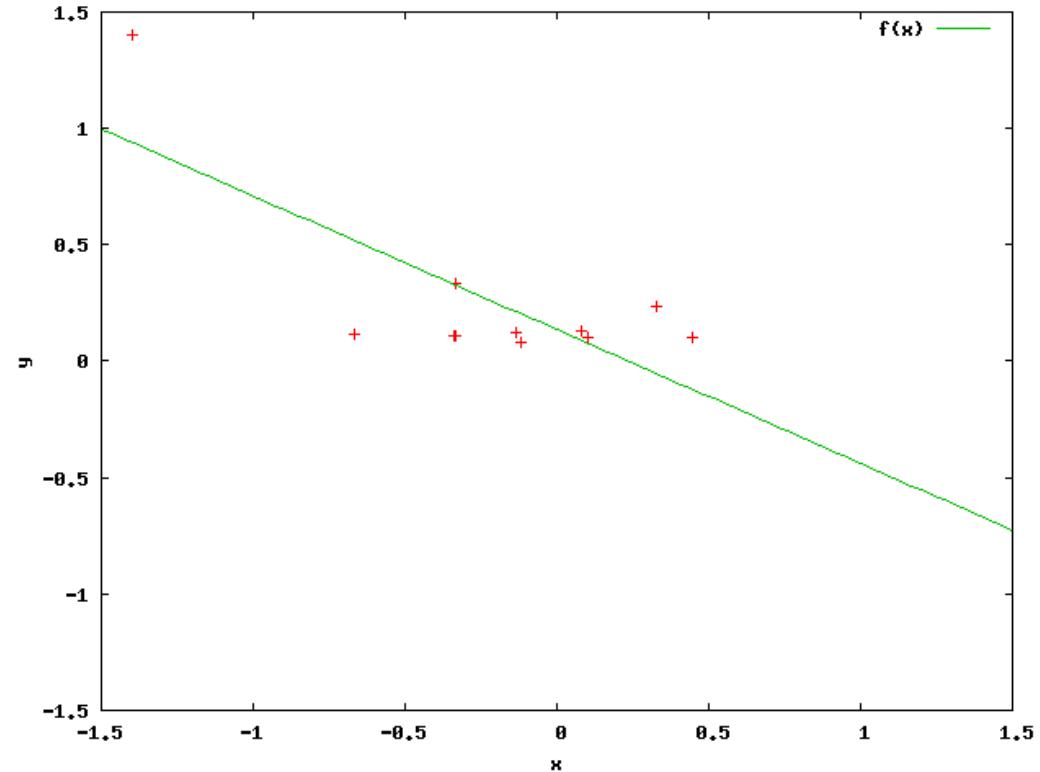
Line Extraction

- Q/ Let's say we have a point set, that was derived from an underlying line, e.g. edge contours
- How can we extract a line from the point set?
- Ans/ Least Mean Squares!



Line Extraction

- Now let's say there is one more point (top left) - this is due to noise, for example
- The impact is to completely bias the result of the line fitting routine
 - the new point is an "outlier"
 - doesn't fit the model
- LMS is called "brittle", as the input does not degrade gracefully in the face of outliers
- Want a method that is "robust" to some degree of outliers



Line Extraction

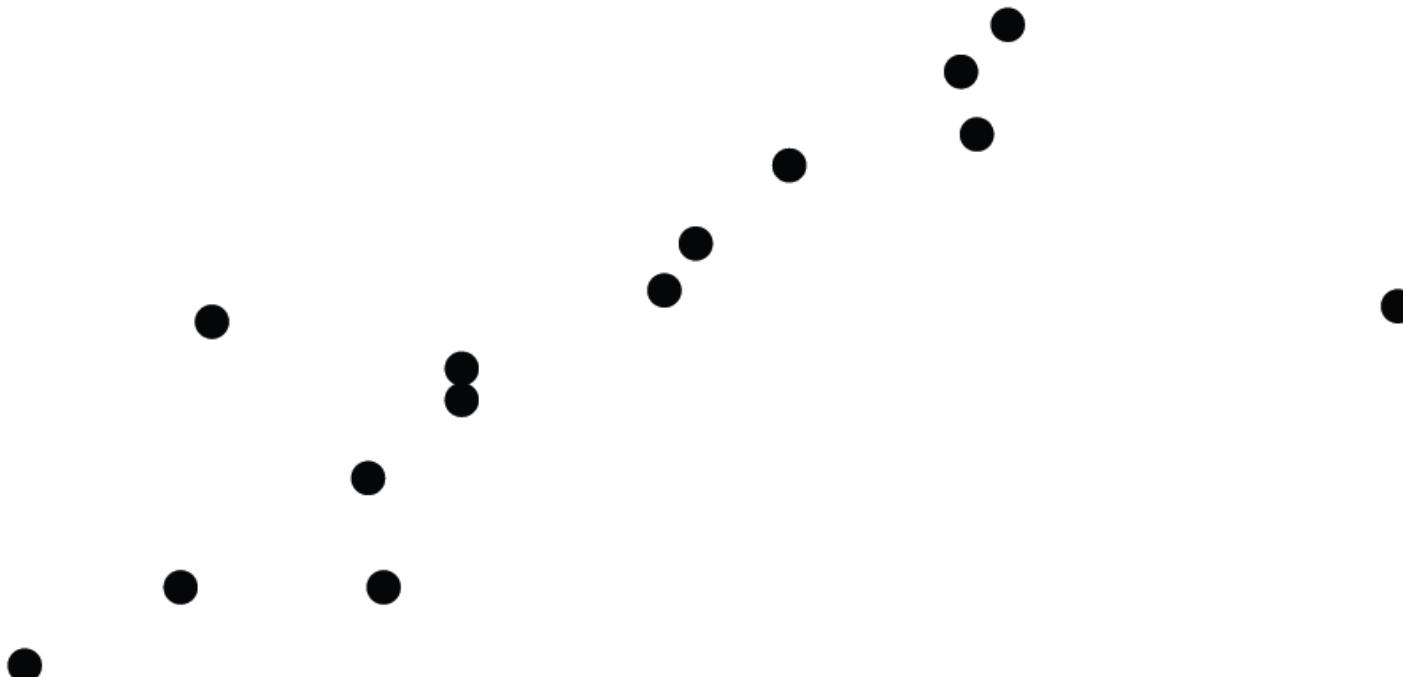
- **RANSAC**
 - **RANDom SAmple Consensus**
 - Generate-and-Test method
 - Based on Robust Statistics
 - Simple, and surprisingly flexible for many model fittings
 - **Idea**
 - Discard **outliers**
 - i.e. those points that do not nicely fit the model
 - Only use **inliers**
 - i.e. those points that do nicely fit the model
 - If an outlier is used to hypothesize a model, it will not be supported (voted) by the rest of the features

RANSAC

1. Randomly select a seed set of points
 - Only select minimum number of seeds that are required to hypothesize a line
2. Compute the transformation from seed set
3. Find inliers to this transformation
 - Loop through all other points, and check their distance to the hypothesized line
4. Repeat steps 1 to 3 and keep the transformation with largest number of inliers
5. Optional refinement
 - re-compute least-square estimate of final transformation using inliers only

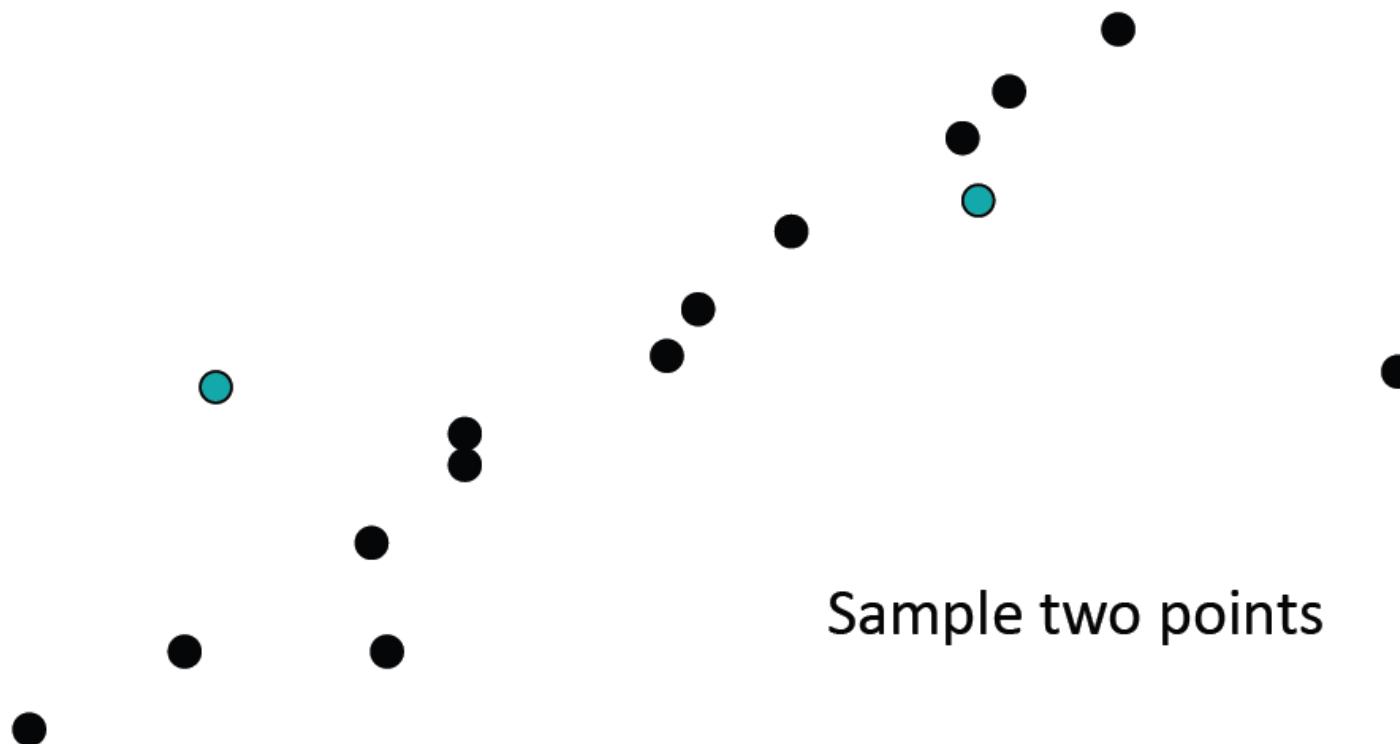
RANSAC for line extraction

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



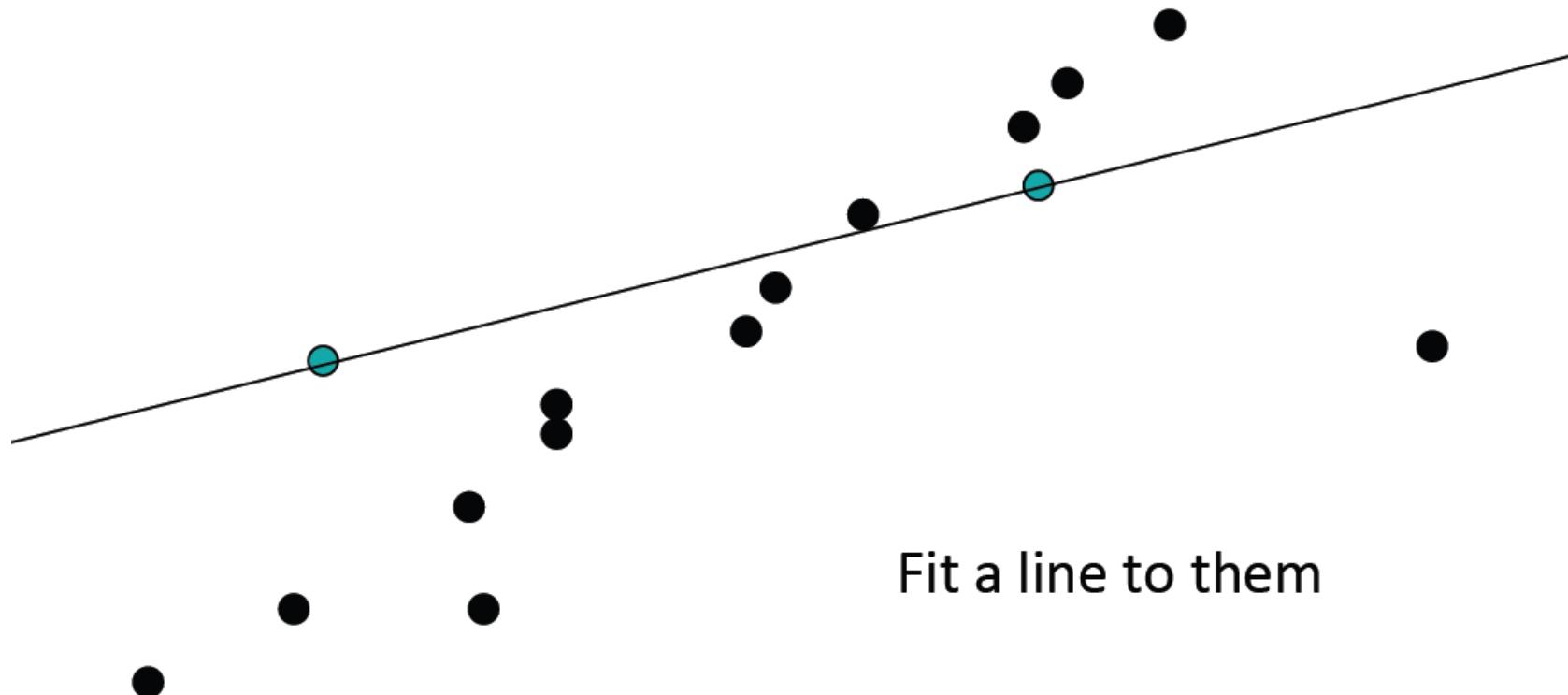
RANSAC for line extraction

- Task: Estimate the best line



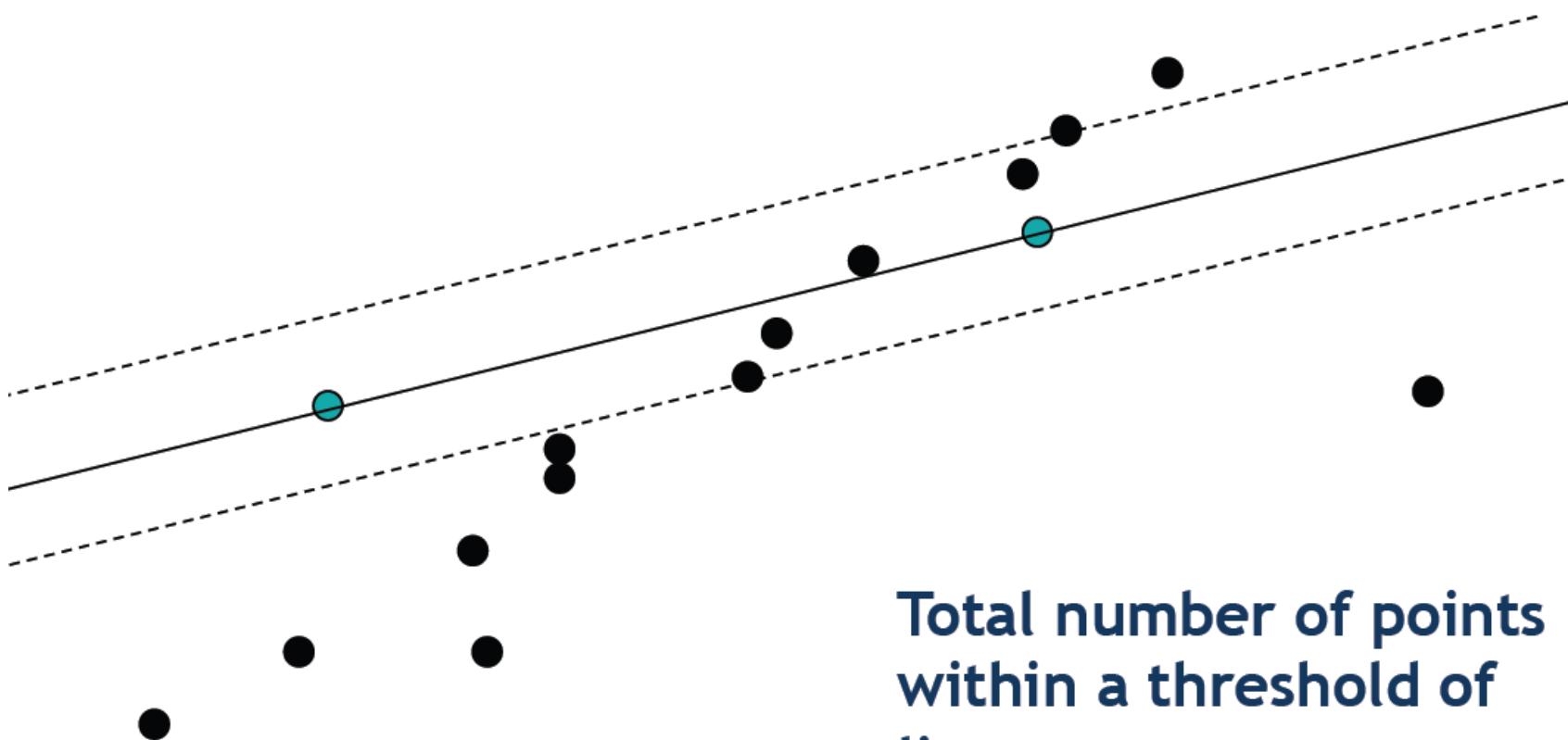
RANSAC for line extraction

- Task: Estimate the best line



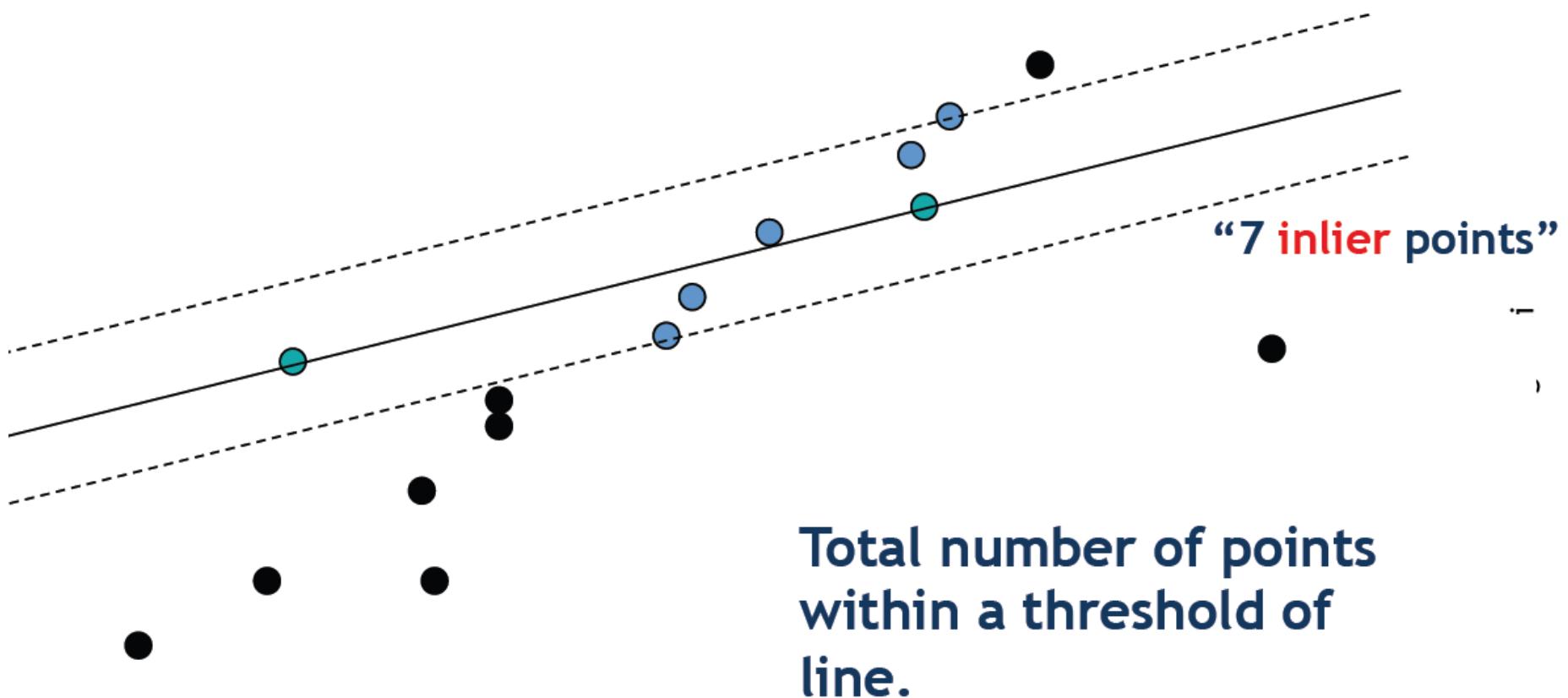
RANSAC for line extraction

- Task: Estimate the best line



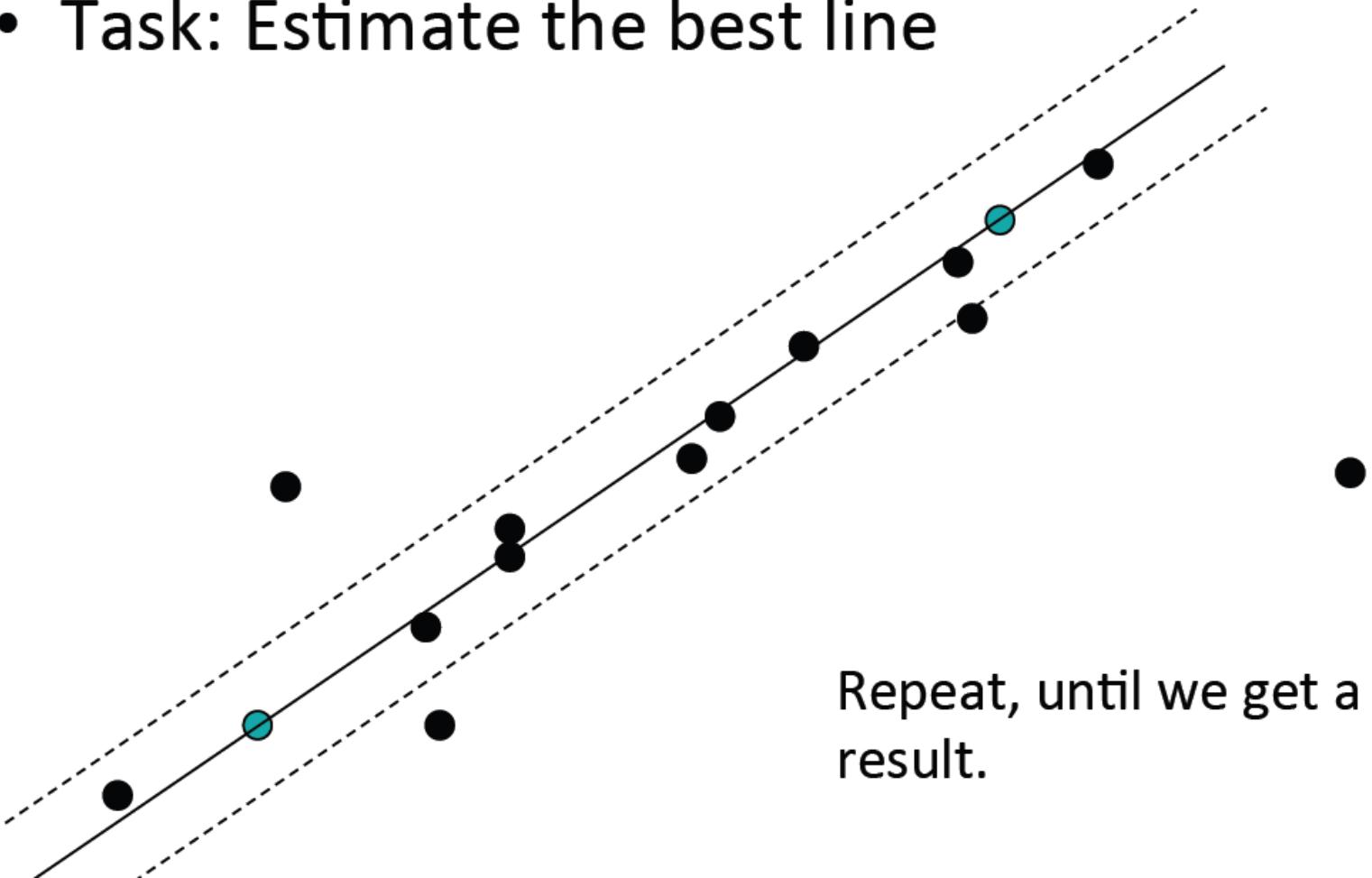
RANSAC for line extraction

- Task: Estimate the best line



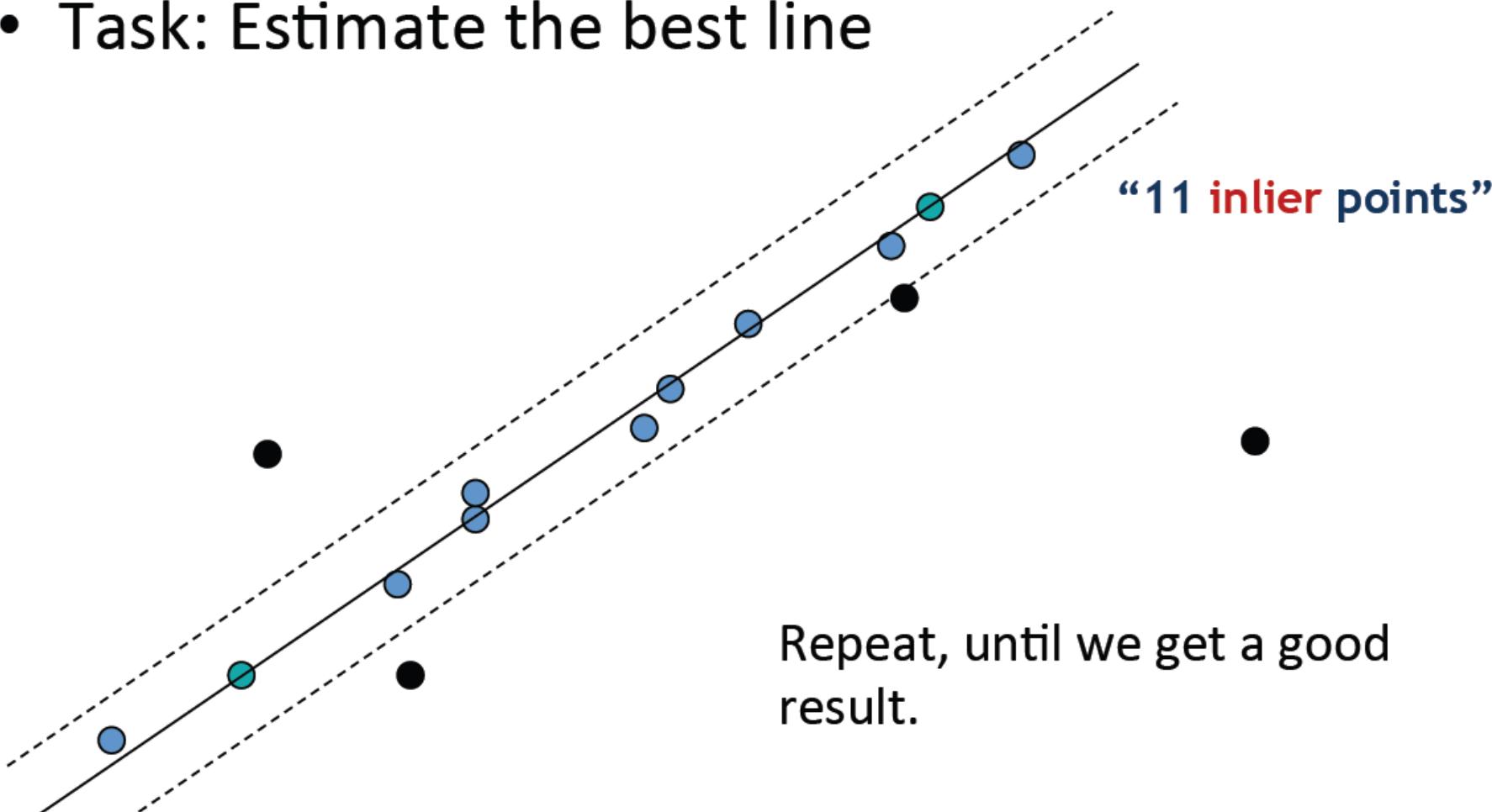
RANSAC for line extraction

- Task: Estimate the best line



RANSAC for line extraction

- Task: Estimate the best line



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

RANSAC

- How many random samples are needed?
 - Let ω is the probability of choosing an inlier each time a single point is selected

$$\omega = \frac{\text{total inliers}}{\text{total points}}$$

- n = points required to define a model
 - for line $n = 2$
- k = number of samples chosen randomly

RANSAC

- Probability that a single sample of n is an inlier:

$$P_i = \omega^n$$

- Probability that a single sample fails:

$$P_f = 1 - \omega^n$$

- Probability that all k samples fail:

$$P_f = (1 - \omega^n)^k$$

- k must be chosen as high enough to keep this below desired failure rate

RANSAC

k	w	Pf	Ps=1-Pf
1	0.75	0.4375	0.5625
2	0.75	0.191406	0.808594
3	0.75	0.08374	0.91626
4	0.75	0.036636	0.963364
5	0.75	0.016028	0.983972
6	0.75	0.007012	0.992988
7	0.75	0.003068	0.996932
8	0.75	0.001342	0.998658
9	0.75	0.000587	0.999413
10	0.75	0.000257	0.999743
100	0.75	1.25E-36	1
1000	0.75	0	1

k	w	Pf	Ps=1-Pf
1	0.5	0.75	0.25
2	0.5	0.5625	0.4375
3	0.5	0.421875	0.578125
4	0.5	0.316406	0.683594
5	0.5	0.237305	0.762695
6	0.5	0.177979	0.822021
7	0.5	0.133484	0.866516
8	0.5	0.100113	0.899887
9	0.5	0.075085	0.924915
10	0.5	0.056314	0.943686
100	0.5	3.21E-13	1
1000	0.5	1.2E-125	1

k	w	Pf	Ps=1-Pf
1	0.1	0.99	0.01
2	0.1	0.9801	0.0199
3	0.1	0.970299	0.029701
4	0.1	0.960596	0.039404
5	0.1	0.95099	0.04901
6	0.1	0.94148	0.05852
7	0.1	0.932065	0.067935
8	0.1	0.922745	0.077255
9	0.1	0.913517	0.086483
10	0.1	0.904382	0.095618
100	0.1	0.366032	0.633968
1000	0.1	4.32E-05	0.999957

RANSAC

□ Pros:

- General method suited for a wide range of model fitting problems
- Easy to implement and easy to calculate its failure rate

□ Cons:

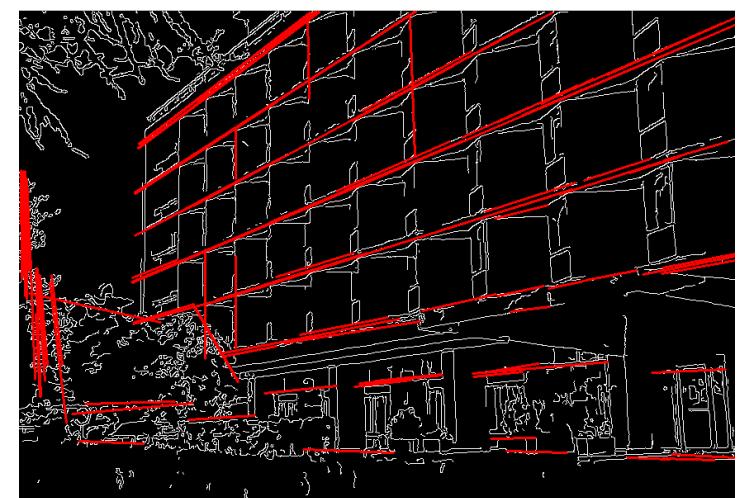
- Only handles a moderate percentage of outliers without cost blowing up
 - especially true for higher values of n
- Many real problems have high rate of outliers
 - Sometimes, a selective choice of random subsets can also help

RANSAC: Computed k ($P_s = 0.99$)

Sample size	Proportion of outliers						
	5%	10%	20%	25%	30%	40%	50%
n							
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

Hough Transform

- There is a very elegant and powerful technique to extract lines (and other primitives) called the *Hough Transform*
- An alternative to RANSAC, that can be more time efficient
 - Although less space efficient
- Very well studied
 - A large collection of techniques



Voting

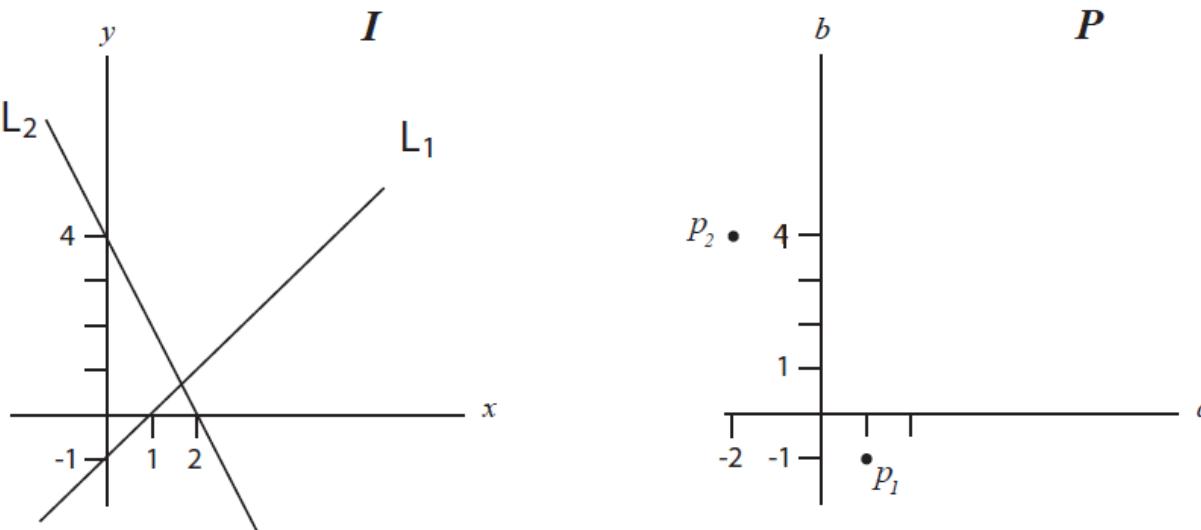
- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- Voting is a general technique where we let the features vote for all models that are compatible with it.
 - Cycle through features, cast votes for model parameters.
 - Look for model parameters that 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.

Hough Transform - Lines

- For an image $I = x \times y$, any line $L_i \in I$ can be described as

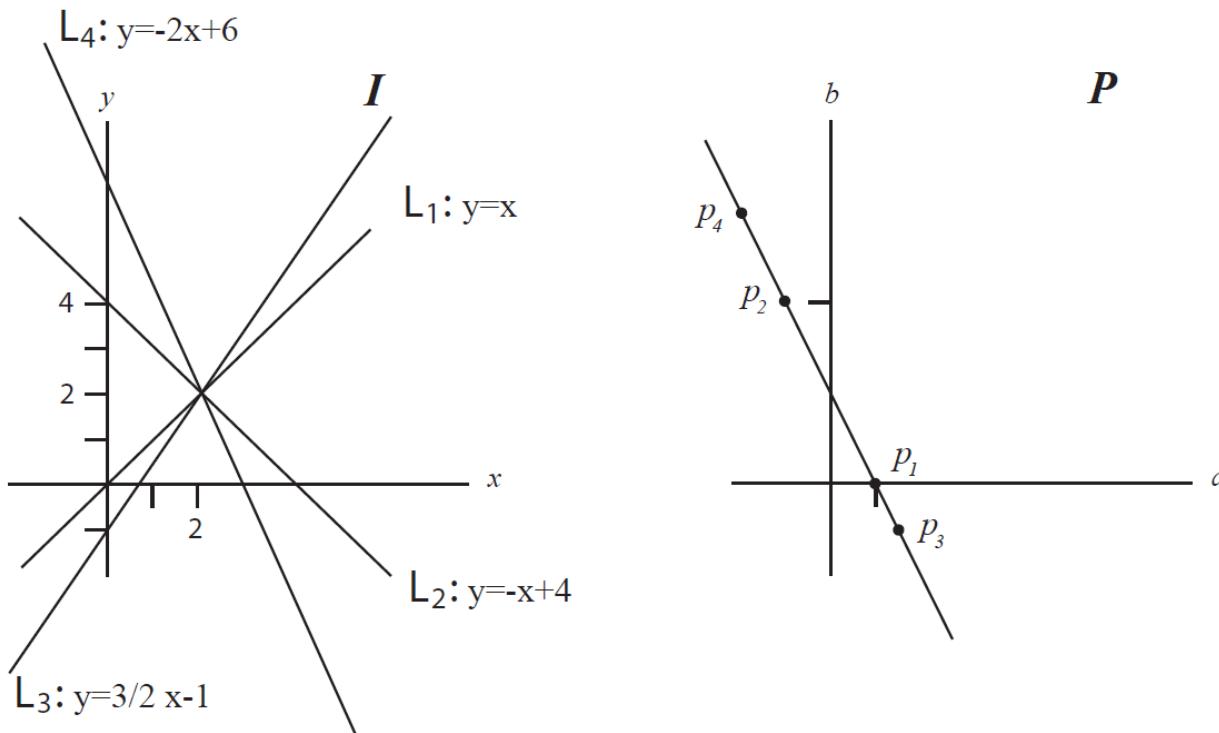
$$y_i = a_i x_i + b_i$$

- A dual parameter space $P = a \times b$ where every point $p_i \in P$ corresponds to some line $L_i \in I$



Hough Transform - Lines

- Consider a point $m_i = (x_i, y_i)$
 - There exist a family of lines that intersect at m_i
 - Each line will map to a point $p_i \in P$

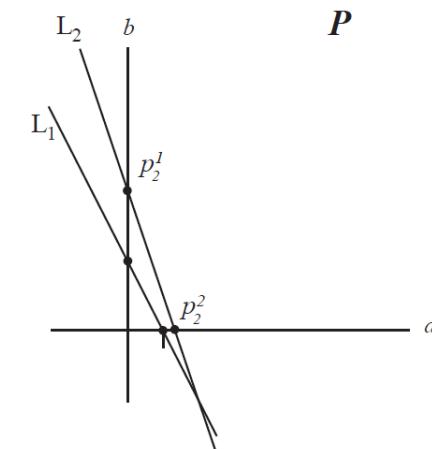
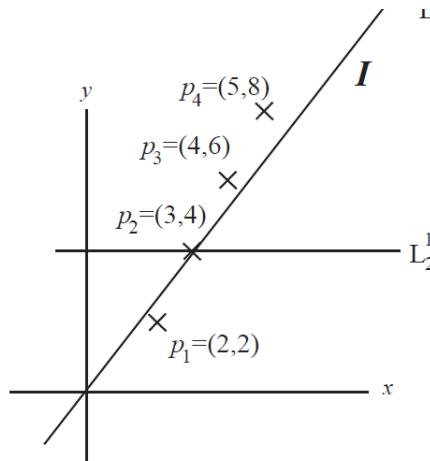
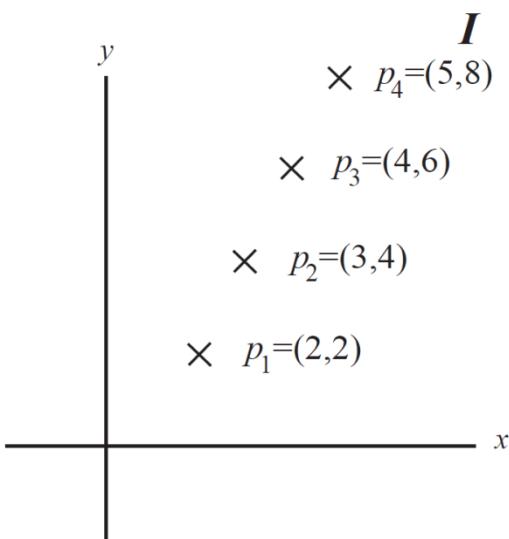


Hough Transform - Lines

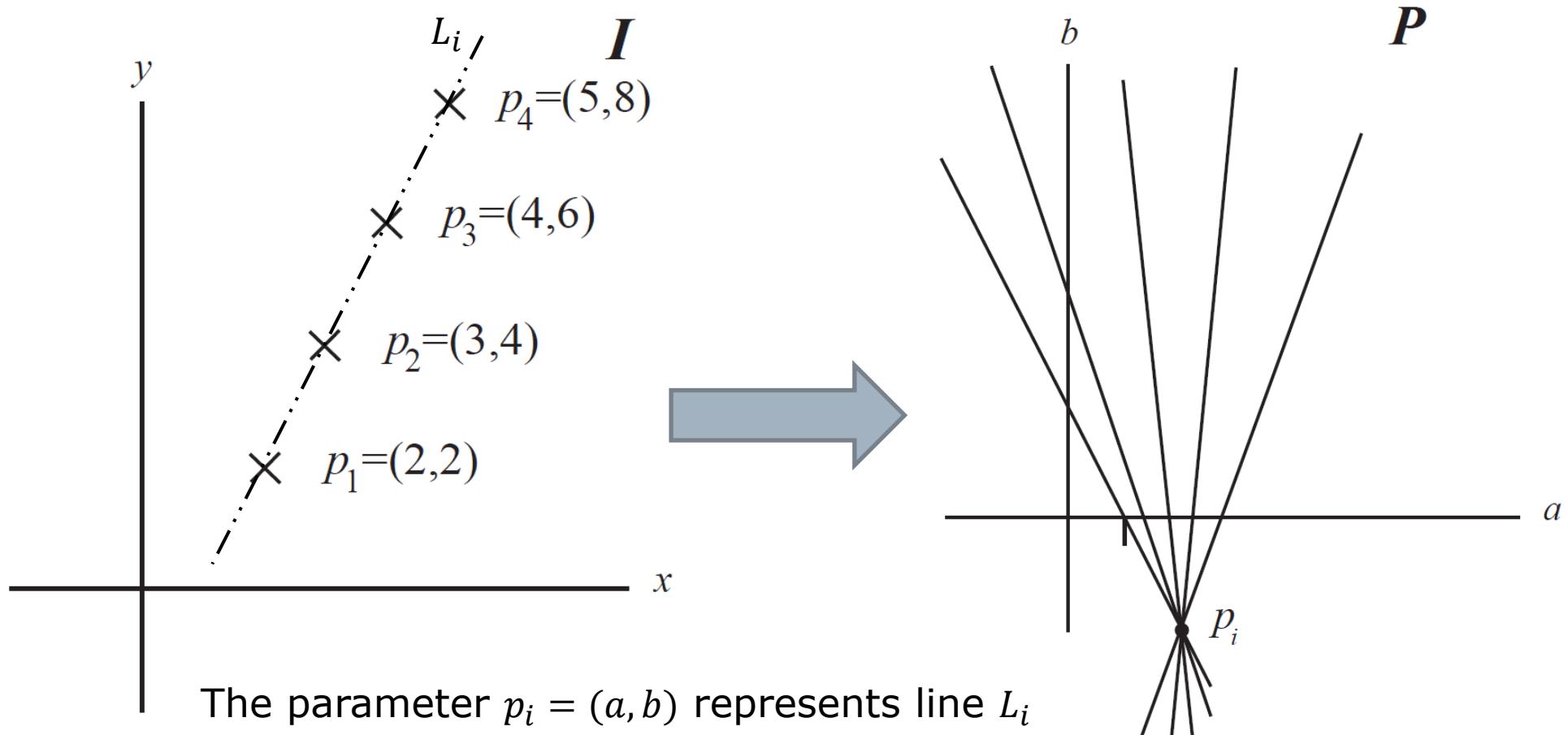
- The relationship between lines and points in two spaces is given as

$$L_i \in I \Leftrightarrow p_i \in P$$

$$p_i \in I \Leftrightarrow L_i \in P$$



Hough Transform - Lines



Suggest a data structure to find $p_i.. ???$

Hough Transform - Lines

□ Algorithm

1. For each $p_i \in I$, the associated $L_i \in P$ is calculated
2. Every point $p_j \in P$ that intersects with L_i has its associated bin values incremented
 - Following this, bins with high values represent possible lines in I
3. The parameter space P is thresholded, and the line equations are extracted
 - Highest peak in parameter space is best line in image space
 - Second highest peak is next best line, etc.

Hough Transform - Lines

□ Pros

- Robust: tolerant to dropouts and occlusions
 - The line doesn't have to be connected or continuous
 - robust to outliers
- Efficient: $O(mn)$
 - With n is number of projected points (features) in image
 - m is the maximum length of a line in the parameters space
 - Peak detection is achieved with a linear search through parameter space

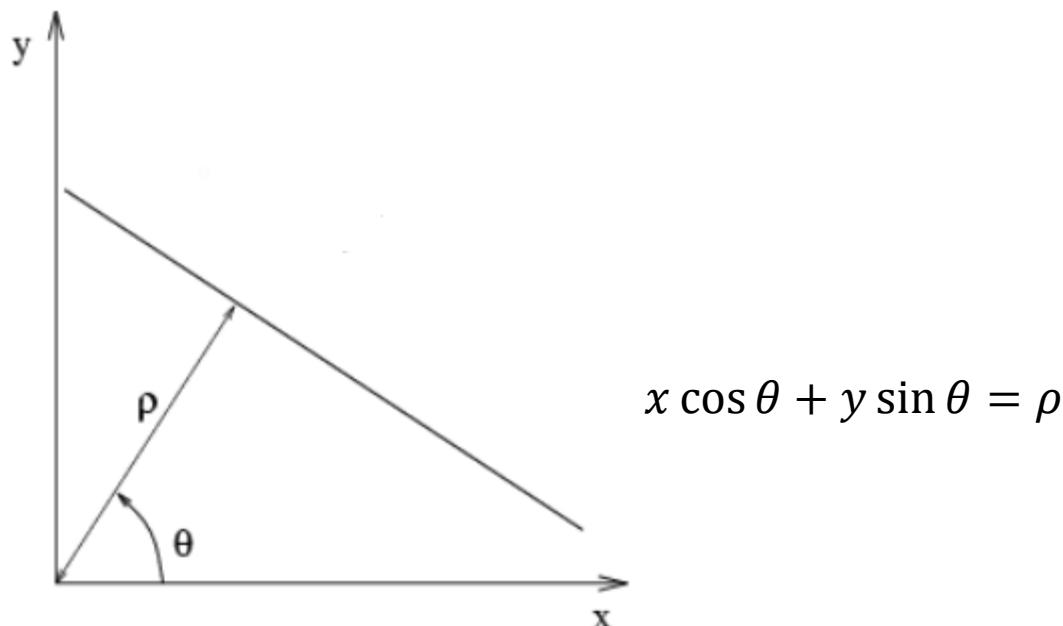
□ Cons

- $a, b \in [-\infty, \infty]$, not possible to bin over this entire space
- Vertical lines require infinite a

What is the solution?

Hough Transform - Lines

- Use Polar representation
 - A line $L \in I$ is represented uniquely by the perpendicular distance ρ from the line L to the origin and angle θ between ρ and the x -axis.

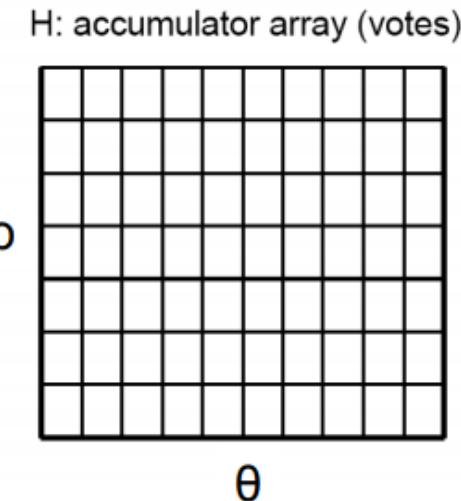


Hough Transform - Lines

- Under the polar parametrization, a point $p \in I$ maps to a sinusoidal curve in P . The intersection of a family of such sinusoids in P still represents a line in I
- The main advantage of the polar parametrization is that both ρ and θ are bounded and can be uniformly quantized over their finite domains

Algorithm Outline

- Initialize accumulator H to all zeros
- For each edge point (x, y) in the image
 - For $\theta = 0$ to 180
 - $\rho = x \cos \theta + y \sin \theta$
 - $H(\theta, \rho) = H(\theta, \rho) + 1$
 - end
- end
- Find the value(s) of (θ, ρ) where $H(\theta, \rho)$ is a local maximum
 - The detected line in the image is given by
$$\rho = x \cos \theta + y \sin \theta$$



Hough Transform - Lines

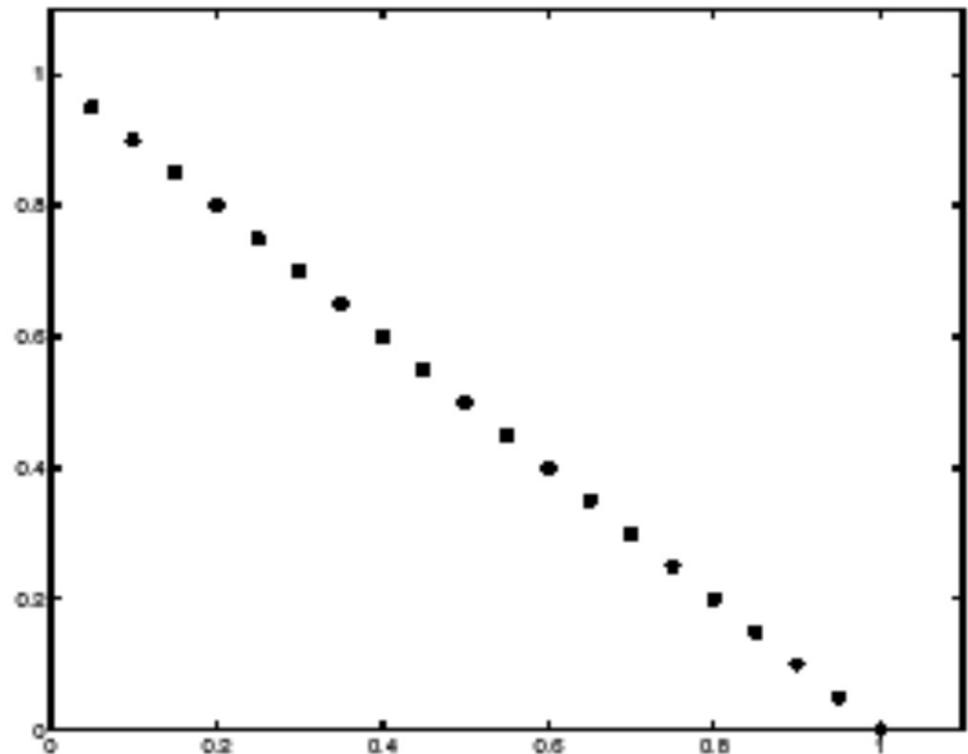
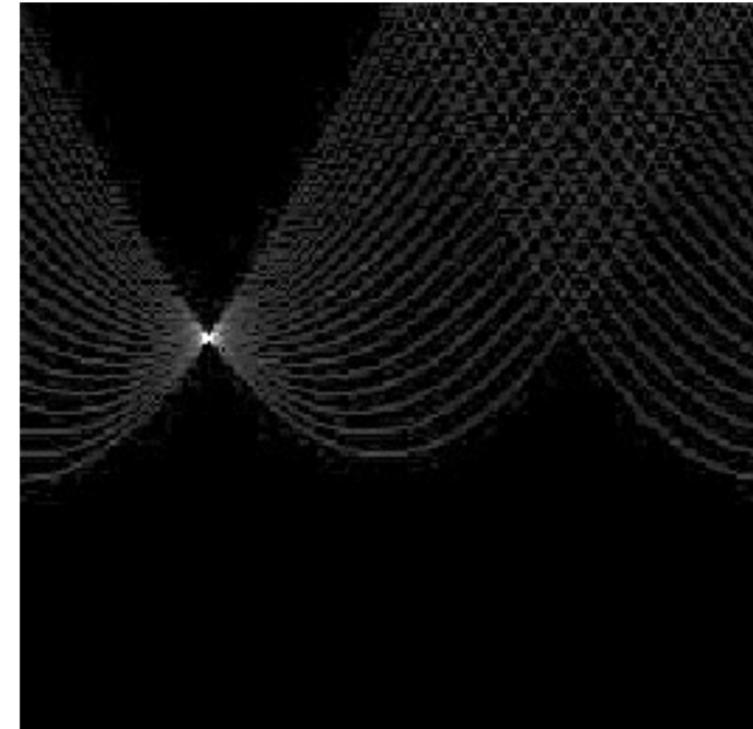


Image space



Votes

In practice, measurements are noisy...

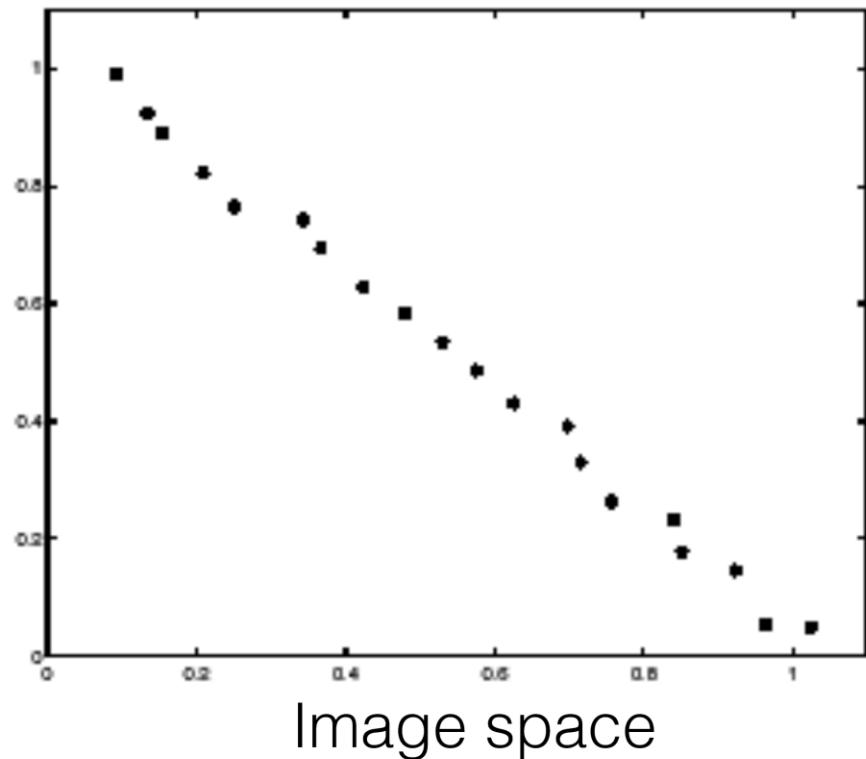
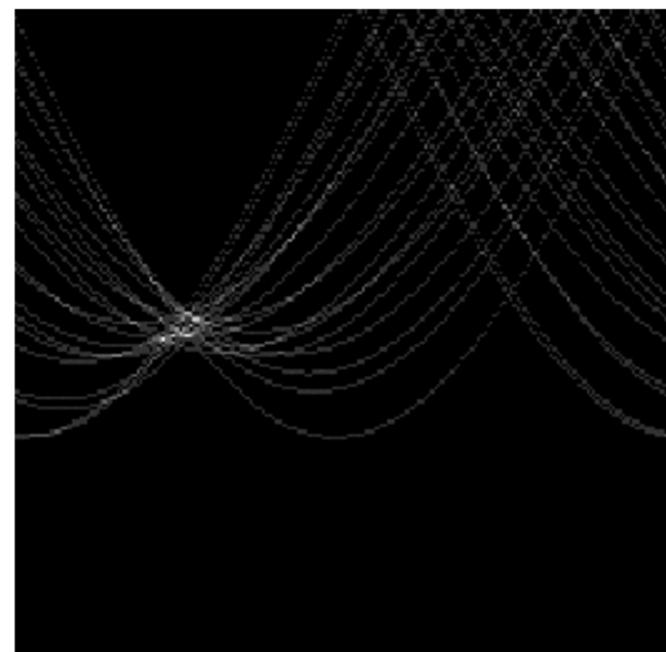


Image space



Votes

Too much noise ...

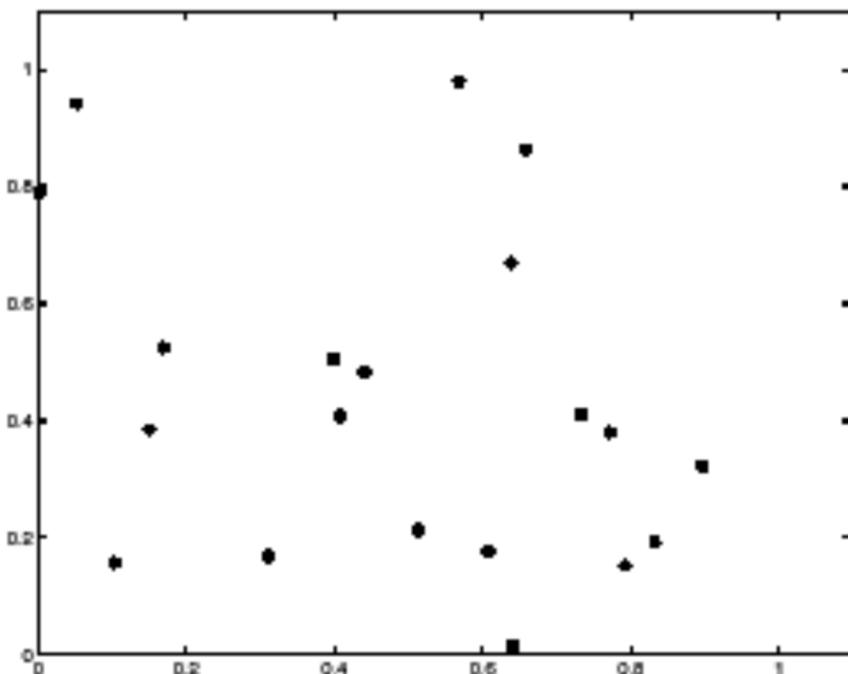
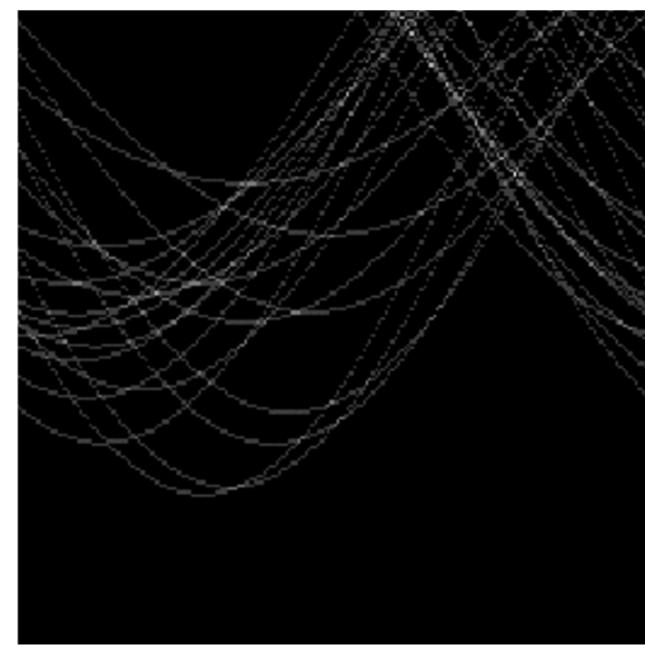


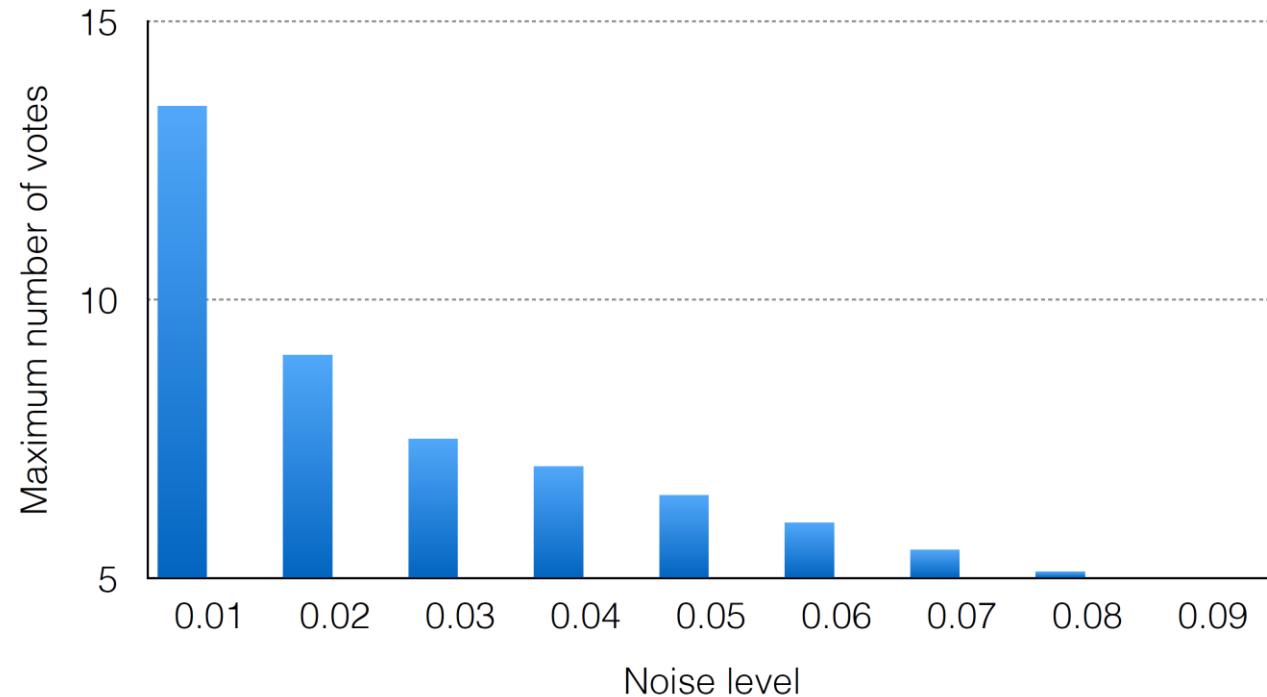
Image space



Votes

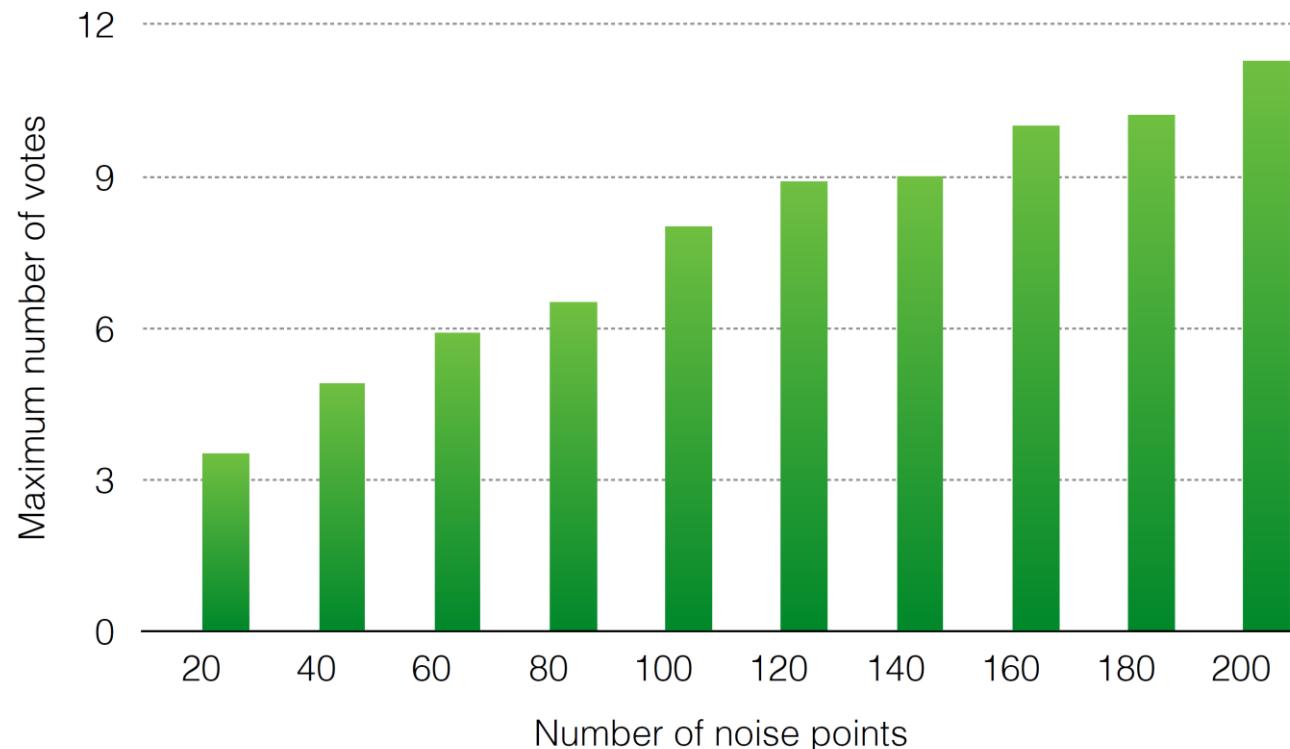
Effect of Noise Levels

Number of votes for a line of 20 points with increasing noise



More noise, less votes (in the right bin)

Effect of Noise Points

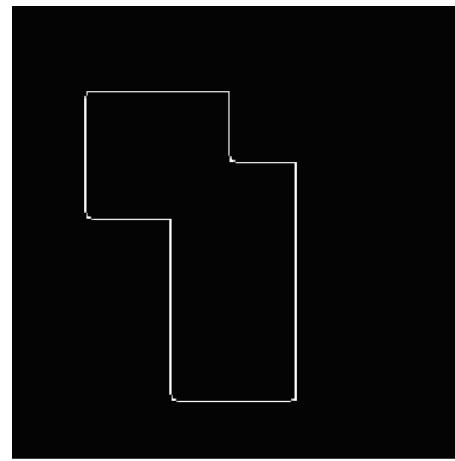
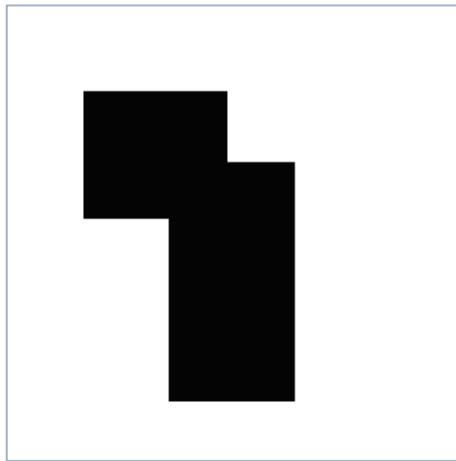


More noise, more votes (in the wrong bin)

Hough Transform - Lines

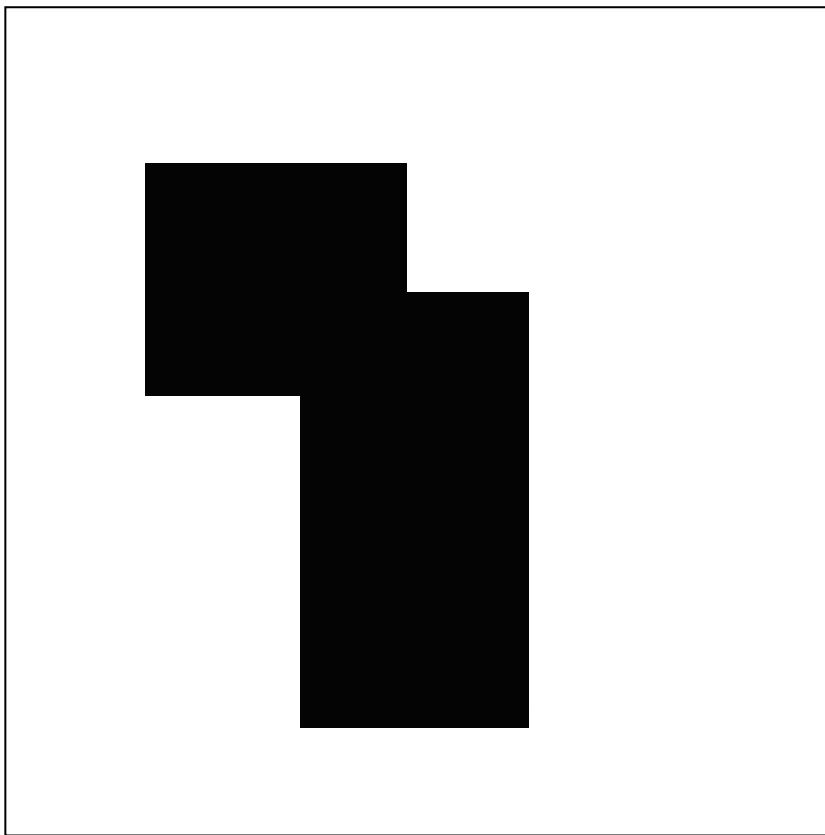
□ Synthetic Image

- Overlapping Rectangles
- Extracted Edges (Canny)
- Polar Parameter Space Mapping

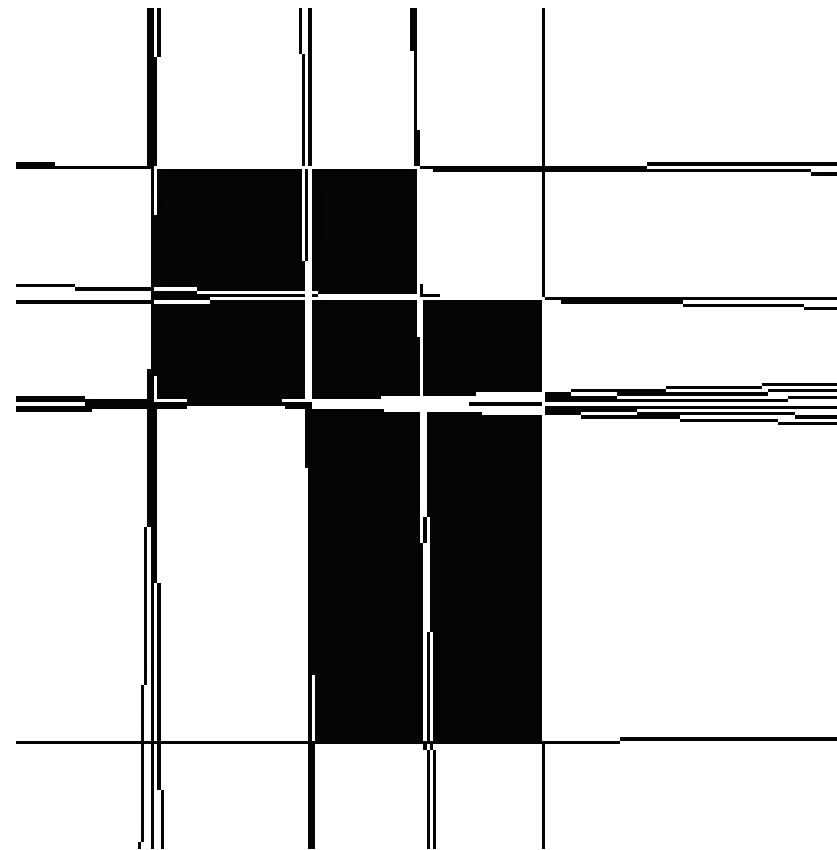


Hough Transform - Lines

Original Image

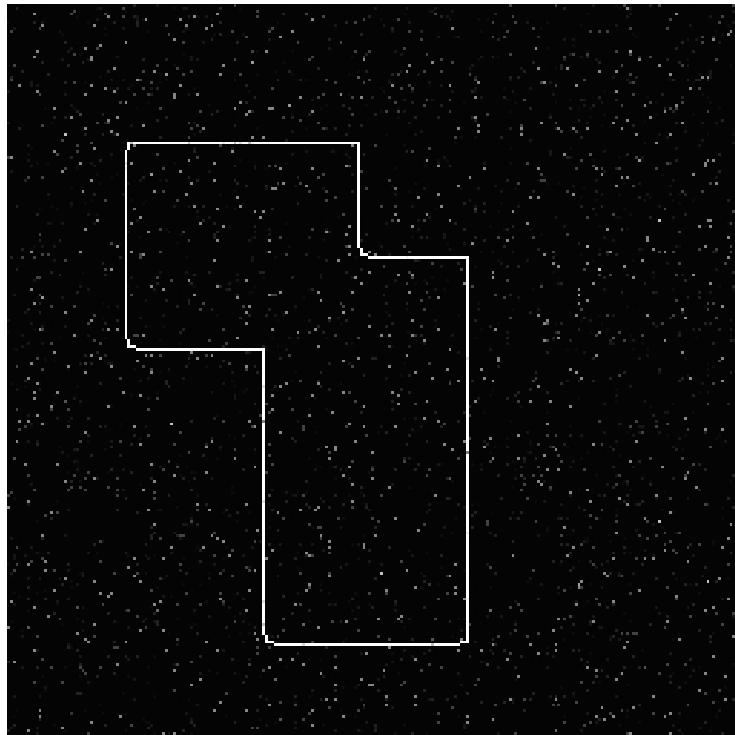


Output with lines

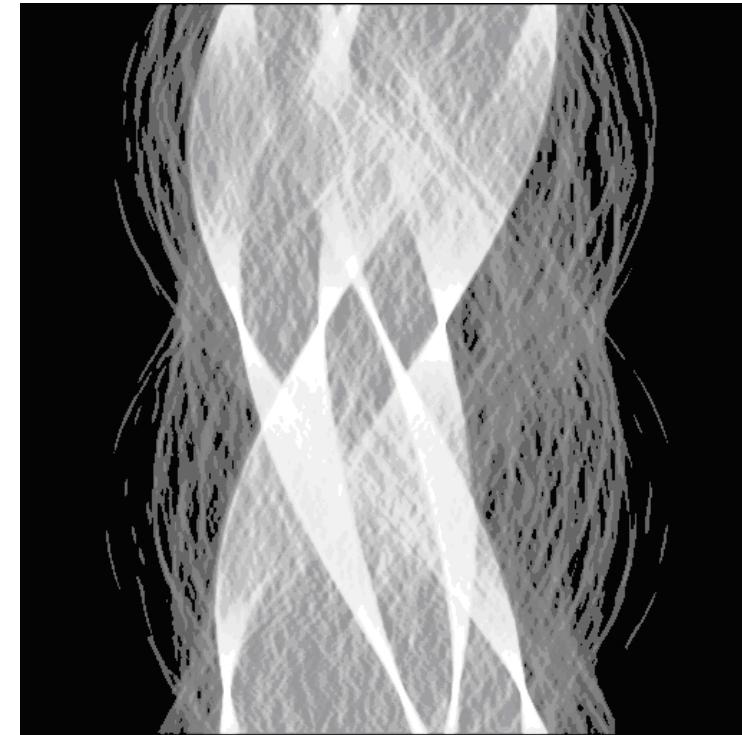


Hough Transform - Lines

Added Salt & Pepper Noise to
edge image

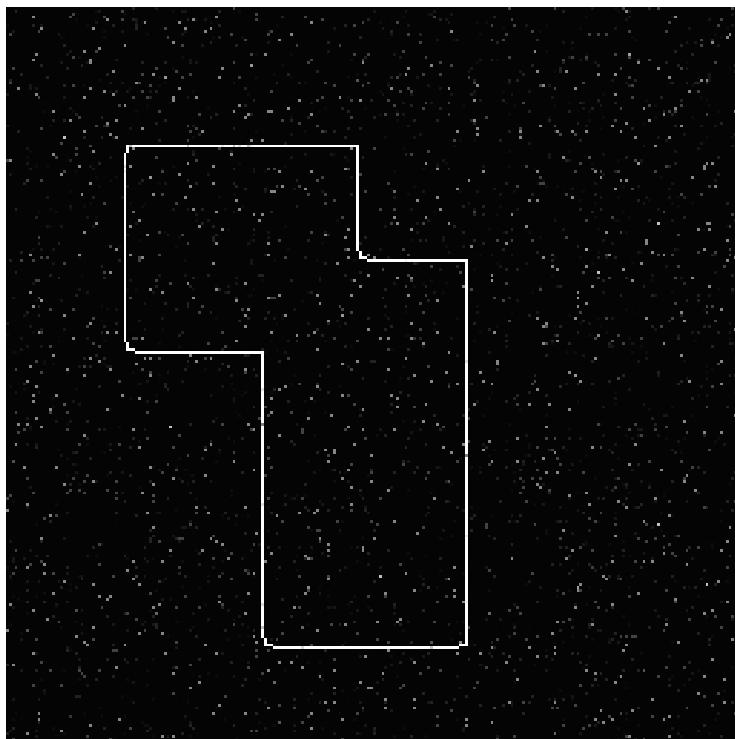


Polar Parameters

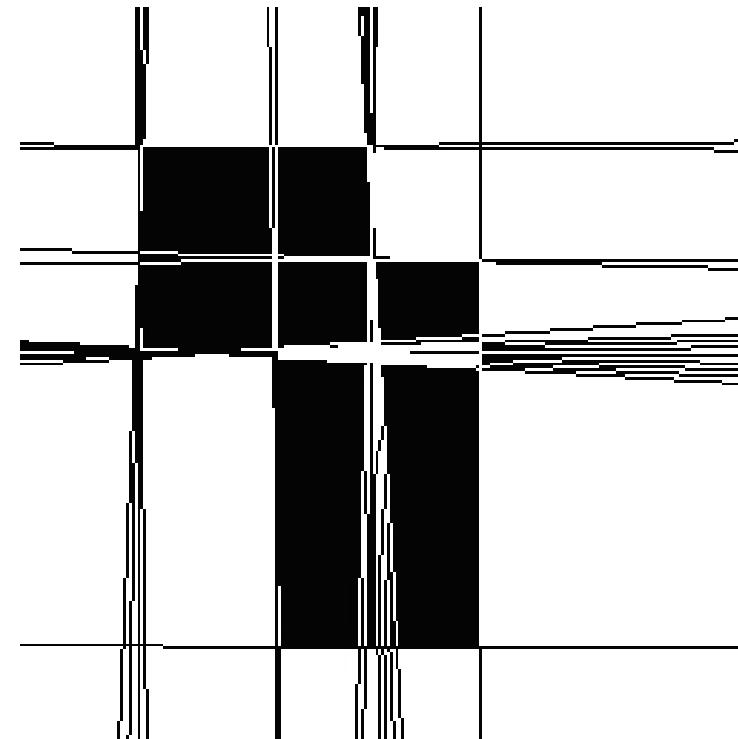


Hough Transform - Lines

Noisy edges

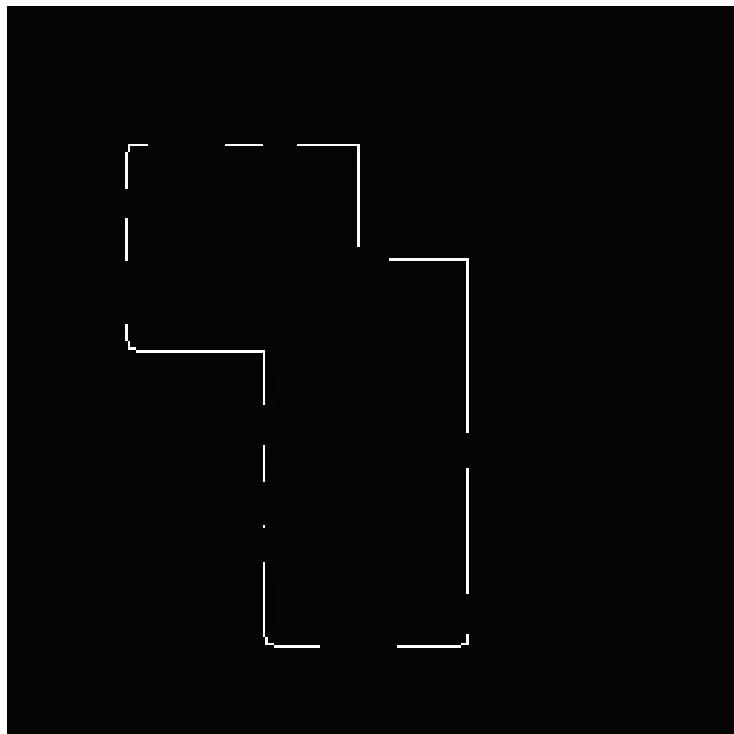


Output

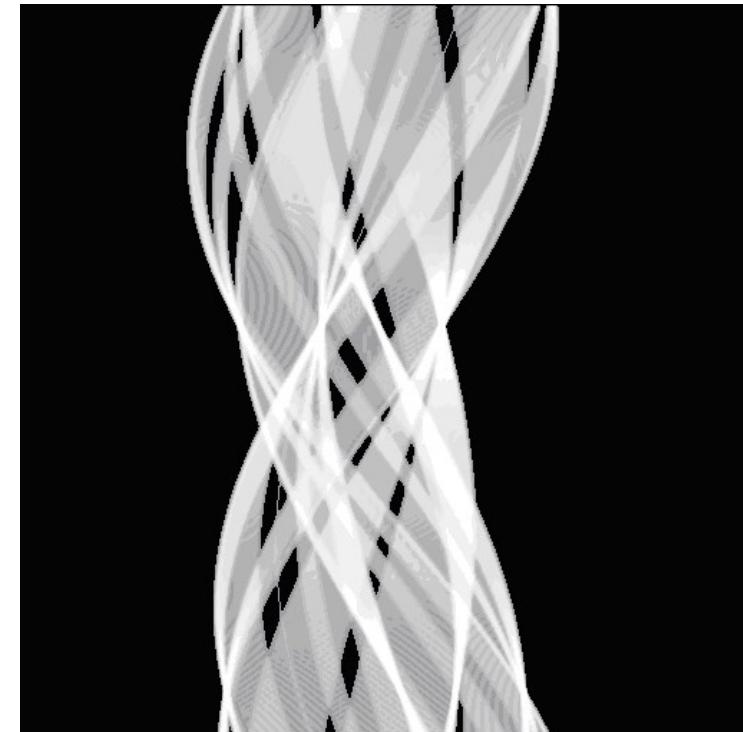


Hough Transform - Lines

Extracted Edges (Canny)
with Dropouts

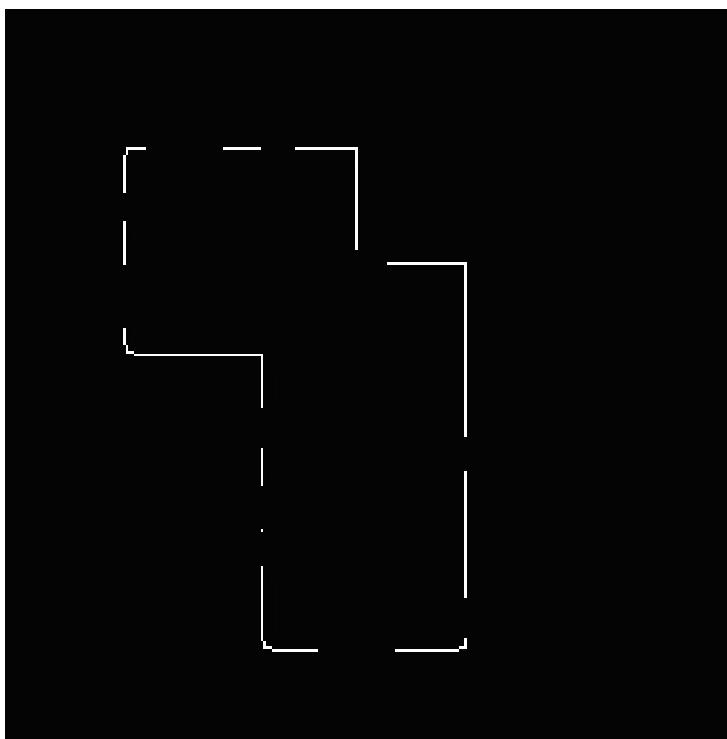


Polar Coordinate

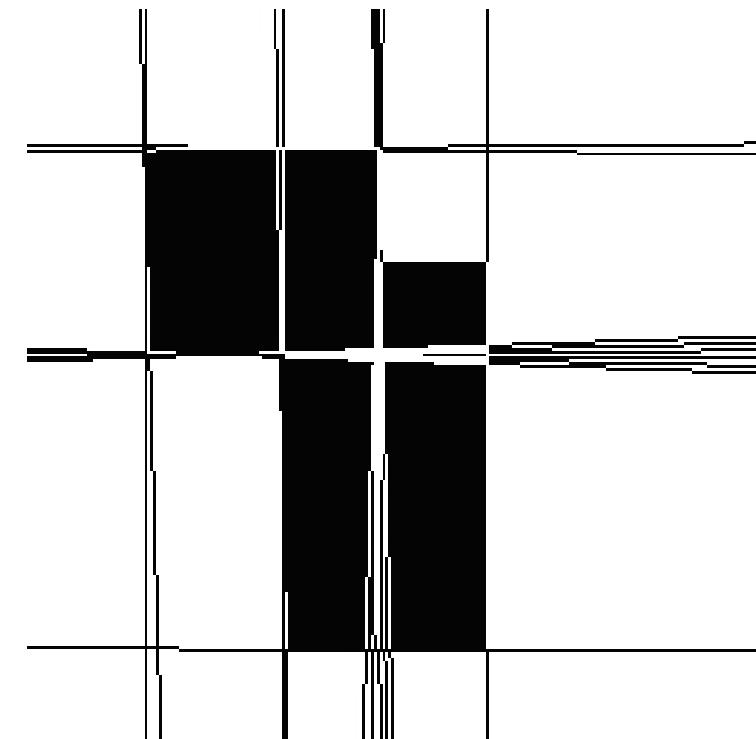


Hough Transform - Lines

Extracted Edges (Canny)
with Dropouts

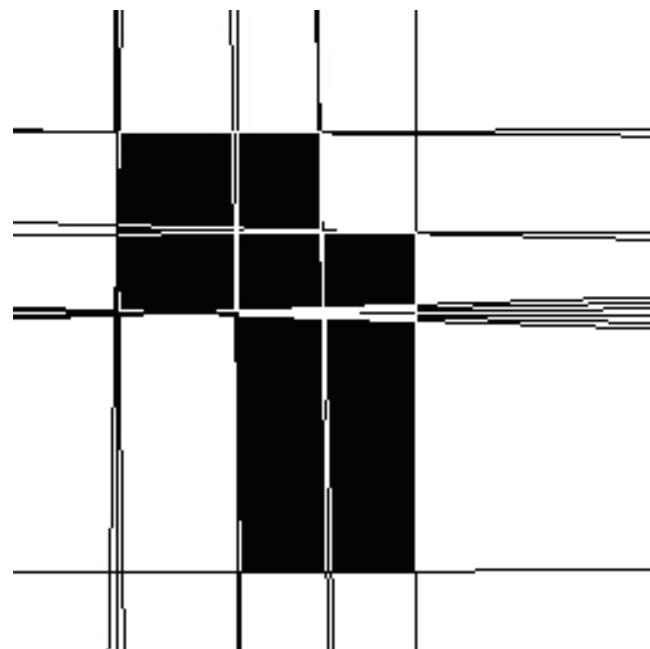


Output

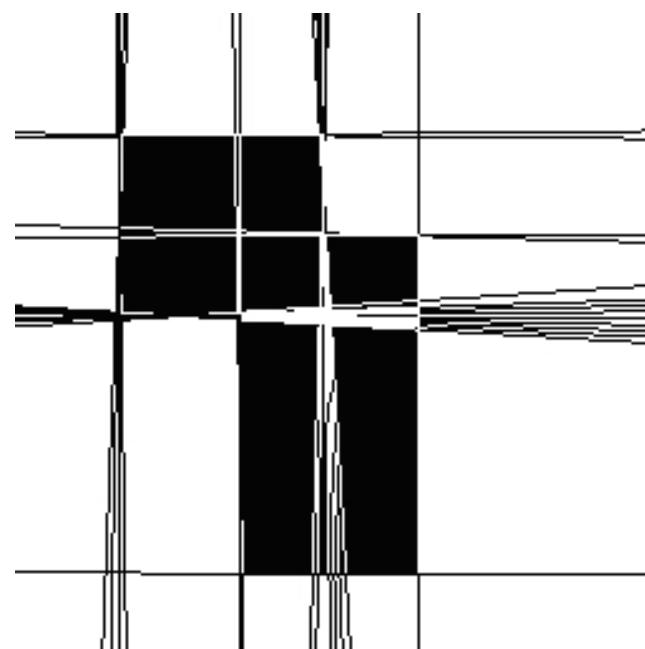


Hough Transform - Lines

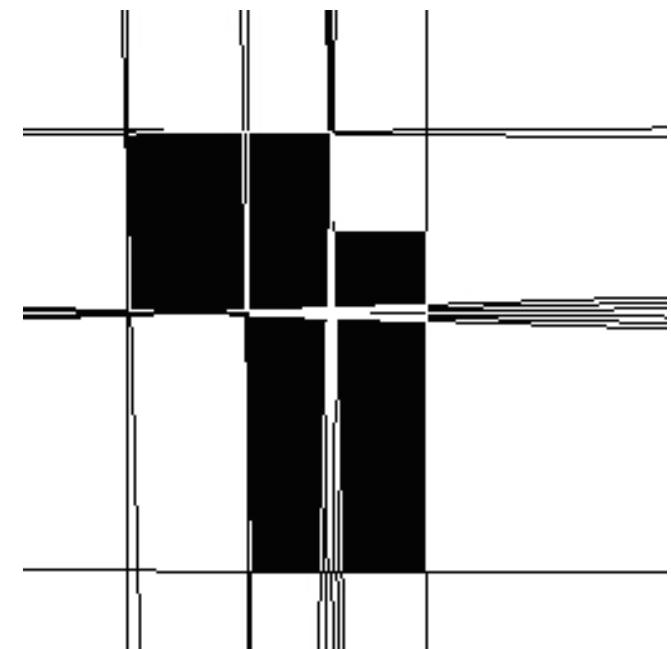
Output without
Noise



Output with
Noise



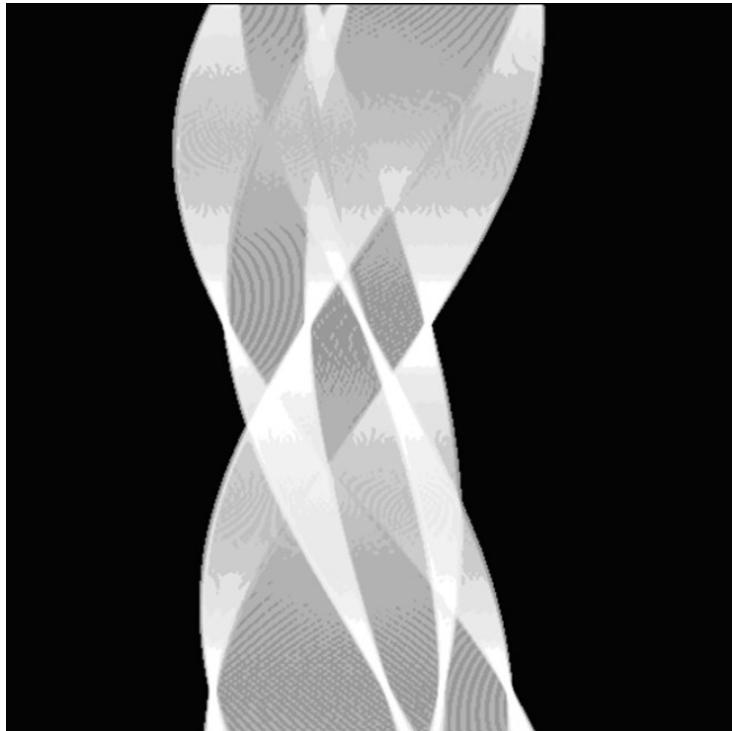
Output with
Dropouts



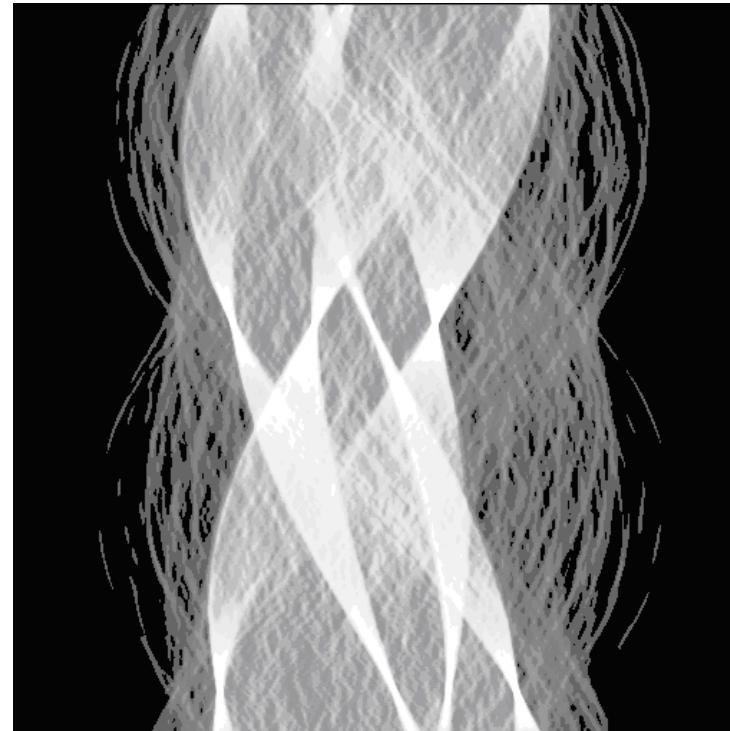
Hough Transform - Lines

Polar Coordinates

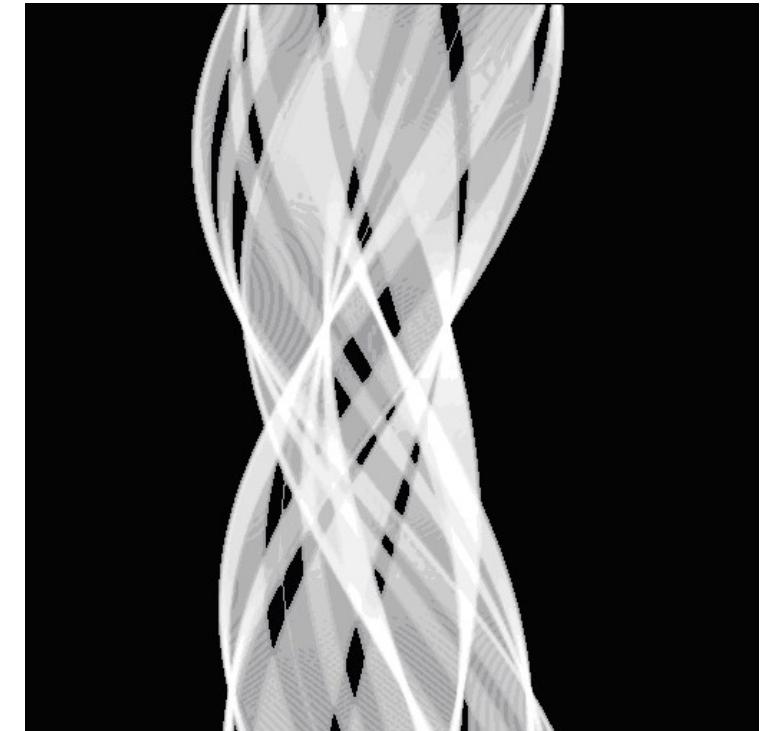
Without Noise

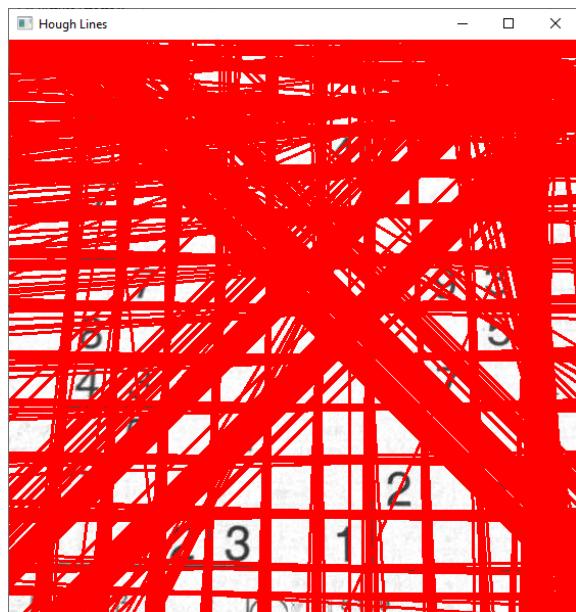
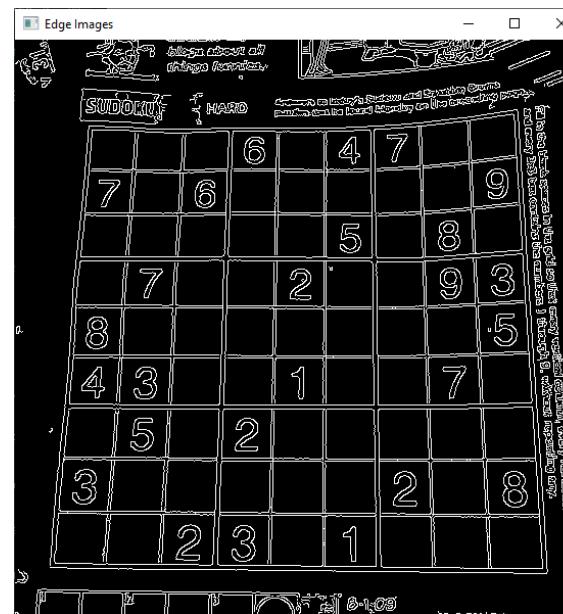


With Noise

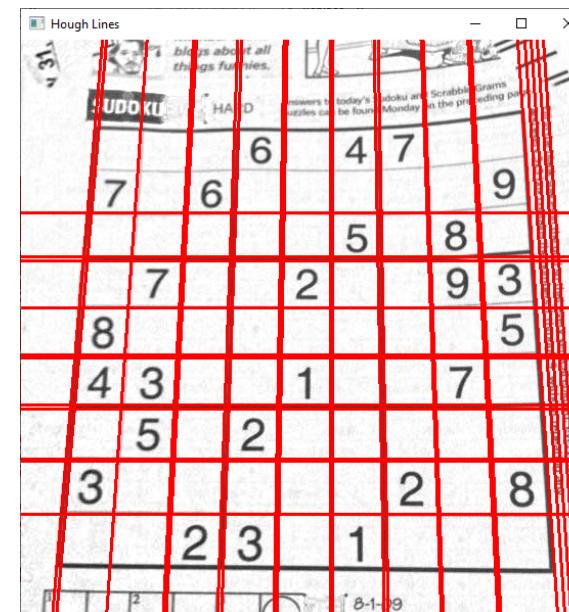


With Dropouts

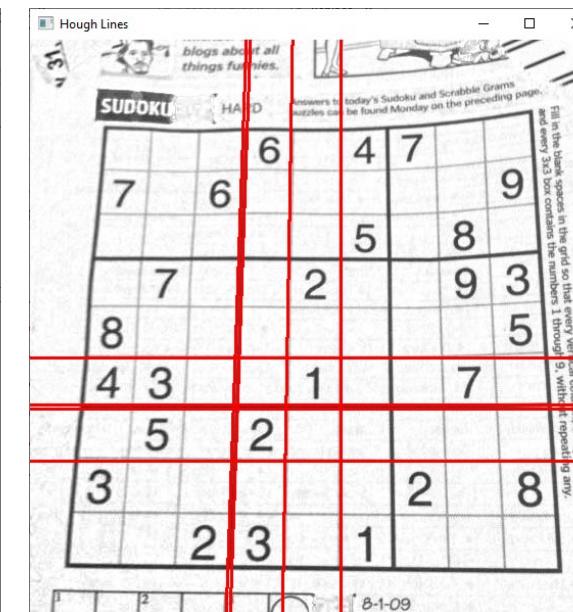




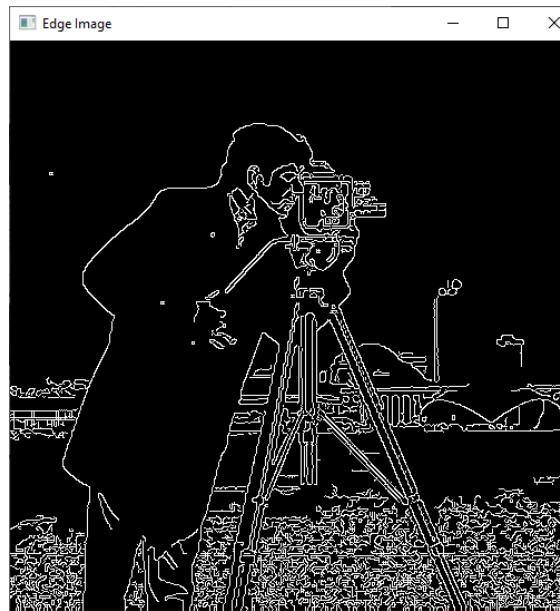
threshold=100



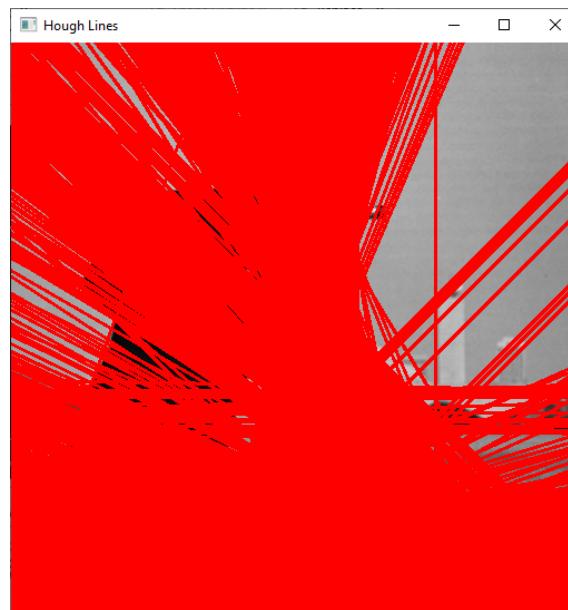
threshold=200



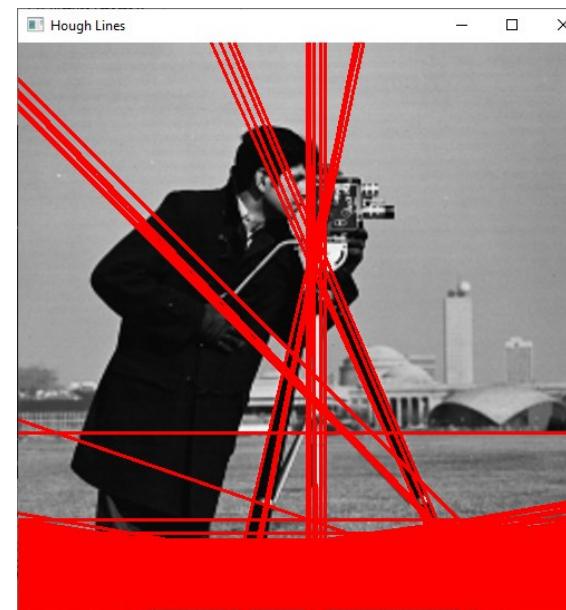
threshold=300



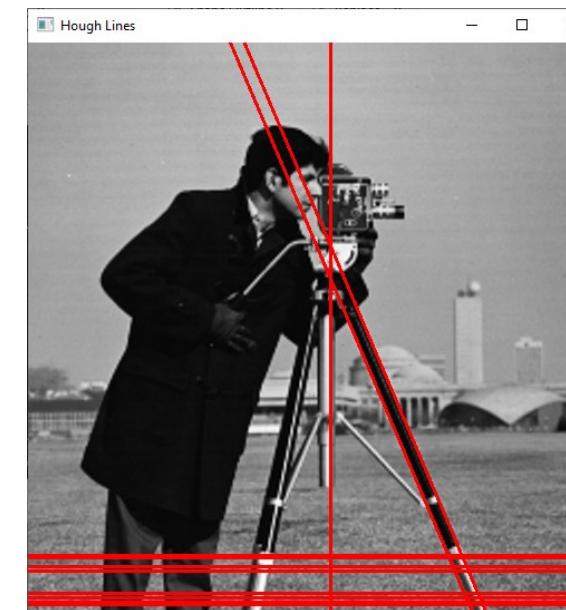
Canny Threshold low = 50
Canny Threshold high = 150



threshold=100



threshold=150



threshold=200



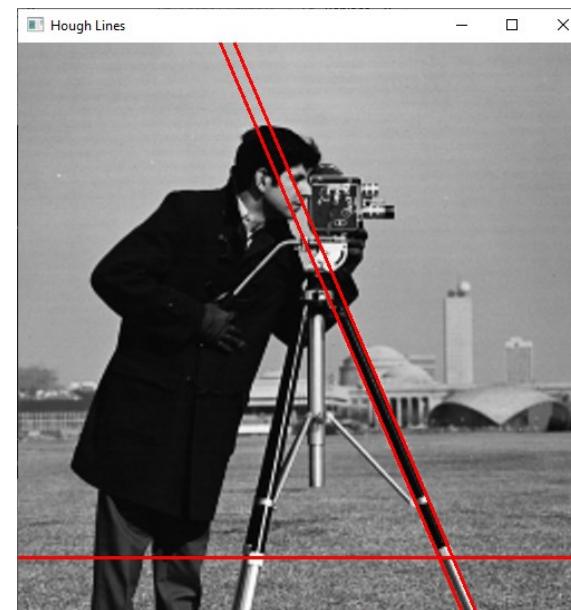
Canny Threshold low = 100
Canny Threshold high = 200



threshold=100



threshold=150



threshold=200

Hough Transform - Circle

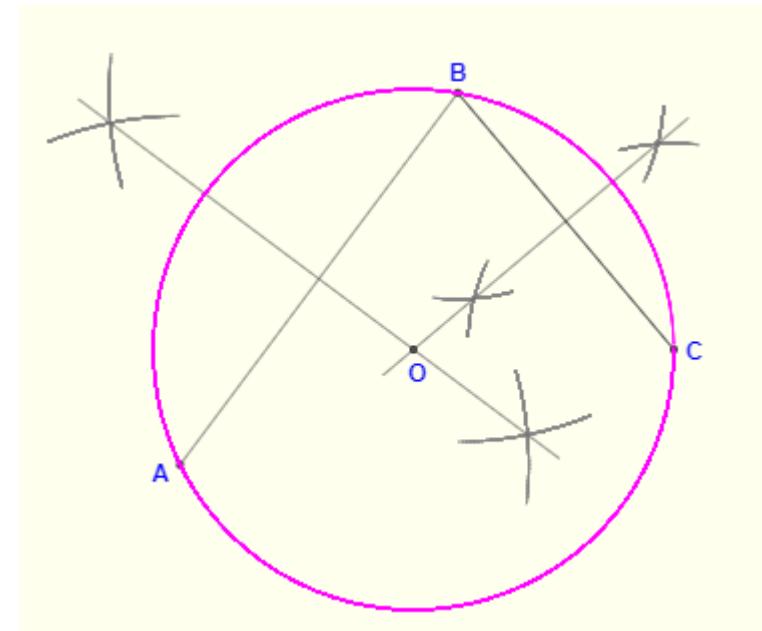
- The Hough transform can be extended to handle other types of geometric primitives.
- After lines, the next most popular primitives to extract are circles.
- There are two main changes to apply Hough transform to circles.
 - The first change is that the parameter space is 3-D rather than 2-D. The 3 axes of parameter space represent the x and y coordinates of the center of a circle, and the radius r .
 - The second required modification is that, whereas in the case of lines a vote was cast into parameter space for each image point, for circles, three points are required to generate each vote.

Hough Transform - Circle

- The method proceeds as follows:
 1. Randomly choose 3 non-collinear points $\{p_1, p_2, p_3\}$ from I .
 2. Generate an expression for the unique circle that intersects with $\{p_1, p_2, p_3\}$, and project its center and radius values $\{(x_c, y_c), r\}$ into P , incrementing the bin that contains the parameter set.
 - There are multiple ways to find $\{(x_c, y_c), r\}$ from $\{p_1, p_2, p_3\}$ (see next slides)
- Repeat Steps 1 and 2 until all possible sets of 3 non-collinear points have been selected.
 1. Threshold P to determine the peaks.

Hough Transform - Circle

- How to compute center and radius, using three non-collinear points?
 - Method 1:
Use perpendicular bisectors



Hough Transform - Circle

- How to compute center and radius, using three non-collinear points?
 - Method 2:

Use matrices and circle equation in general form
<https://www.johndcook.com/blog/2023/06/18/circle-through-three-points/>

Hough Transform - Circle

- There are two problems with the given formation
 1. Two or more of the three selected points are too close, the estimate of the circle parameters can be numerically inaccurate.
 - To avoid this, votes are generated only if the randomly selected three points are sufficiently separated.
 2. There will in general be far too many combinations of three points in an edge map to enumerate them all exhaustively. For n points, there are

$$\frac{n(n-1)(n-2)}{6} = O(n^3) \text{ triplets.}$$

- For example, for $n = 10^4$ (a reasonably small edge image), then there will be $> 10^{12}$ triples, which is a huge and unmanageable number.

What to do to resolve this?

Randomized Hough Transform

- The Randomized Hough Transform takes a **probabilistic** approach to reduce the computational load.
- Rather than exhaustively generating all possible sets of triplets, the RHT generates only a relatively small number.
- If more image edge pixels fall on the circle, RHT is likely to perform better.

Randomized Hough Transform

- Let p be the likelihood that an edge pixel falls on the circle.
 - The chance of randomly selecting a triplet that falls on the circle is therefore p^3 .
 - If $p = 0.5$, then half of the image edge pixels fall on the circle. The likelihood is that $p^3 = 12.5\%$ of a sufficiently large sample of triples will fall on the circle.
- Randomized Hough Transform is particularly useful when real-time or efficient processing is required, and a high level of accuracy can be sacrificed to some extent.

Generalized Hough Transform

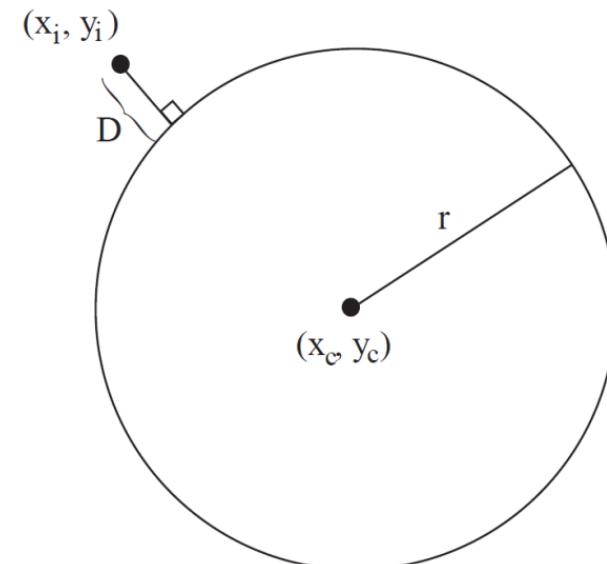
- Hough transform can be used for other parametric shapes
 - For example, an ellipse is defined by 5 points so 5 D parameter space.
 - Templates can also be searched in the images.
 - A template is a set of edge points of the shape.
 - The accumulator space is incremented by aligning the template with each possible image point for all possible transformations (translations and rotations) of the template.

RANSAC - Circle

- The algorithm steps are the same as those for RANSAC-Line.
- For a circle, center (x_c, y_c) and radius r .
- The distance of a point $p_i = (x_i, y_i)$ from the circle is calculated as:

$$D = |\sqrt{(x_i - x_c)^2 + (y_i - y_c)^2} - r|$$

- Points that are within the distance are accumulated as score for the quality of particular model.
- Define termination conditions



RANSAC - Circle

- Algorithm Steps:
 - Pick 3 points
 - Fit a circle
 - Find number of inliers
 - Continue until termination condition occurs
- If the number of elements to extract is known a priori, and a minimum score is known which signifies a correct model, then the algorithm can iterate until all models that achieve higher than the minimum score are found and extracted.
- Alternately, if the number of models is not known a priori, then the algorithm can iterate a fixed number of times, and the maximum scored model can be returned.

Resources and Credits

- "Digital Image Processing (Fourth Edition)", Rafael C. Gonzalez and Richard E. Woods, Prentice Hall, 2017

Image Processing & Pattern Recognition

Feature Detectors and Descriptors Part 1: Feature Detectors

Dr. Zia Ud Din

Outline

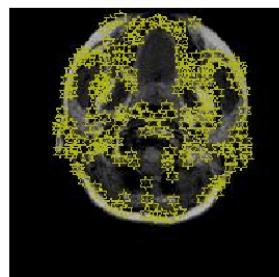
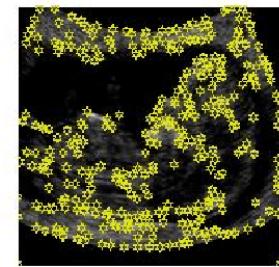
- Introduction
- Harris Corner Detector
- Scale Invariant Feature Transform (SIFT)

Outline

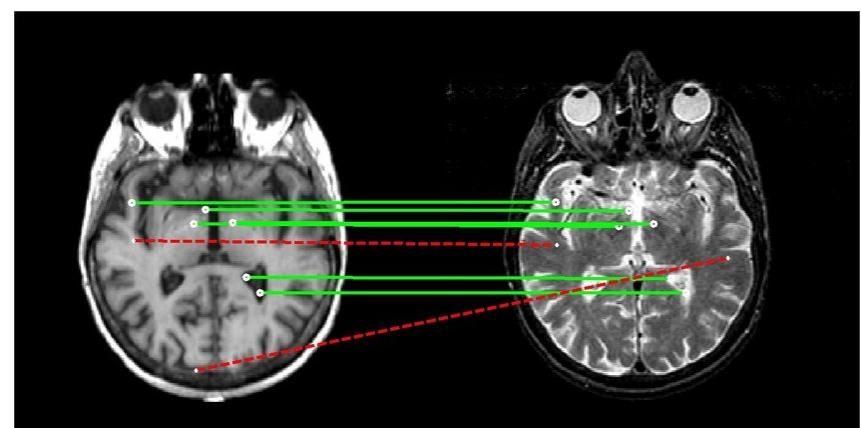
- **Introduction**
- Harris Corner Detector
- Scale Invariant Feature Transform (SIFT)

Keypoint Detectors and Descriptors

- A **keypoint** (also called **interest point** or **feature**) **detector** finds and extracts “keypoints” in an image



- A **keypoint descriptor** describes detected interest points in a manner than makes it feasible to match those across different images and objects



Source: Guohua Lv

Source: Mohamed Ali Hajjaji

Motivation: Features for Recognition

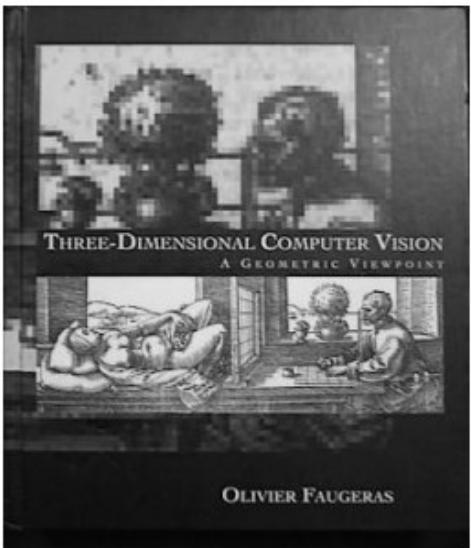


Image search: find the book in an image.

Motivation: Build a Panorama



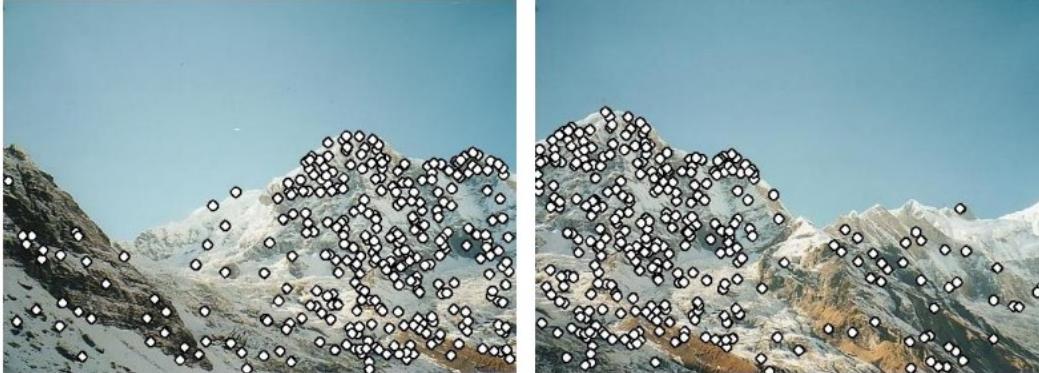
Source: Brown & Lowe 2003

How do we build panorama?

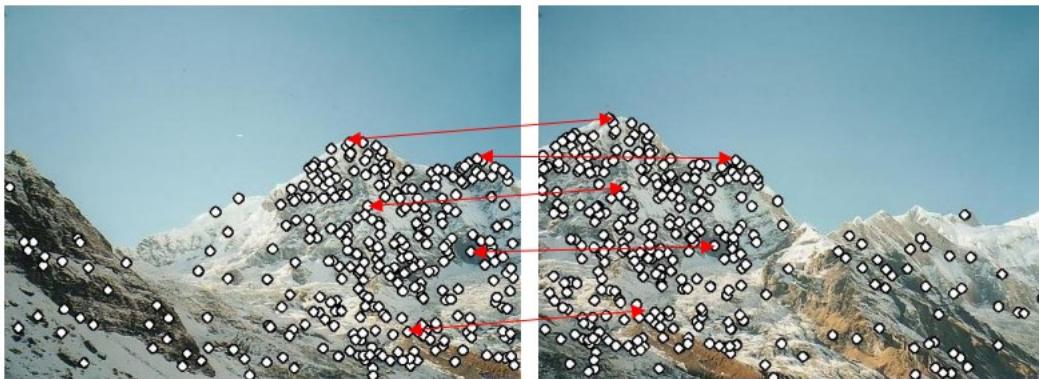
- We need to match (align) images



1. Detect feature points in both images



2. Find corresponding pairs



3. Use these pairs to align images



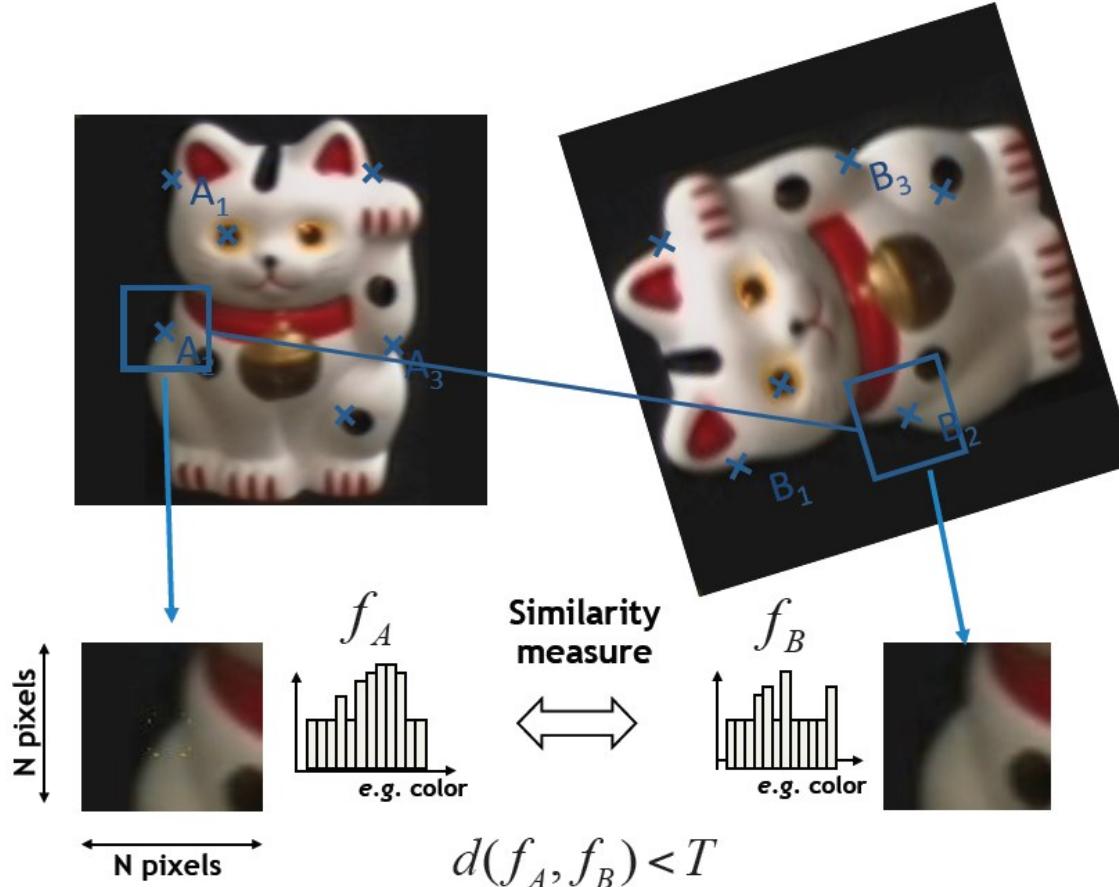
- Feature points are used also for:
 - Point matching for computing disparity
 - Image alignment (homography, fundamental matrix)
 - 3D reconstruction
 - Motion tracking
 - Motion based segmentation
 - Object recognition
 - Image retrieval and indexing
 - Robot navigation
 - ... other

Local Features

- Local vs global features
- Which local/global features have you studied so far?

Local Features

- General Approach
 1. Find a set of distinctive interest points
 2. Define a region around each interest point
 3. Extract and normalize the region content
 4. Compute a local descriptor from the normalized region
 5. Match local descriptors



Source: K. Grauman, B. Leibe

Main Challenges

- Change in position, scale, and rotation
- Change in viewpoint
- Occlusion
- Articulation, change in appearance

Local Features

□ Requirements

1. Detect the same interest points independently in both images

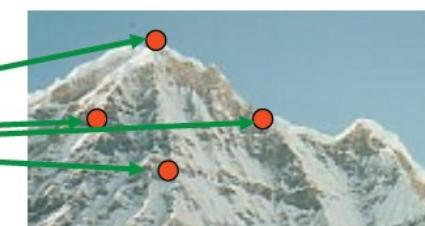
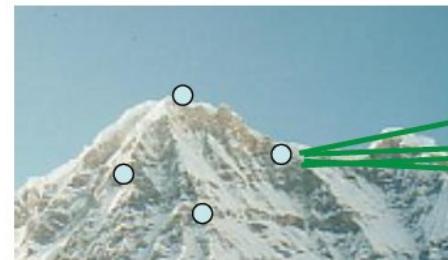
□ We need a **repeatable detector**.



No chance of match!

2. For each interest point correctly recognize the corresponding one

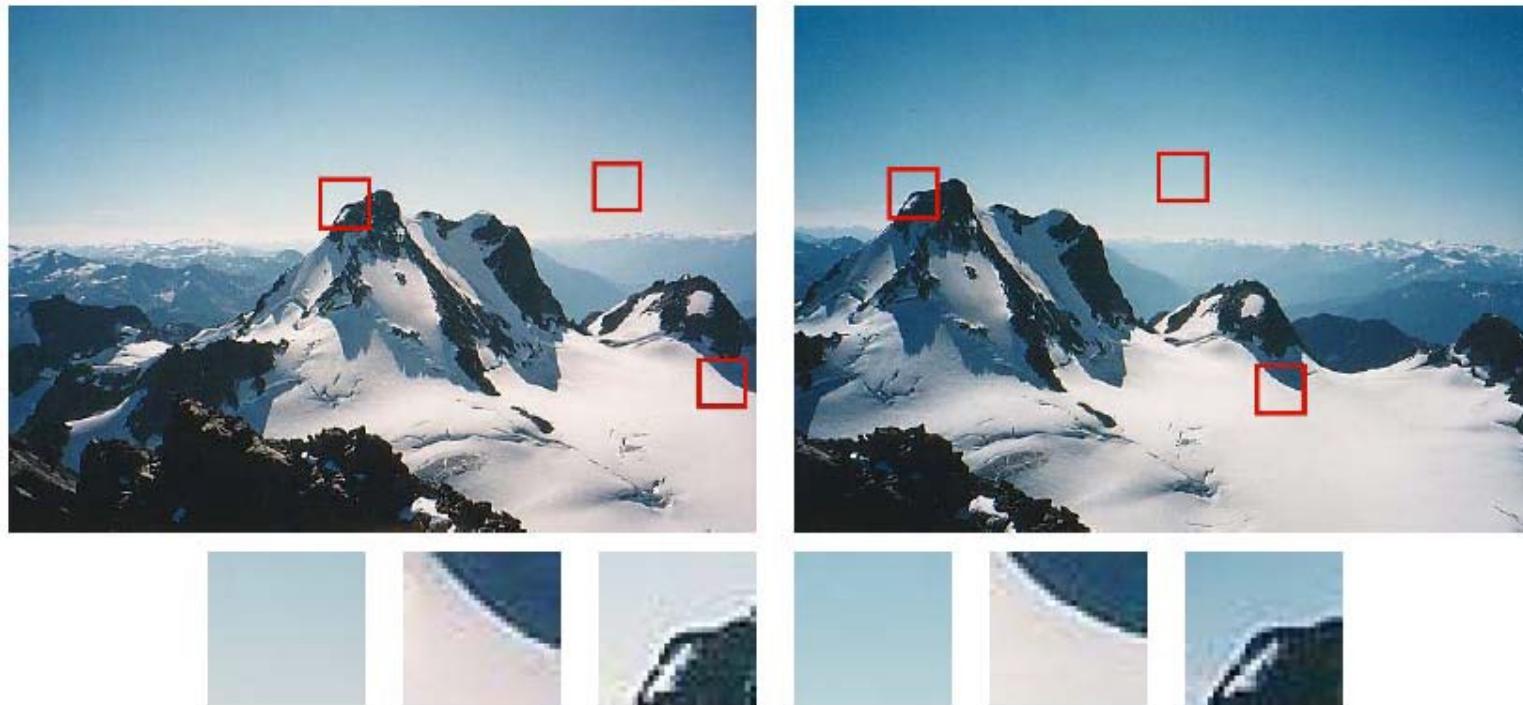
□ We need a **reliable and distinctive descriptor**.



Source: Kristen Grauman

Local Features

- Some patches can be localized or matched with higher accuracy than others

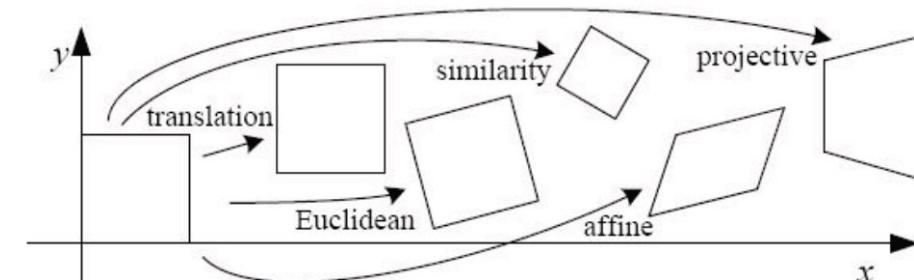
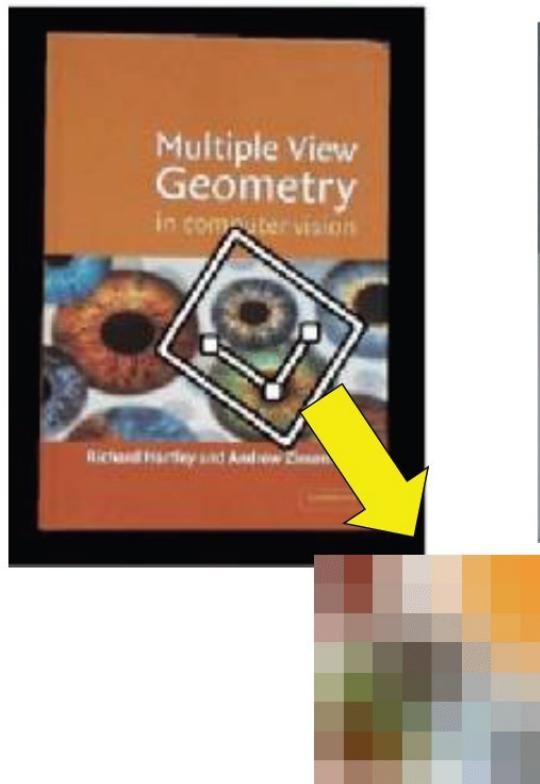


Properties of the ideal feature

- **Local**: features are local, so robust to occlusion and clutter (no prior segmentation)
- **Invariant**: (or covariant)
- **Robust**: noise, blur, discretization, compression, etc. do not have a big impact on the feature
- **Distinctive**: individual features can be matched to a large database of objects
- **Quantity**: many features can be generated for even small objects
- **Accurate**: precise localization
- **Efficient**: close to real-time performance

Invariance

Invariant to geometric transform



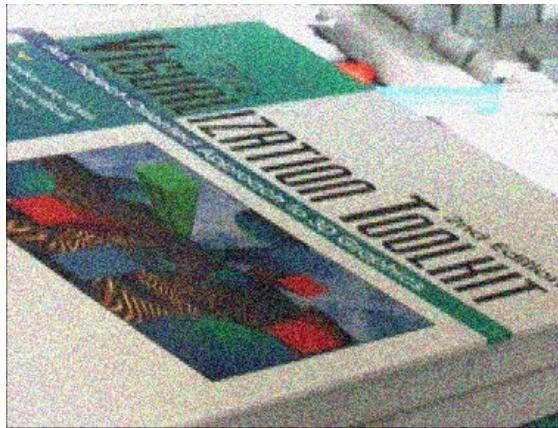
Level of geometric invariance

Source: Steve Seitz, Bastian Leibe

Contents

- Introduction
- Harris Corner Detector
- Scale Invariant Feature Transform (SIFT)

Invariance



Photometric Transformation



Source: Tinne Tuytelaars

Feature Detectors

- There are many feature detector available
 - Harris Corner detector
 - Scale Invariant Feature Transform - SIFT
 - Speeded up robust features - SURF
 - Features from accelerated segment test - FAST
 - Binary Robust Independent Elementary Features - BRIEF
 - Oriented FAST and Rotated BRIEF - ORB

Harris Corner Detector

- C. Harris and M. Stephens. "A Combined Corner and Edge Detector." Proceedings of the 4th Alvey Vision Conference, 1988

A COMBINED CORNER AND EDGE DETECTOR

Chris Harris & Mike Stephens
Plessey Research Roke Manor, United Kingdom
© The Plessey Company plc. 1988

Consistency of image edge filtering is of prime importance for 3D interpretation of image sequences using feature tracking algorithms. To cater for image regions containing texture and isolated features, a combined corner and edge detector based on the local auto-correlation function is utilised, and it is shown to perform with good consistency on natural imagery.

INTRODUCTION

The problem we are addressing in Alvey Project MMI149 is that of using computer vision to understand the unconstrained 3D world, in which the viewed scenes will in general contain too wide a diversity of objects for top-down recognition techniques to work. For example, we desire to obtain an understanding of natural scenes, containing roads, buildings, trees, bushes, etc., as typified by the two frames from a sequence illustrated in Figure 1. The solution to this problem that we are pursuing is to use a computer vision system based upon motion analysis of a monocular image sequence from a mobile camera. By extraction and tracking of image features, representations of the 3D analogues of these features can be constructed.

To enable explicit tracking of image features to be performed, the image features must be discrete, and not form a continuum like texture, or edge pixels (edgels). For this reason, our earlier work¹ has concentrated on the extraction and tracking of feature-points or corners, since

they are discrete, reliable and meaningful². However, the lack of connectivity of feature-points is a major limitation in our obtaining higher level descriptions, such as surfaces and objects. We need the richer information that is available from edges³.

THE EDGE TRACKING PROBLEM

Matching between edge images on a pixel-by-pixel basis for stereo, because of the known epi-polar camera geometry. However for the motion problem, where the camera motion is unknown, the aperture problem prevents us from undertaking explicit edgel matching. This could be overcome by solving for the motion beforehand, but we are still faced with the task of tracking each individual edge pixel and estimating its 3D location from, for example, Kalman Filtering. This approach is unattractive in comparison with assembling the edgels into edge segments, and tracking these segments as the features.

Now, the unconstrained imagery we shall be considering will contain both curved edges and texture of various scales. Representing edges as a set of straight line fragments⁴, and using these as our discrete features will be inappropriate, since curved lines and texture edges can be expected to fragment differently on each image of the sequence, and so be untrackable. Because of ill-conditioning, the use of parametrised curves (eg. circular arcs) cannot be expected to provide the solution, especially with real imagery.



a

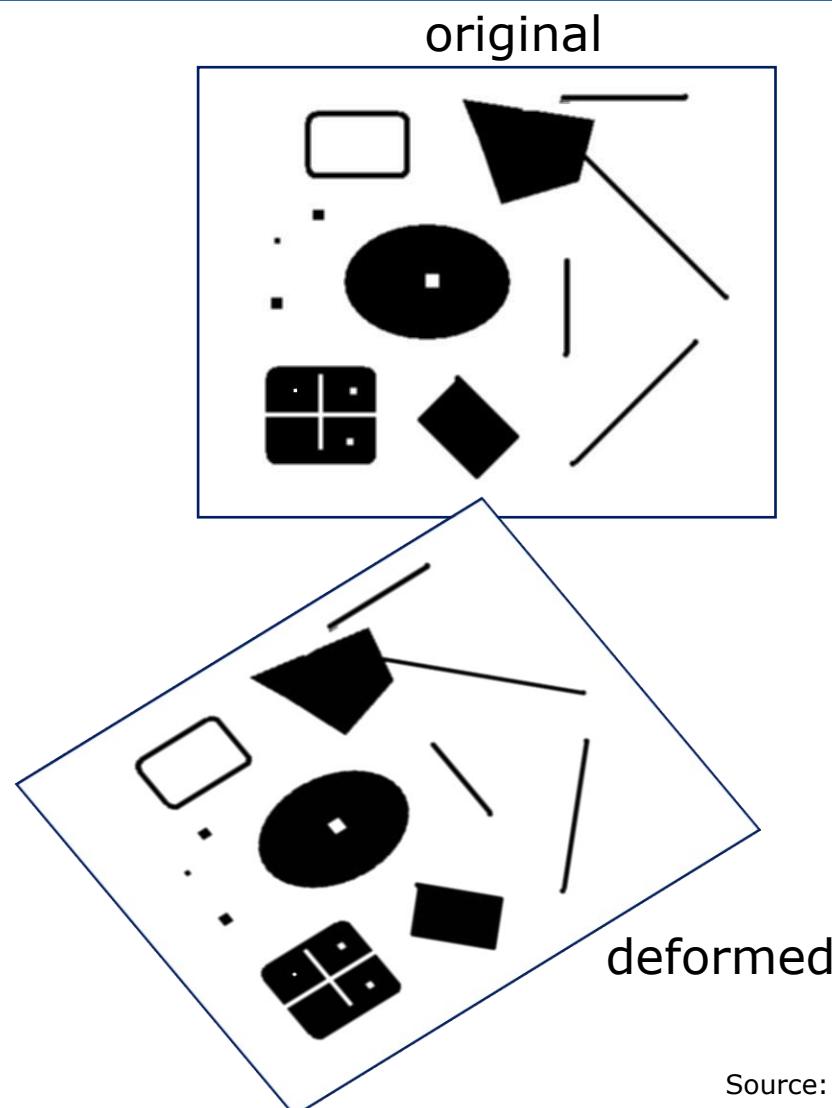


b

Figure 1. Pair of images from an outdoor sequence.

Harris Corner Detector

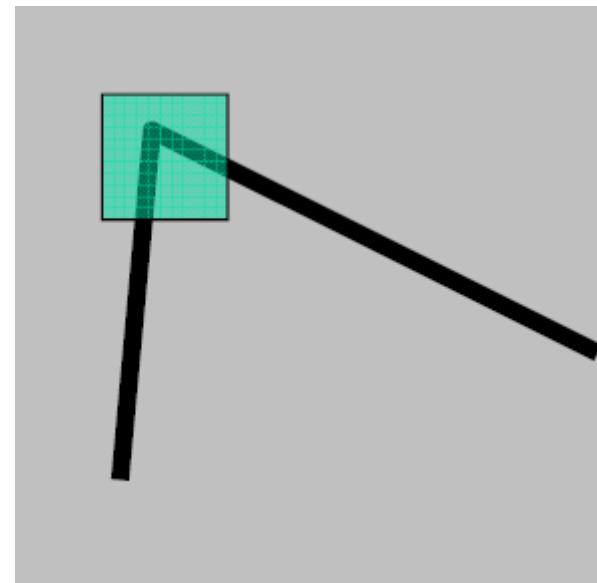
- Why corners as interest points?
 - Corners are *repeatable* and *distinctive*



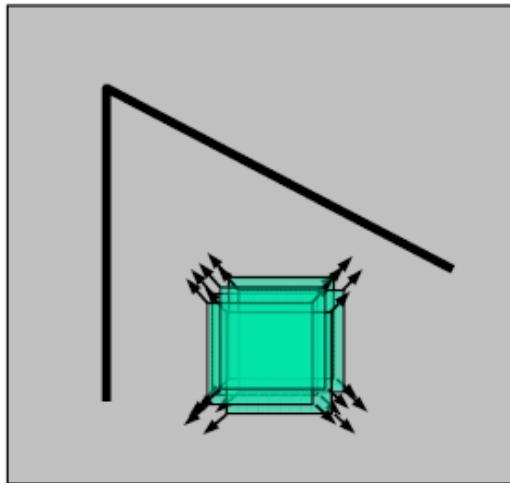
Source: James Hays

Harris Corner Detector

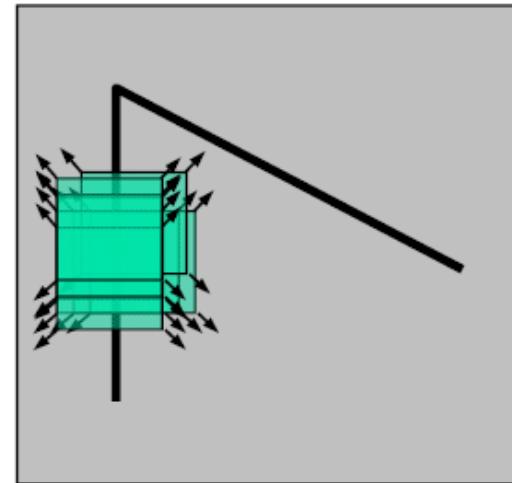
- Corner point can be recognized in a window
- Shifting a window in any direction should give a large change in intensity



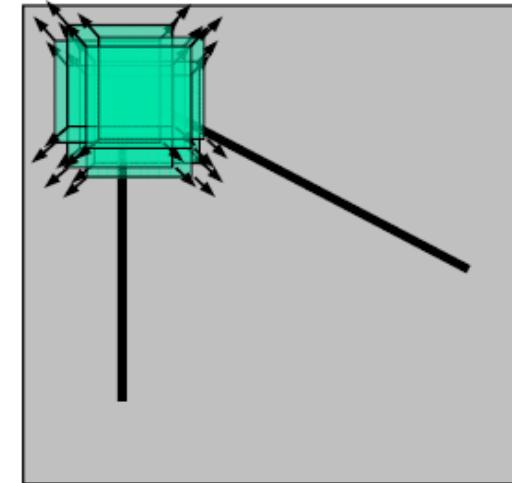
Basic Idea



“flat” region:
no change in
all directions



“edge”:
no change along
the edge direction



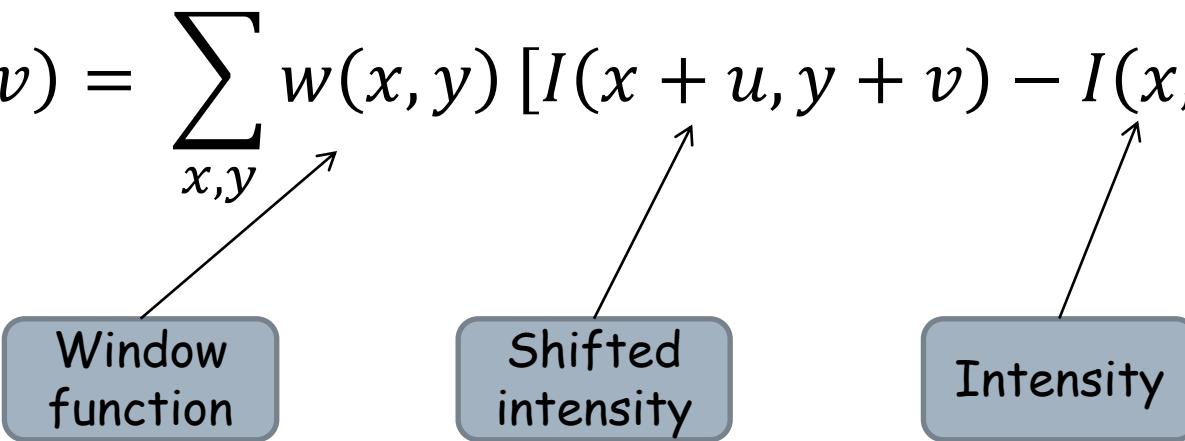
“corner”:
significant change
in all directions

Source: Mubarak Shah

Harris Detector Formulation

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

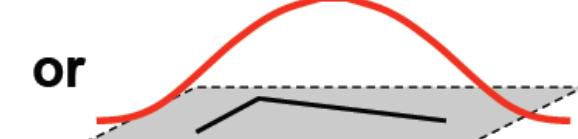
$$E(u, v) = \sum_{x,y} w(x, y) [I(x + u, y + v) - I(x, y)]^2$$


Window function Shifted intensity Intensity

Window function $w(x,y) =$



1 in window, 0 outside



Gaussian

Harris Detector Formulation

$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{window function}} \left[\underbrace{I(x + u, y + v)}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}} \right]^2$$
$$E(u, v) = \sum_{x,y} \underbrace{w(x, y)}_{\text{window function}} \left[\underbrace{I(x, y) + uI_x + vI_y}_{\text{shifted intensity}} - \underbrace{I(x, y)}_{\text{intensity}} \right]^2 \quad \text{Taylor Series}$$
$$E(u, v) = \sum_{x,y} w(x, y) [uI_x + vI_y]^2$$
$$E(u, v) = \sum_{x,y} w(x, y) \begin{bmatrix} u & v \end{bmatrix} \begin{pmatrix} I_x \\ I_y \end{pmatrix}^2$$
$$E(u, v) = \sum_{x,y} w(x, y) (u - v) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \begin{pmatrix} I_x & I_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$
$$E(u, v) = (u - v) \left[\sum_{x,y} w(x, y) \begin{pmatrix} I_x \\ I_y \end{pmatrix} \begin{pmatrix} I_x & I_y \end{pmatrix} \right] \begin{pmatrix} u \\ v \end{pmatrix} \quad E(u, v) = (u - v) M \begin{pmatrix} u \\ v \end{pmatrix}$$

Taylor Series

- It is a series expansion of a function $f(x)$ about a point a in terms of its derivatives

$$f(a) + \frac{f'(a)}{1!}(x - a) + \frac{f''(a)}{2!}(x - a)^2 + \frac{f'''(a)}{3!}(x - a)^3 + \dots,$$

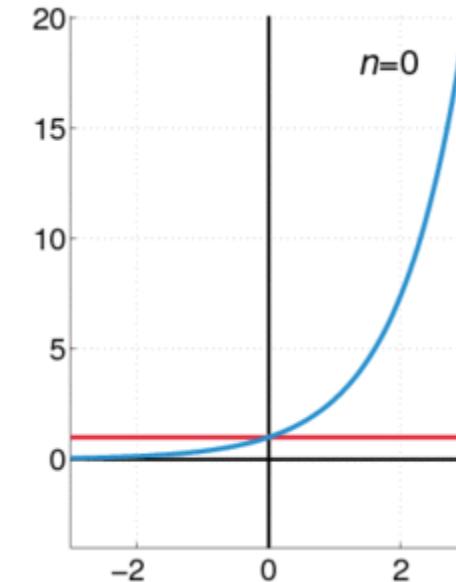
- Example:

Exponential function [edit]

The exponential function e^x (with base e) has Maclaurin series

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

It converges for all x .



The [exponential function](#) e^x (in blue), and the sum of the first $n + 1$ terms of its Taylor series at 0 (in red).

Harris Detector Formulation

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

Sum over image region – the area we are checking for corner

Gradient with respect to x , times gradient with respect to y

$$M = \begin{bmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} [I_x \ I_y]$$

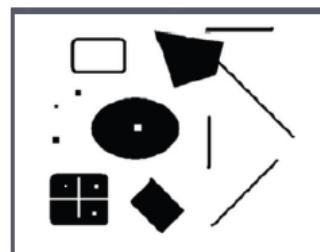
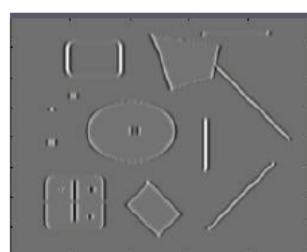


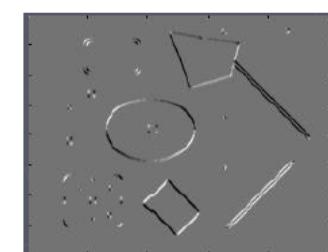
Image I



I_x



I_y



$I_x I_y$

Harris Detector Formulation

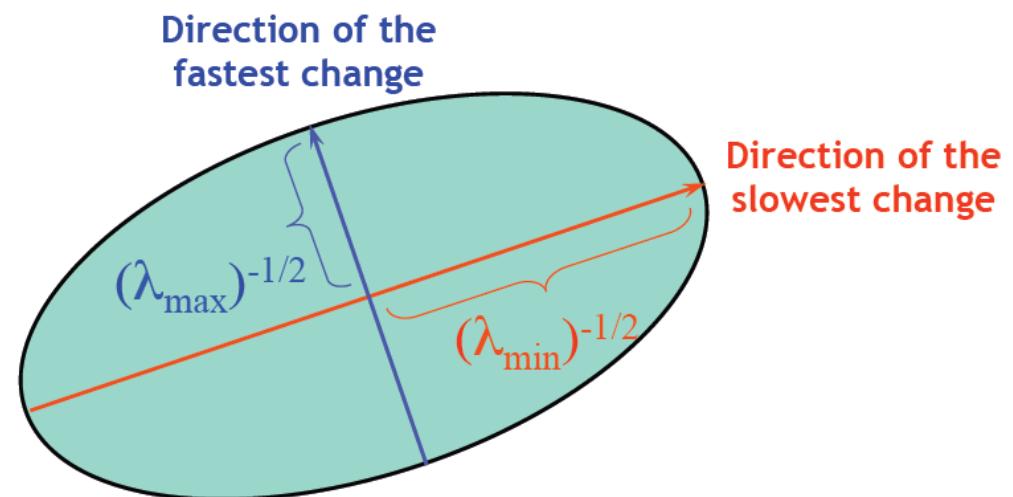
□ General Case

- Since M is symmetric, we can decompose M using eigenvalue decomposition.

$$M = A \Lambda A^{-1}$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

- We can visualize M as an ellipse with axis lengths determined by the eigenvalues Λ and orientation determined by A .



Eigen Vectors and Eigen Values

The eigen vector, x , of a matrix A is a special vector, with the following property

$$Ax = \lambda x \quad \text{Where } \lambda \text{ is called eigen value}$$

To find eigen values of a matrix A first find the roots of:

$$\det(A - \lambda I) = 0$$

Then solve the following linear system for each eigen value to find corresponding eigen vector

$$(A - \lambda I)x = 0$$

Example

$$A = \begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix}$$

Eigen Values

$$\lambda_1 = 7, \lambda_2 = 3, \lambda_3 = -1$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 4 \\ 4 \end{bmatrix}, \mathbf{x}_2 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}, \mathbf{x}_3 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Eigen Vectors

Eigen Values

$$\det(A - \lambda I) = 0$$

$$\det\left(\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) = 0$$

$$\det\left(\begin{bmatrix} -1-\lambda & 2 & 0 \\ 0 & 3-\lambda & 4 \\ 0 & 0 & 7-\lambda \end{bmatrix}\right) = 0$$

$$(-1-\lambda)((3-\lambda)(7-\lambda)-0) = 0$$

$$(-1-\lambda)(3-\lambda)(7-\lambda) = 0$$

$$\lambda = -1, \quad \lambda = 3, \quad \lambda = 7$$

Eigen Vectors

$$\lambda = -1$$

$$(A - \lambda I)x = 0$$

$$\begin{bmatrix} -1 & 2 & 0 \\ 0 & 3 & 4 \\ 0 & 0 & 7 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 2 & 0 \\ 0 & 4 & 4 \\ 0 & 0 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$0 + 2x_2 + 0 = 0$$

$$0 + 4x_2 + 4x_3 = 0$$

$$0 + 0 + 8x_3 = 0$$

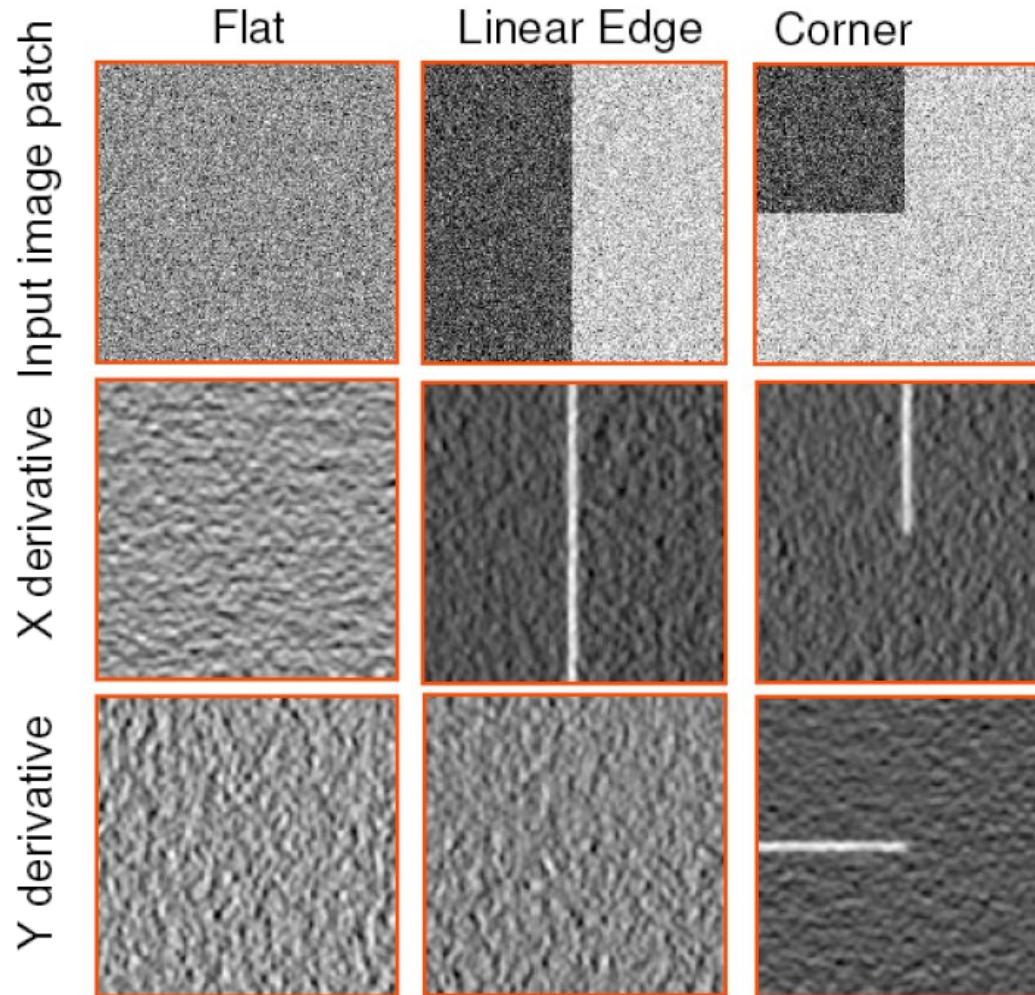
$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 0$$

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Intuitive Way to Understand Harris

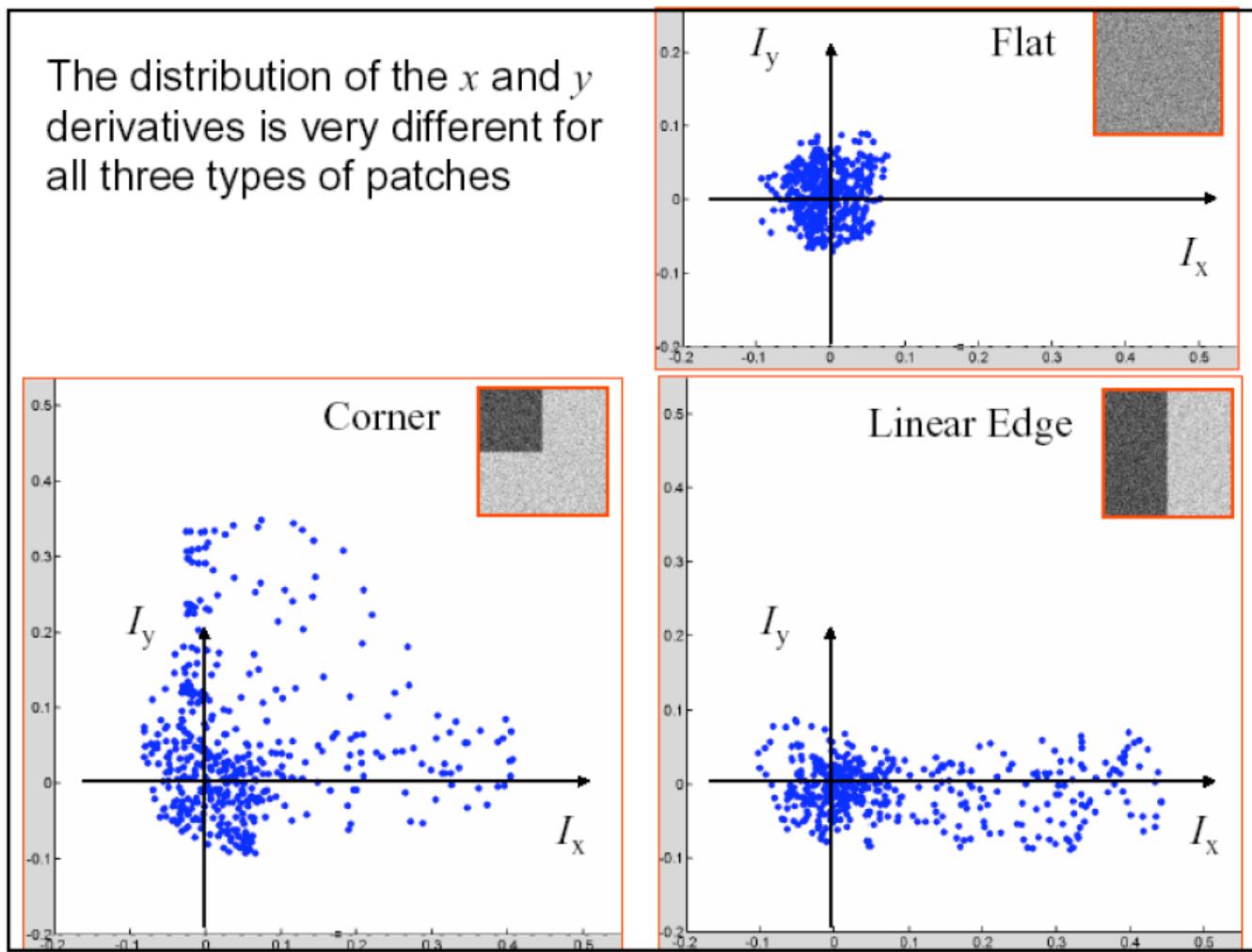
- Treat gradient vectors as a set of (dx, dy) points with a center of mass defined as being at $(0,0)$.
- Fit an ellipse to that set of points via scatter matrix
- Analyze ellipse parameters for varying cases...

Example: Cases and 2D Derivatives



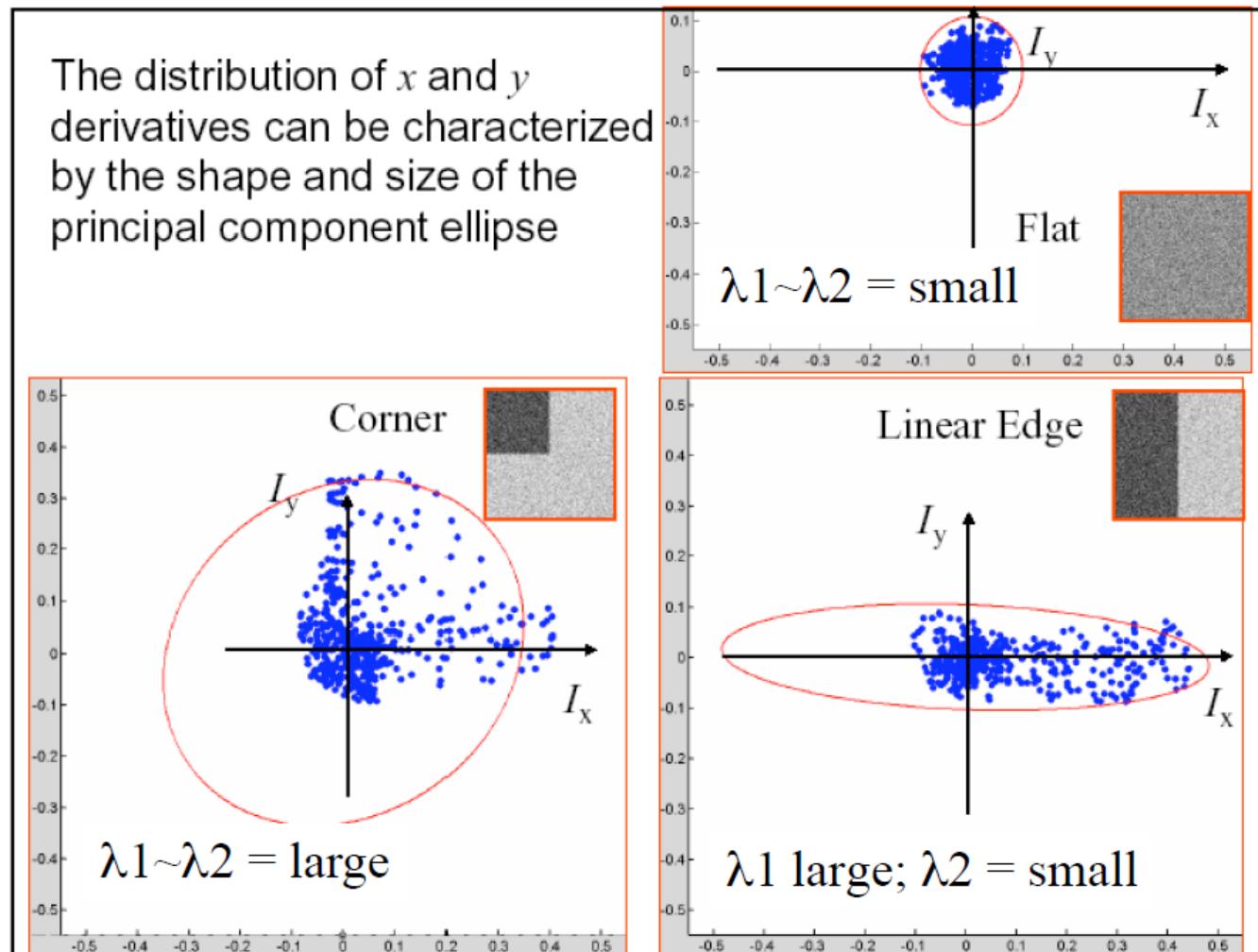
Source: Robert Collins

Plotting Derivatives as 2D Points



Source: Robert Collins

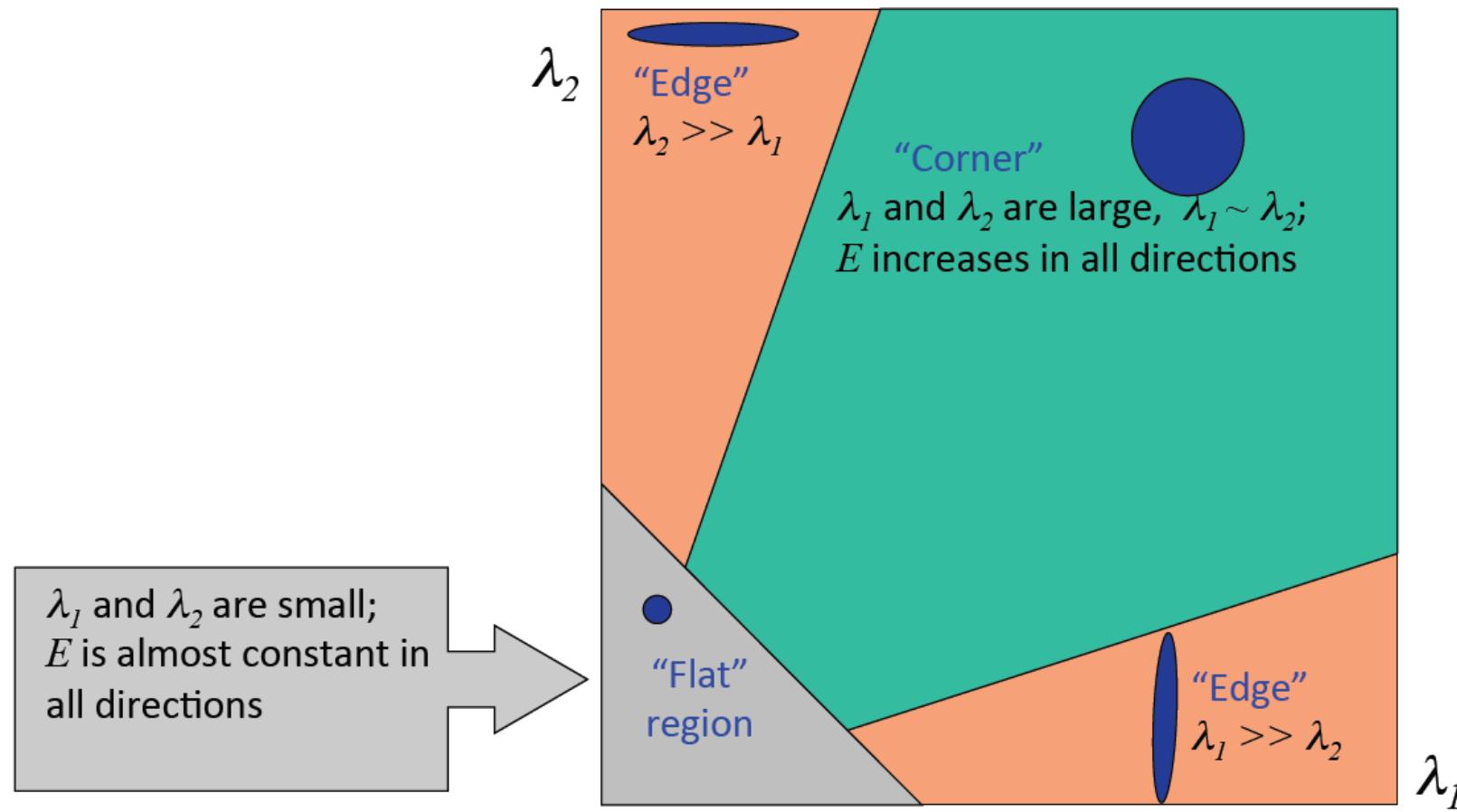
Fitting Ellipse to each Set of Points



Source: Robert Collins

Harris Detector Formulation

- Classification of image points using eigenvalues of

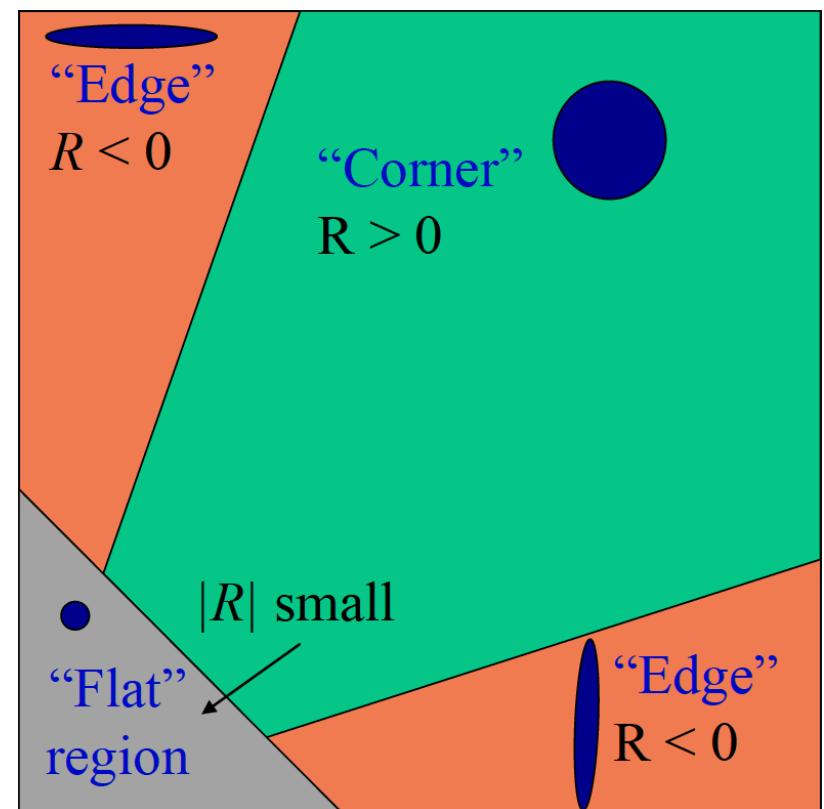


Source: Kristen Grauman

Harris Detector Formulation

$$R = \det(M) - \alpha \operatorname{trace}(M)^2 = \lambda_1 \lambda_2 - \alpha (\lambda_1 + \lambda_2)^2$$

- **Fast approximation**
 - Avoid computing the eigenvalues
 - α : constant (0.04 to 0.06)



Source: Kristen Grauman

Summary: Harris Detector

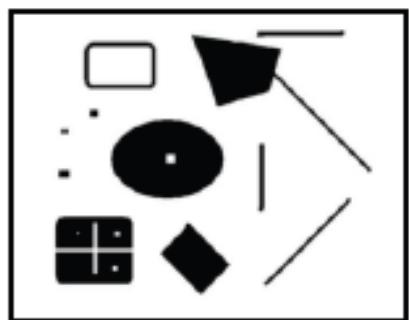
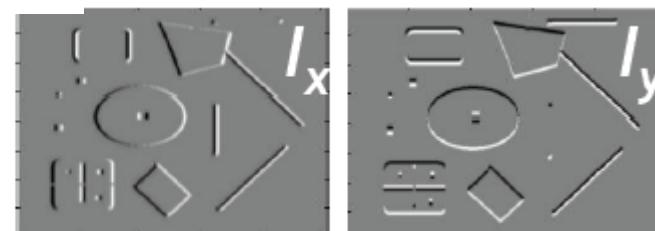
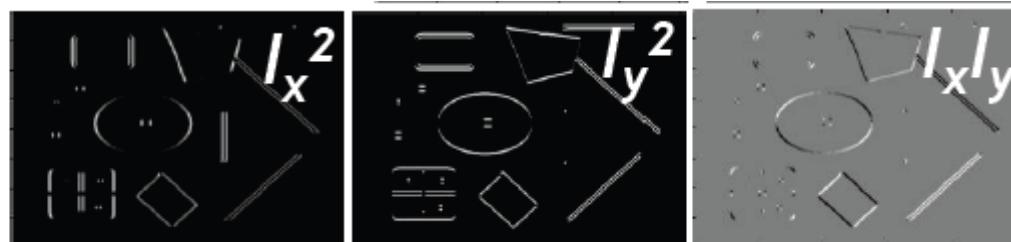


Image derivatives



Square of derivatives



Gaussian filter $g(\sigma)$



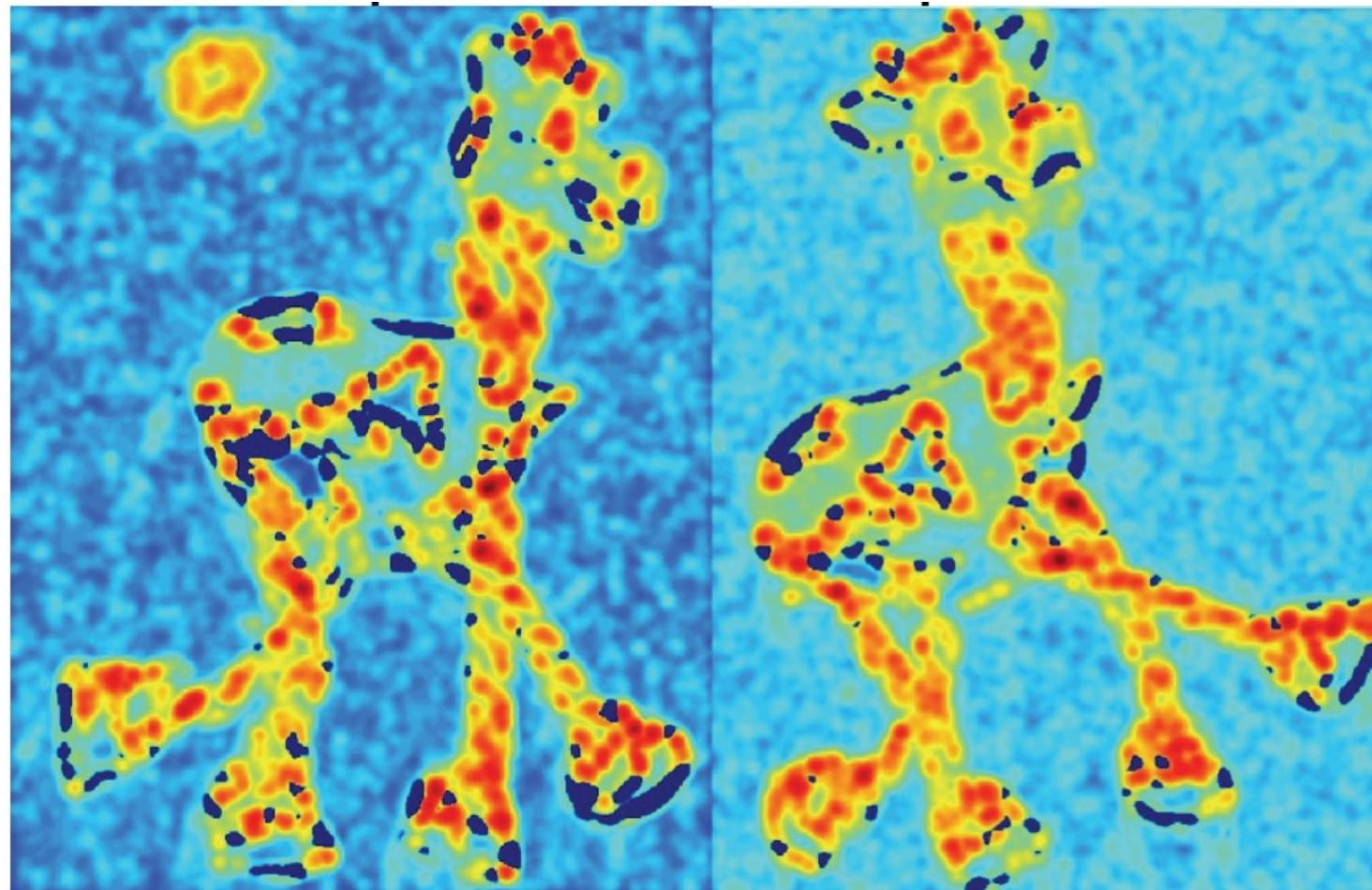
Source: Krystian Mikolajczyk

Harris Detector: Workflow



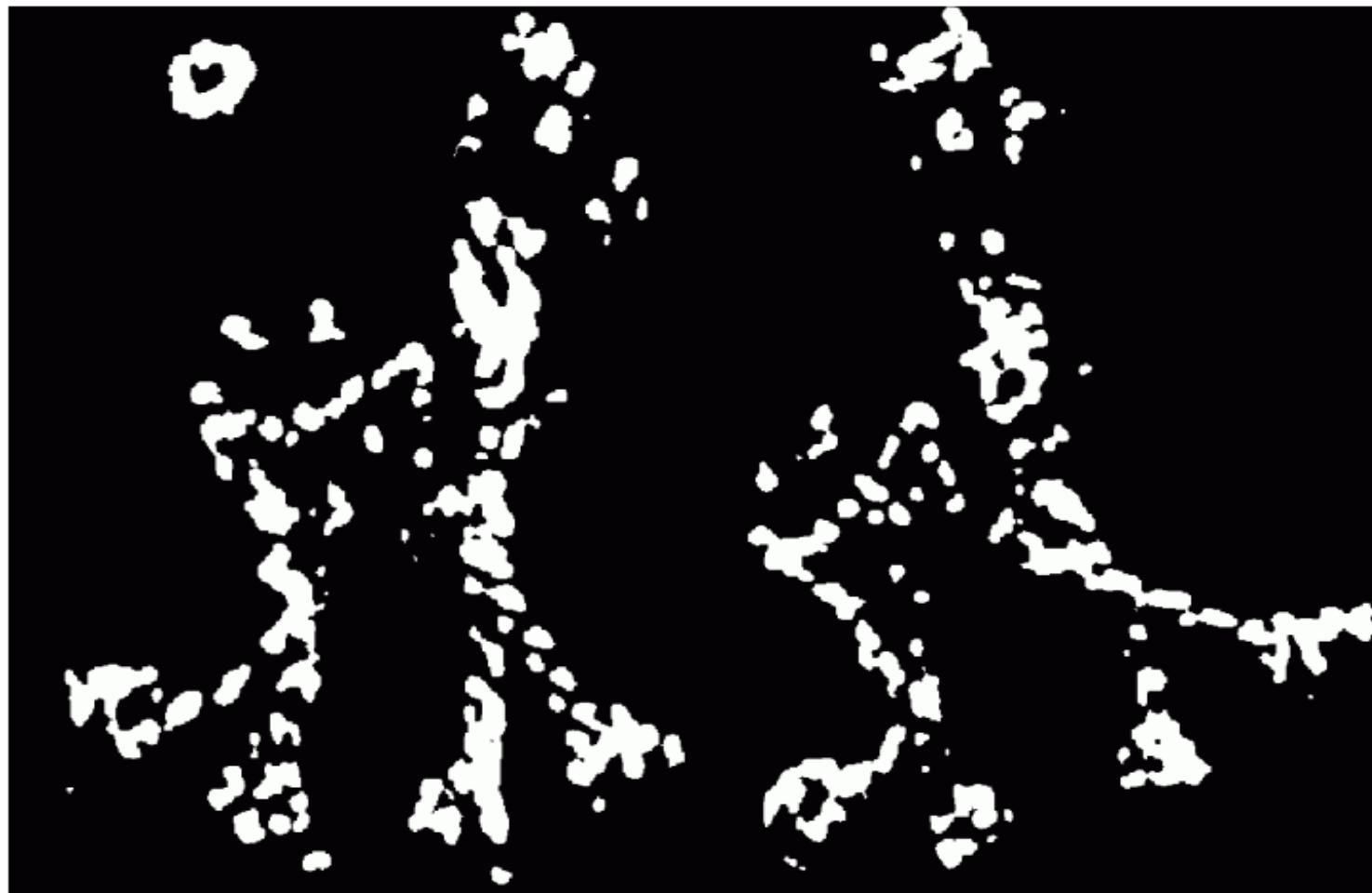
Source: Mubarak Shah

Compute corner responses R



Source: Mubarak Shah

Select points with large corner response ($R > t1$)



Source: Mubarak Shah

Take only the local maxima of R



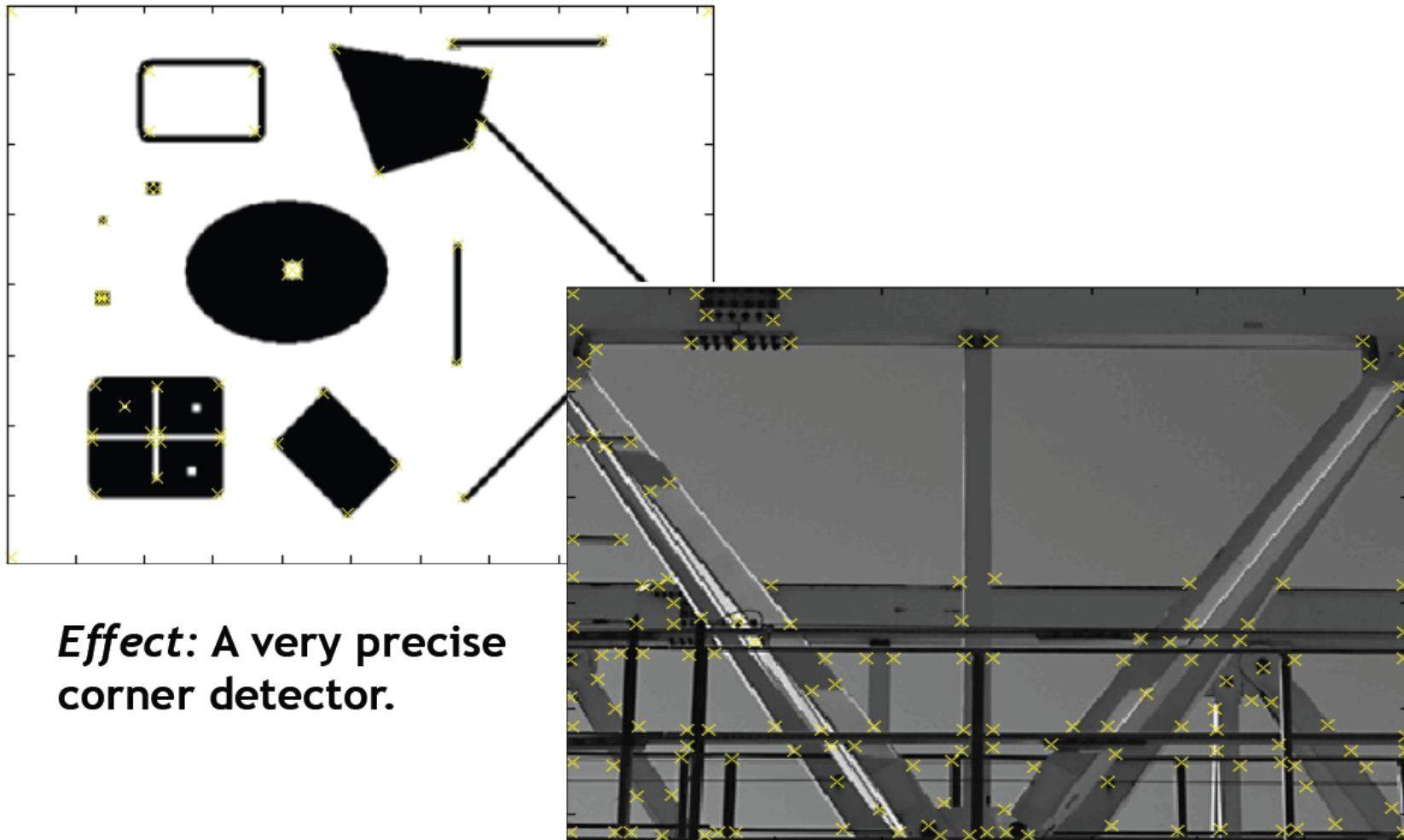
Source: Mubarak Shah

Resulting Harris corner points



Source: Mubarak Shah

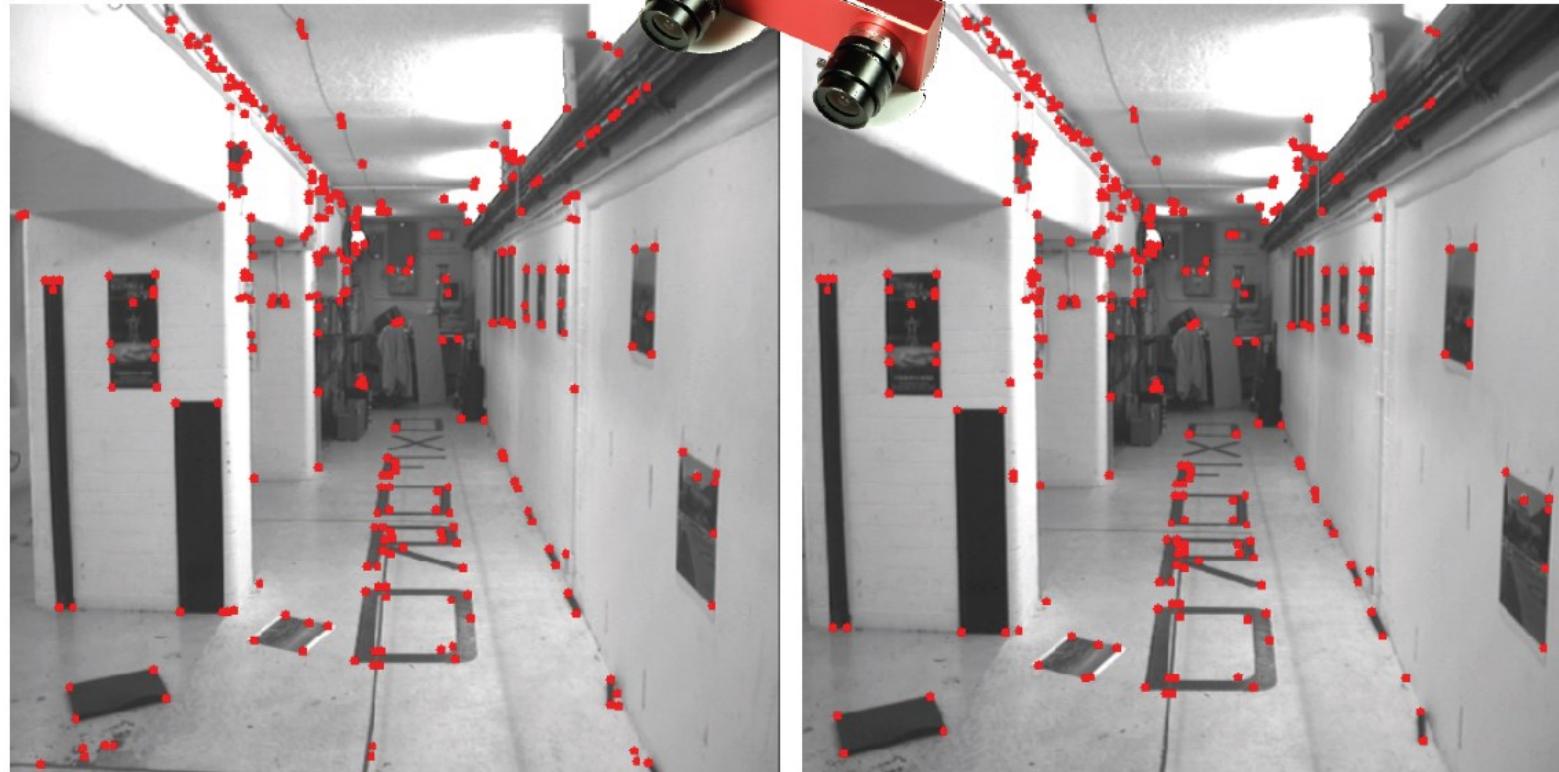
More results ...



Source: Krystian Mikolajczyk

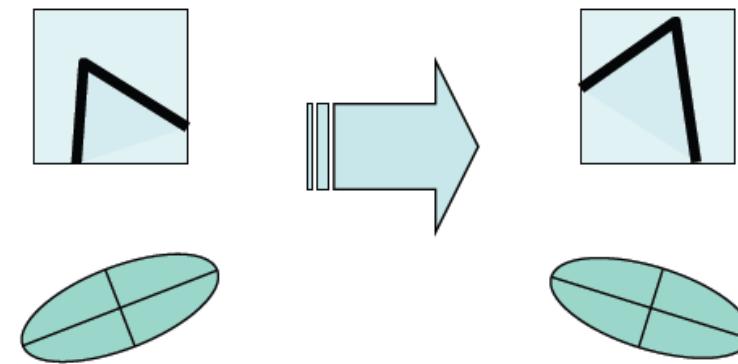
More results ...

- Results are well suited for finding stereo correspondences



Harris Detector: Properties

- Translation invariance?
- Rotation invariance?

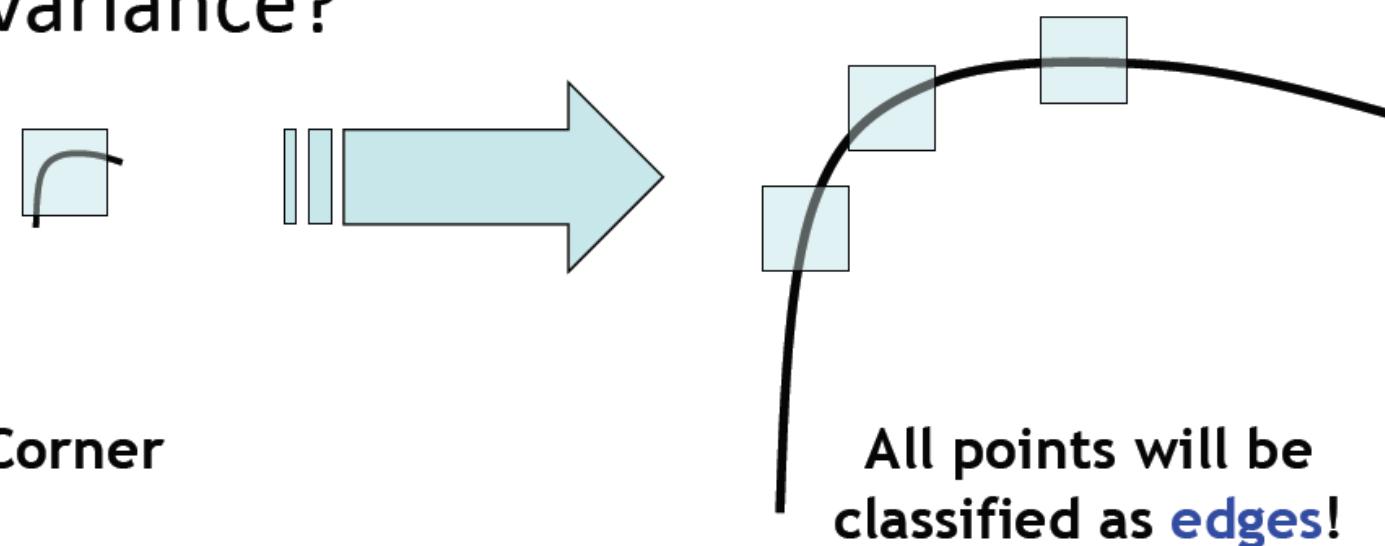


Ellipse rotates but its shape (i.e. eigenvalues) remains the same

Source: James Hays

Harris Detector: Properties

- Scale invariance?



Source: James Hays

Harris Detector: Properties

- Harris Corner detector is
 - Translation invariant?
 - YES ☺
 - Rotation invariant?
 - YES ☺
 - Scale invariant
 - NO ☹

For Next Time...

- Introduction
- Harris Corner Detector
- Scale Invariant Feature Transform (SIFT)

Resources and References

□ Readings

- C. Harris and M. Stephens. "A Combined Corner and Edge Detector." Proceedings of the 4th Alvey Vision Conference, 1988

□ Slides sources

- Fei-Fei Li
- David Lowe
- Mubarak Shah
- Robert Collins
- Chang Shu
- Sanja Fidler

Image Processing & Pattern Recognition

Feature Detectors and Descriptors

Part 2: Feature Description and Matching

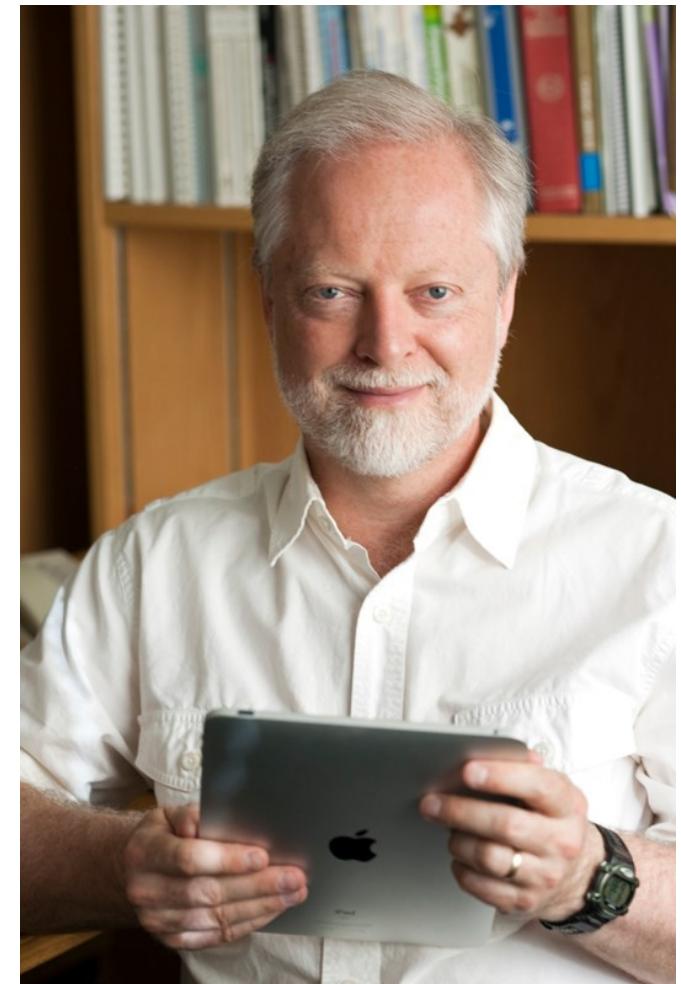
Dr. Zia Ud Din

Outline

- Introduction
- Harris Corner Detector
- Scale Invariant Feature Transform (SIFT)

SIFT

- *Distinctive Image Features from Scale-Invariant Keypoints, David G. Lowe, IJCV, 2004*
 - From University of British Columbia
 - One of the most influential papers in Image Processing and Computer Vision
 - 72430 citations on Google scholar as of Oct 29, 2023
 - It was patented so was not available in OpenCV, but the patent expired in 2020



Scale Invariant Detection

- Problem: How can we **independently select** interest points in each image, such that the detections are **repeatable** across different scales?

image 1



image 2

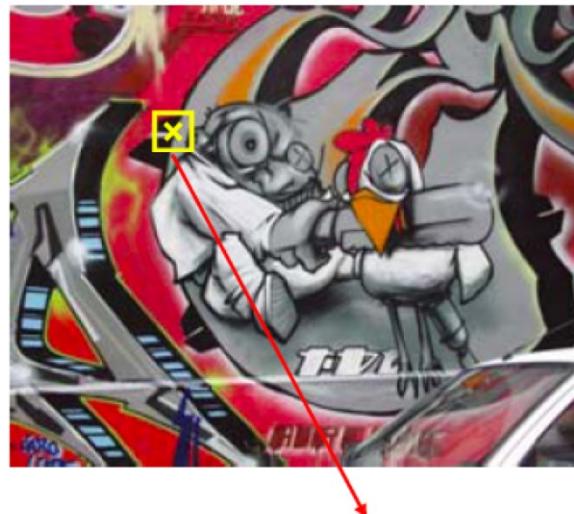


Source: K. Grauman, B. Leibe

Scale Invariant Detection

□ Solution:

- Extract features that are stable in both location and scale.
- Find scale that gives local maxima of a function f in both position and scale.



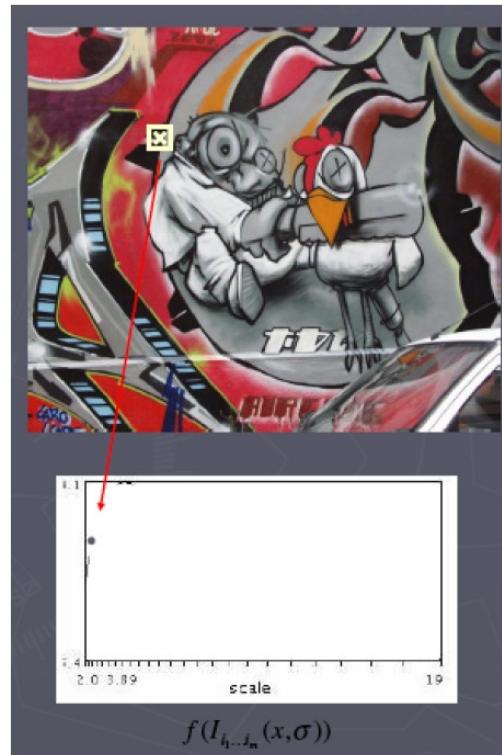
$$f(I_{i_1 \dots i_m}(x, \sigma)) = f(I_{i_1 \dots i_m}(x', \sigma'))$$



Source: K. Grauman, B. Leibe

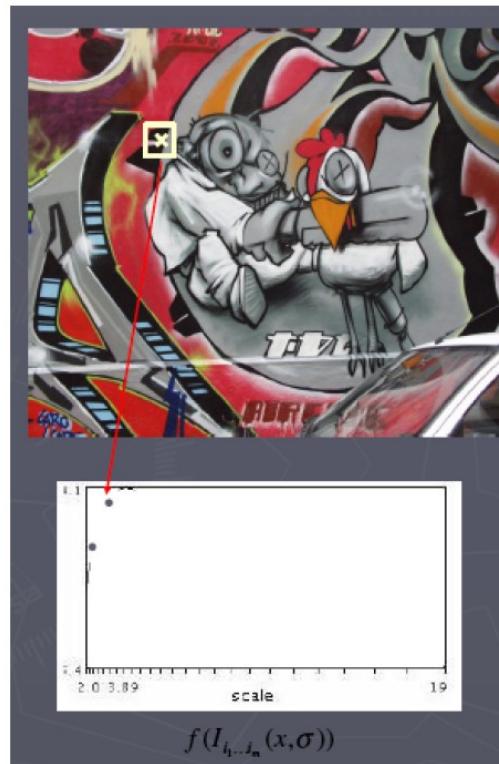
Automatic Scale Selection

- Function responses for increasing scale (**scale signature**).



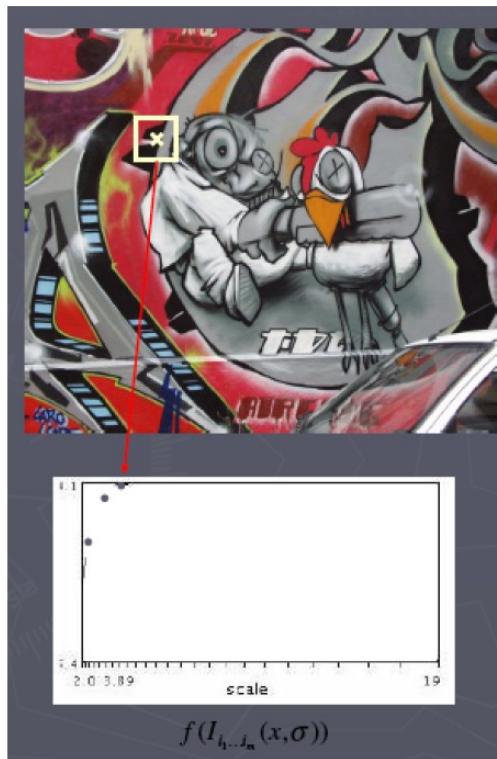
Source: K. Grauman, B. Leibe

Automatic Scale Selection



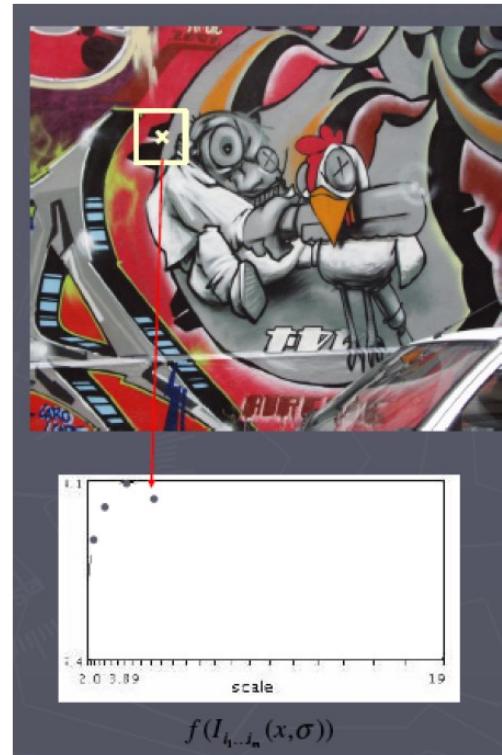
Source: K. Grauman, B. Leibe

Automatic Scale Selection



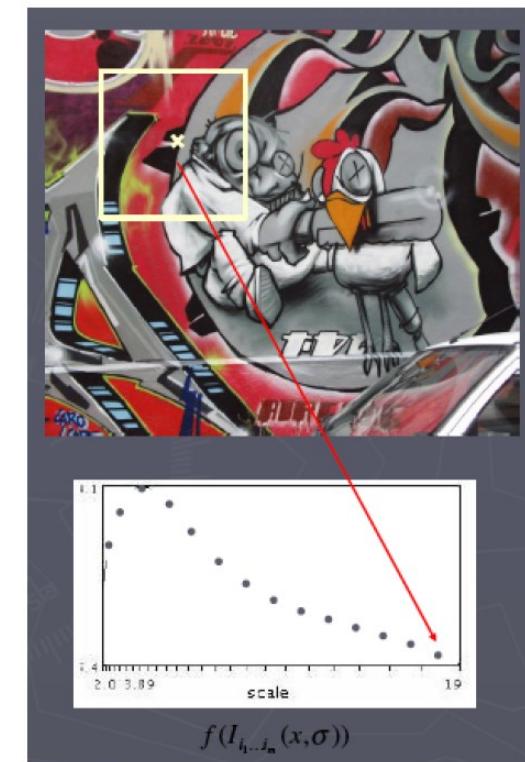
Source: K. Grauman, B. Leibe

Automatic Scale Selection



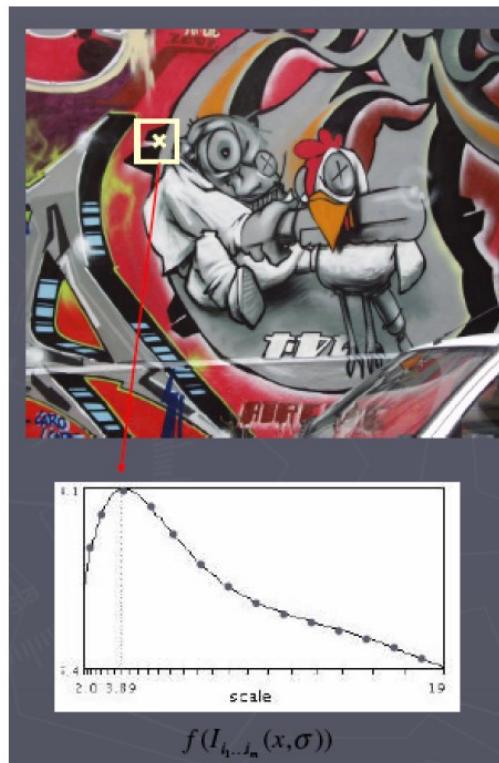
Source: K. Grauman, B. Leibe

Automatic Scale Selection



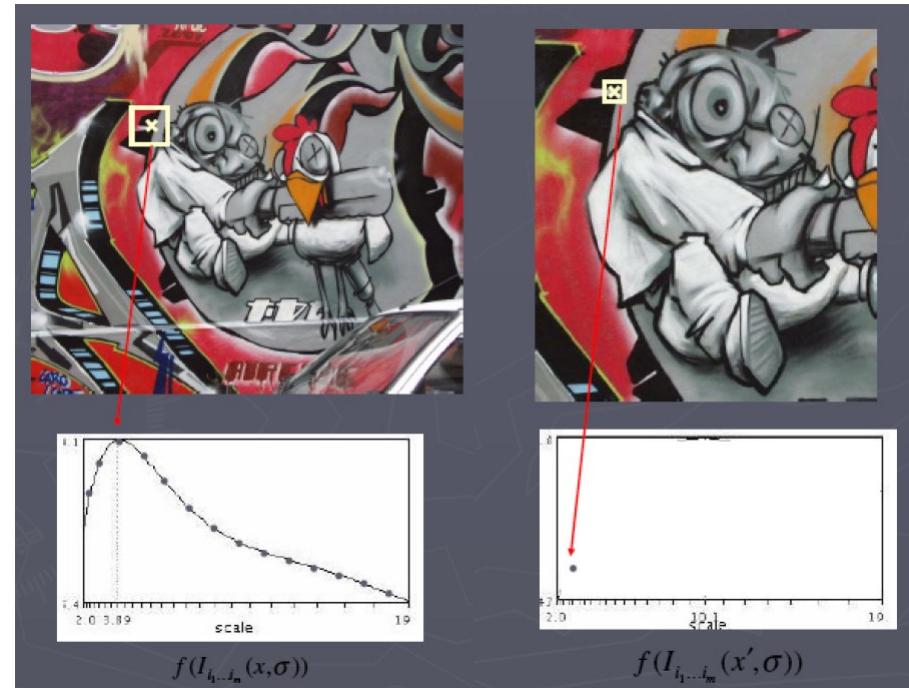
Source: K. Grauman, B. Leibe

Automatic Scale Selection



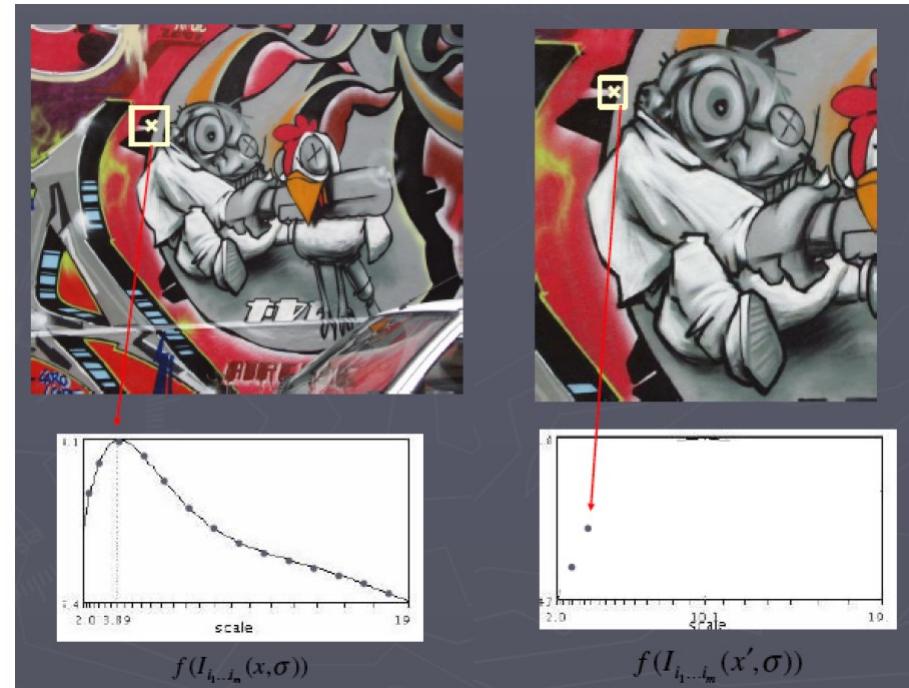
Source: K. Grauman, B. Leibe

Automatic Scale Selection



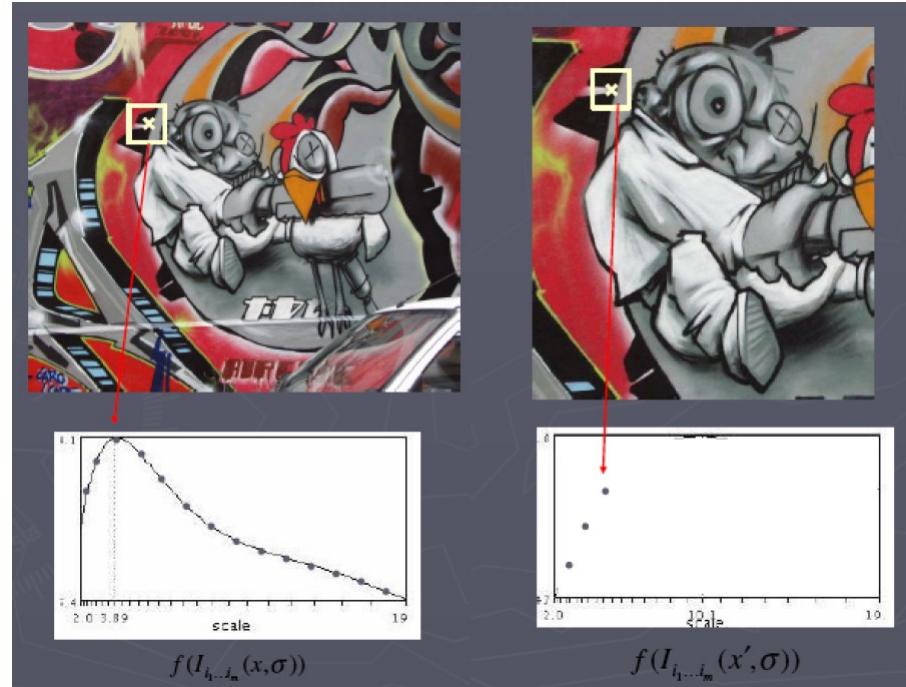
Source: K. Grauman, B. Leibe

Automatic Scale Selection



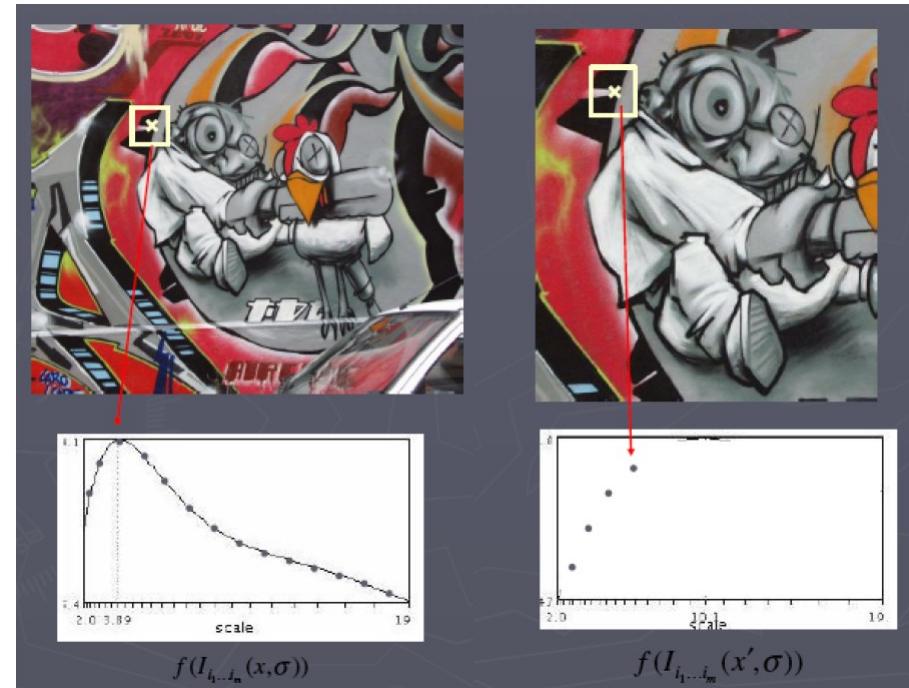
Source: K. Grauman, B. Leibe

Automatic Scale Selection



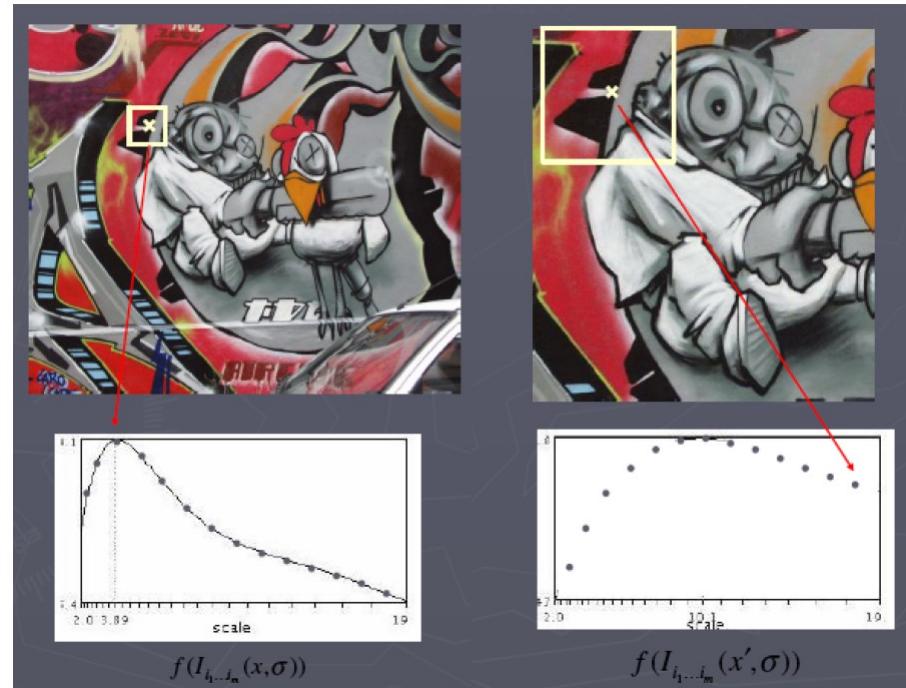
Source: K. Grauman, B. Leibe

Automatic Scale Selection



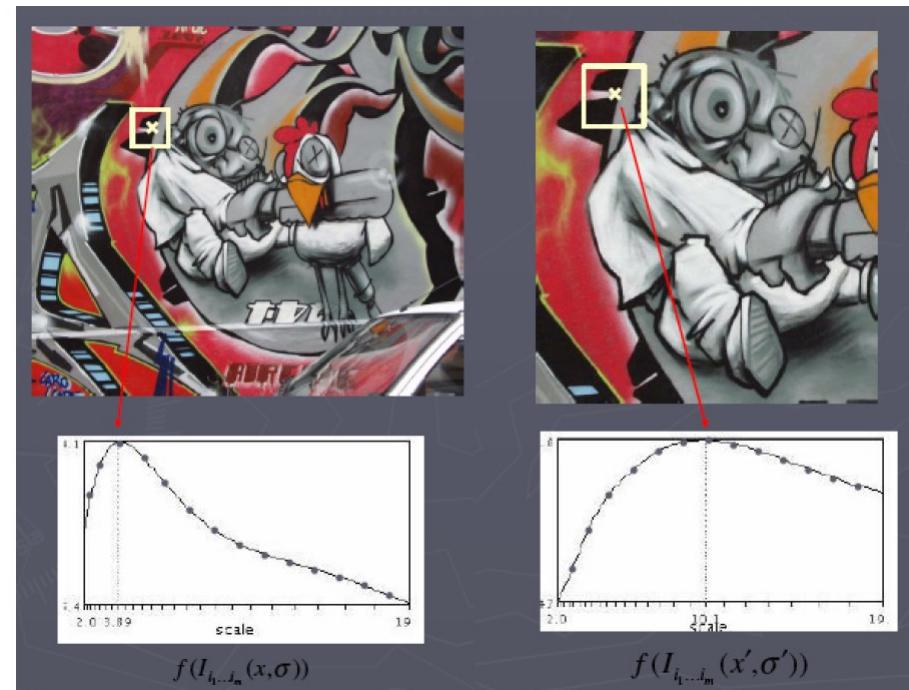
Source: K. Grauman, B. Leibe

Automatic Scale Selection



Source: K. Grauman, B. Leibe

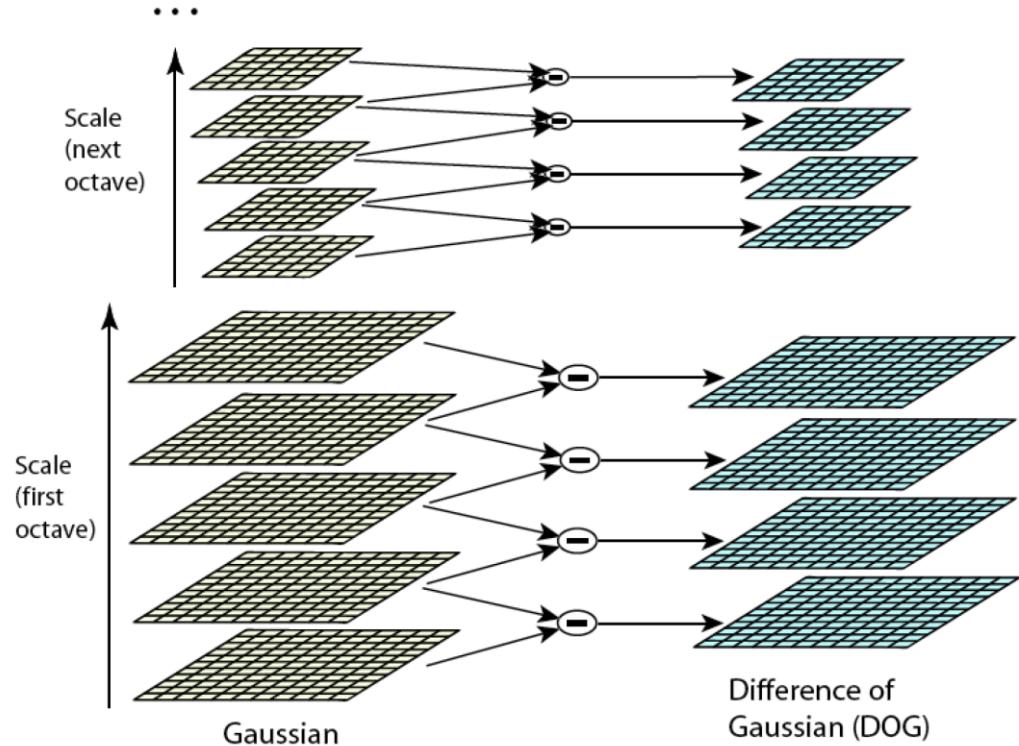
Automatic Scale Selection



Source: K. Grauman, B. Leibe

What Can the Signature Function Be?

- Lowe (2004) proposed computing a set of sub-octave **Difference of Gaussian** filters looking for 3D (space+scale) maxima in the resulting structure.



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

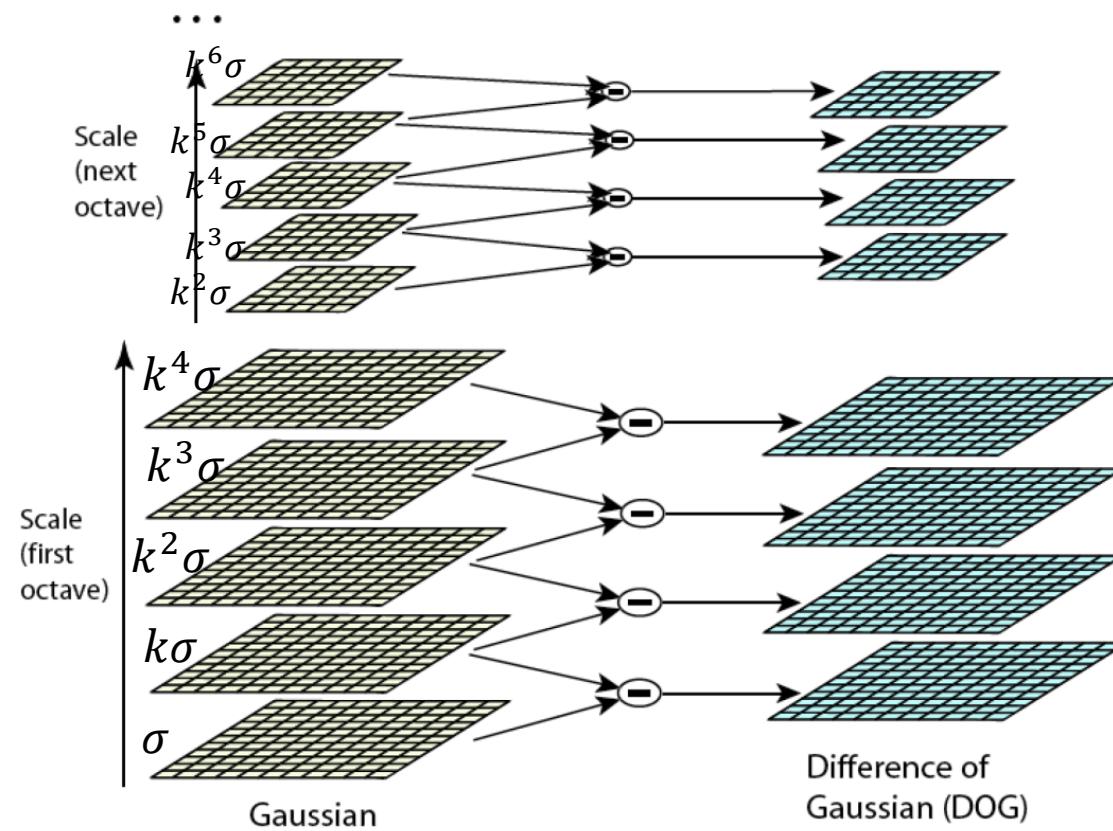
SIFT: Steps

1. Scale-space extrema detection
2. Keypoint localization
3. Orientation assignment
4. Keypoint descriptor

Step 1: Scale-space Extrema Detection

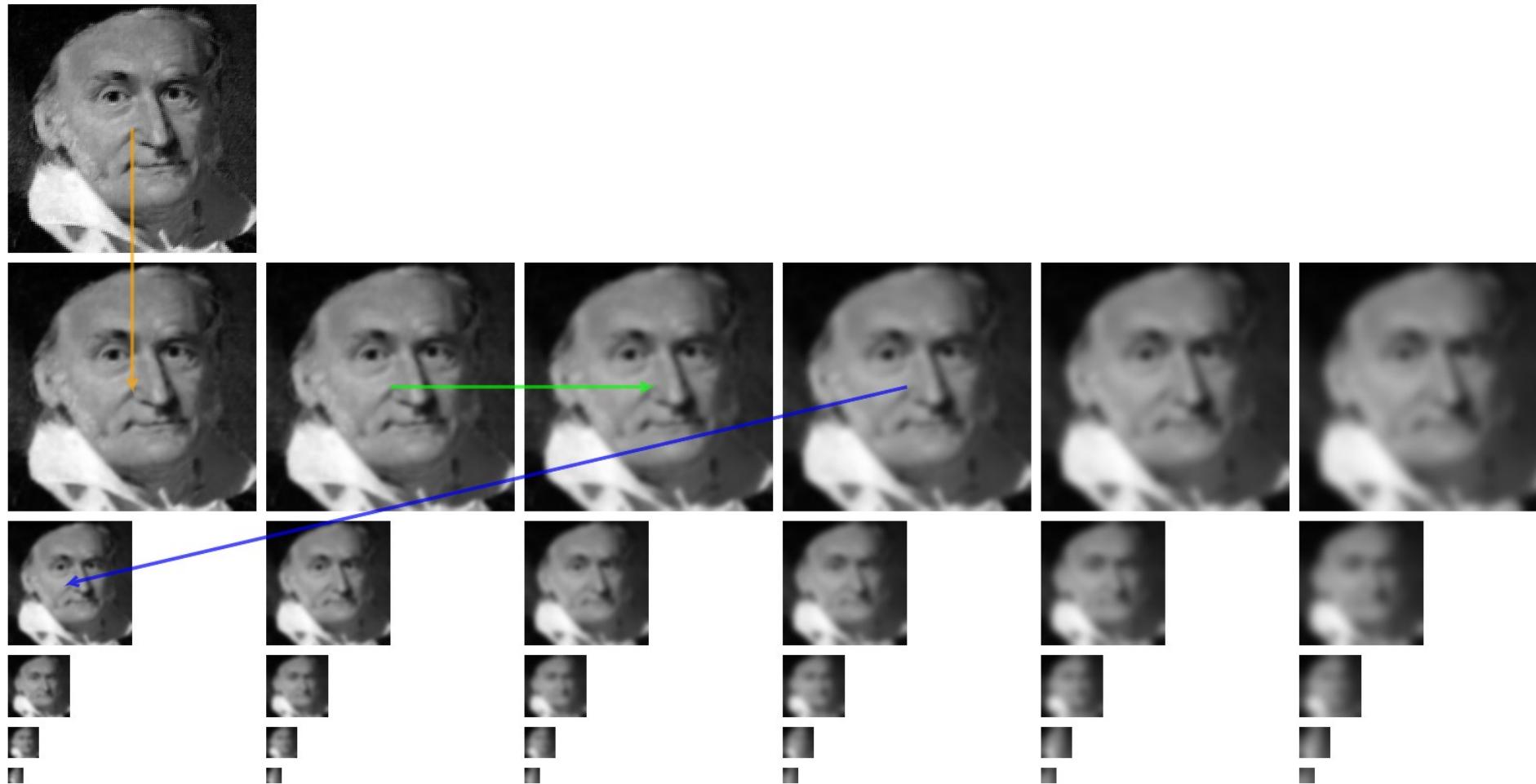
- For each **octave** of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images.
- Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images.
- After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

Scale-space Extrema Detection



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

Example



Source: <http://weitz.de/sift/index.html>

Example

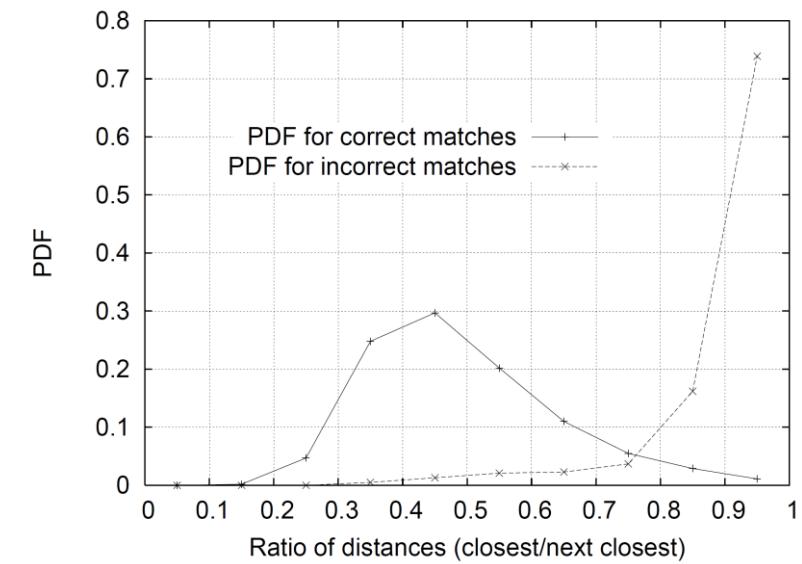
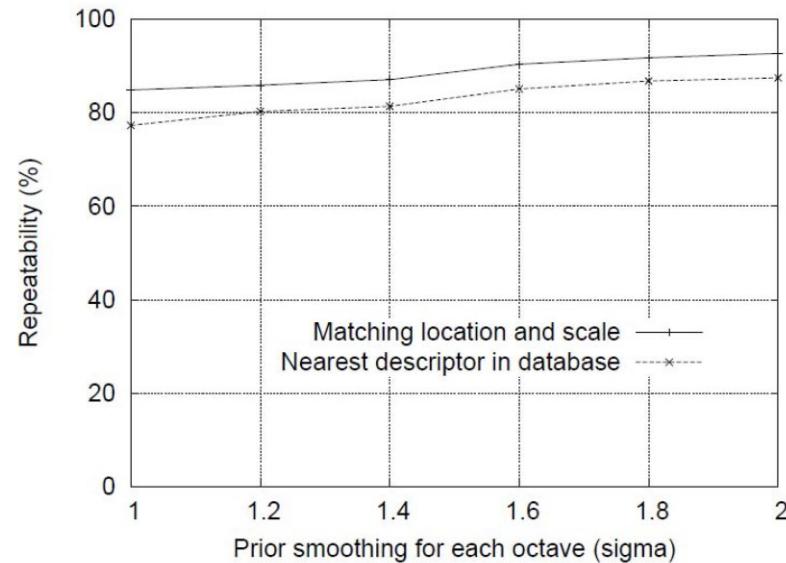
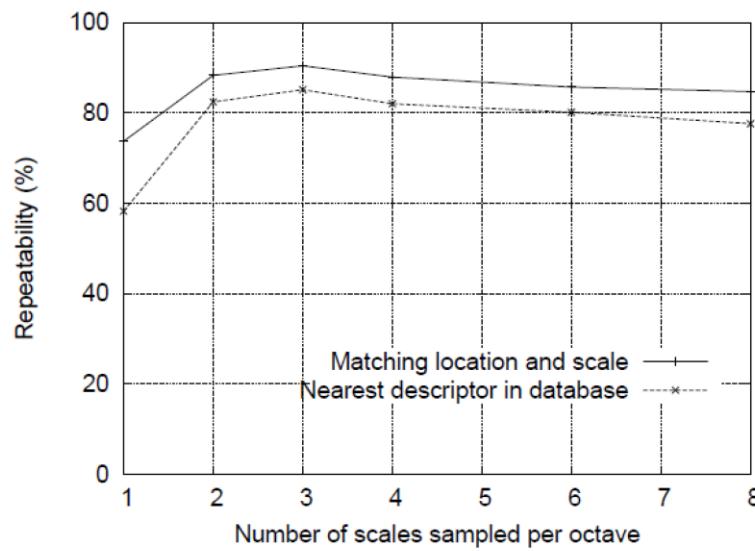


Source: <http://weitz.de/sift/index.html>

Parameter Selection

- 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, contrast, and image noise.
- For matching, a database of 40,000 keypoints generated from 32 real images was used

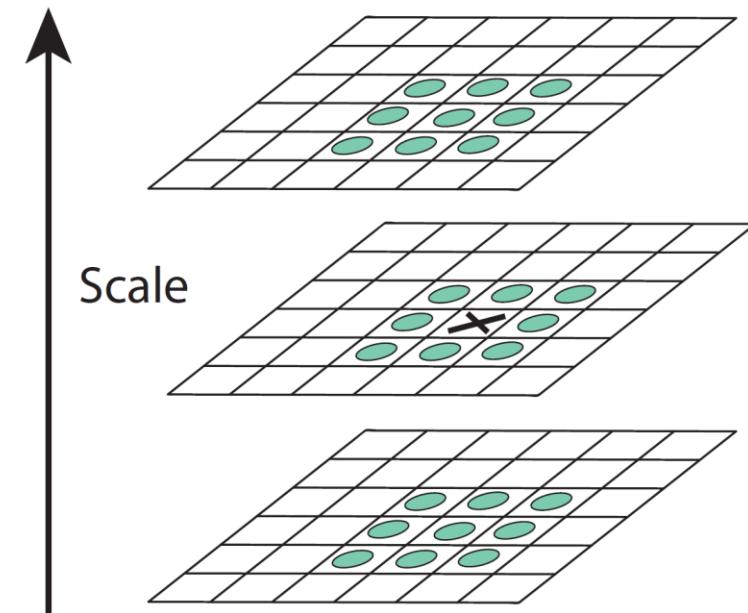
Parameter Selection



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

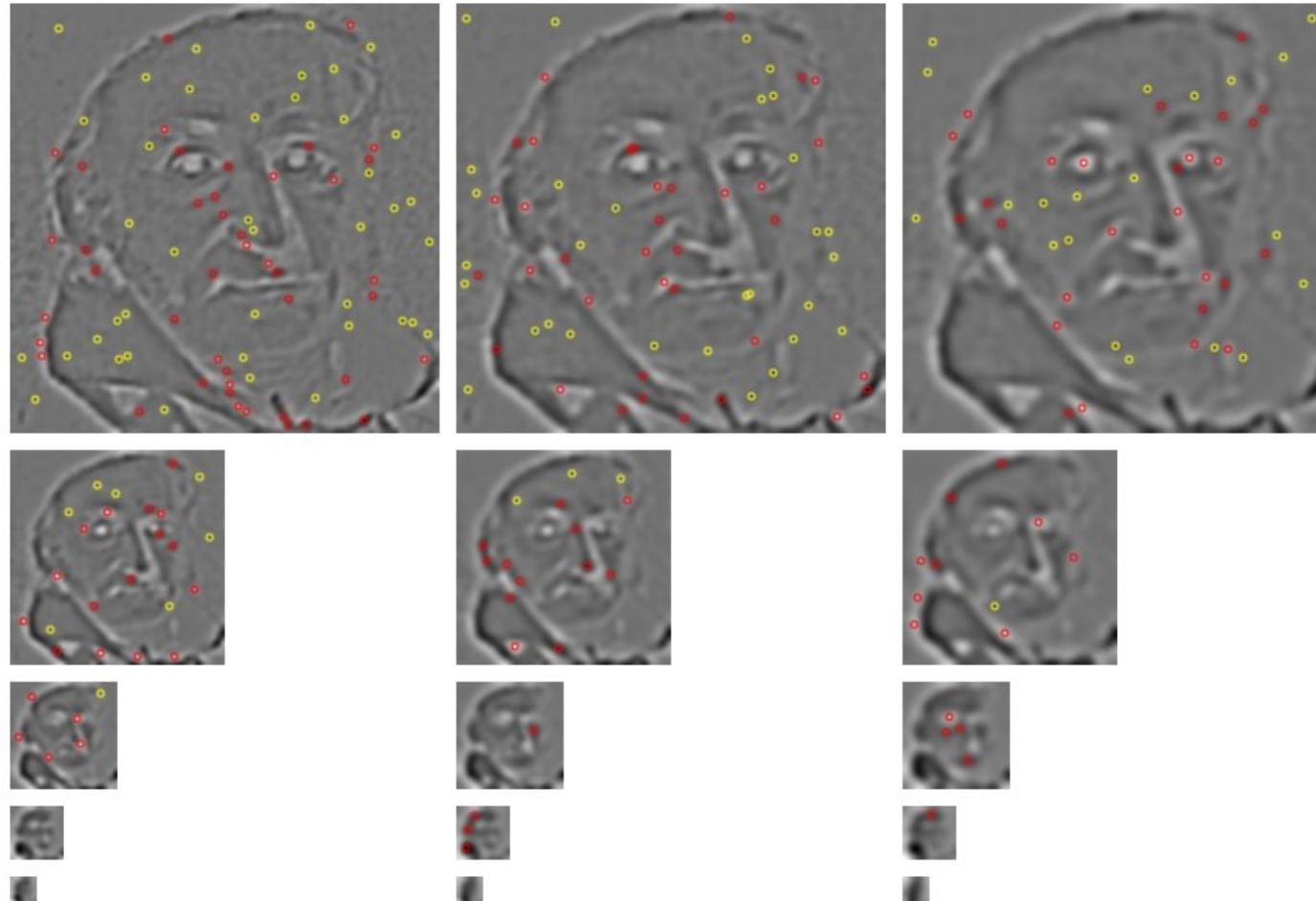
Scale Space Peak Detection

- Compare a pixel (**X**) with 26 pixels in current and adjacent scales (Green Circles)
- Select a pixel (**X**) if larger/smaller than all 26 pixels
 - Cost is low as most sample points will be eliminated following the first few checks.
- The selected pixels are the potential interest points



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

Scale Space Peak Detection



Source: <http://weitz.de/sift/index.html>

Step 2: Keypoint Localization

- Remove outliers (unstable keypoints)
 1. Remove low contrast candidates
 - If the intensity of DoG image at extrema is less than a threshold value (0.03 as per the paper), it is rejected.

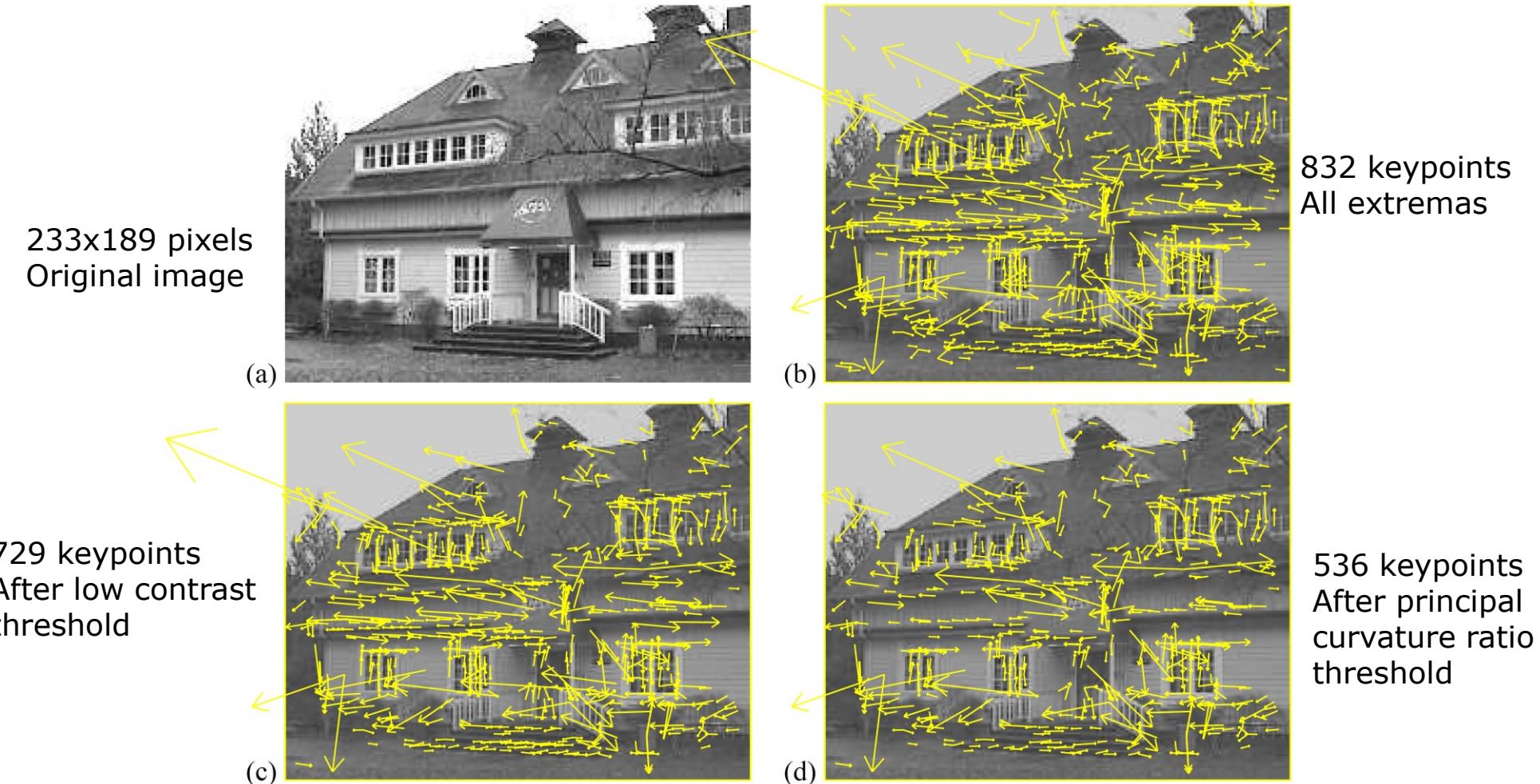
Step 2: Keypoint Localization...

2. Remove poorly localized candidates along an edge
 - DoG has strong response along edge
 - Assume DoG as an ellipse
 - Along the edge one of the PC is very low, across the edge is high
 - Analogous to Harris corner detector, compute Hessian of DoG

$$H = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

- Let r be the ratio of the larger eigen value and the smaller eigen value
$$r = \lambda_1 / \lambda_2$$
- Eliminate key points if r is less than a threshold value (10 as per the paper)

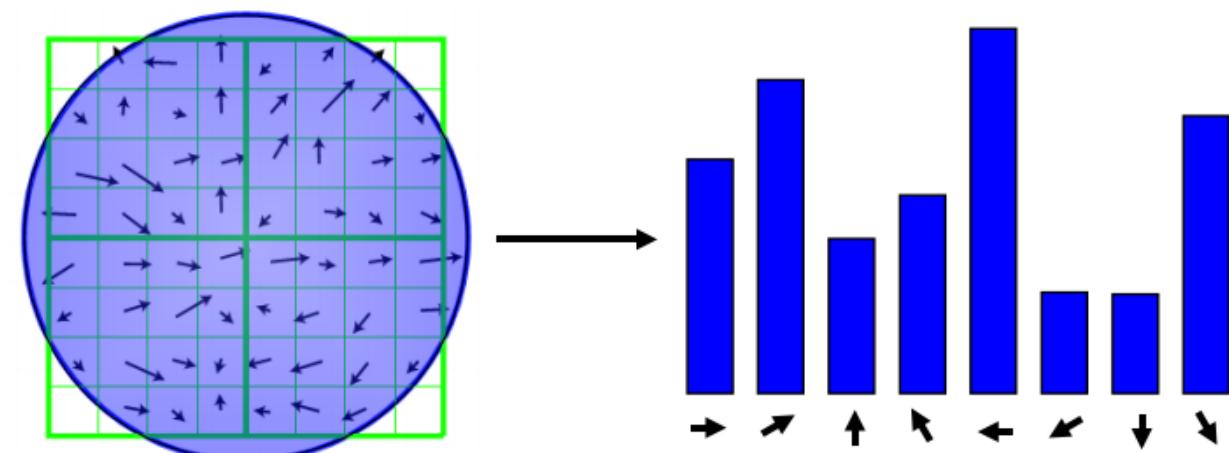
Keypoint Localization



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

Step 3: Orientation Assignment

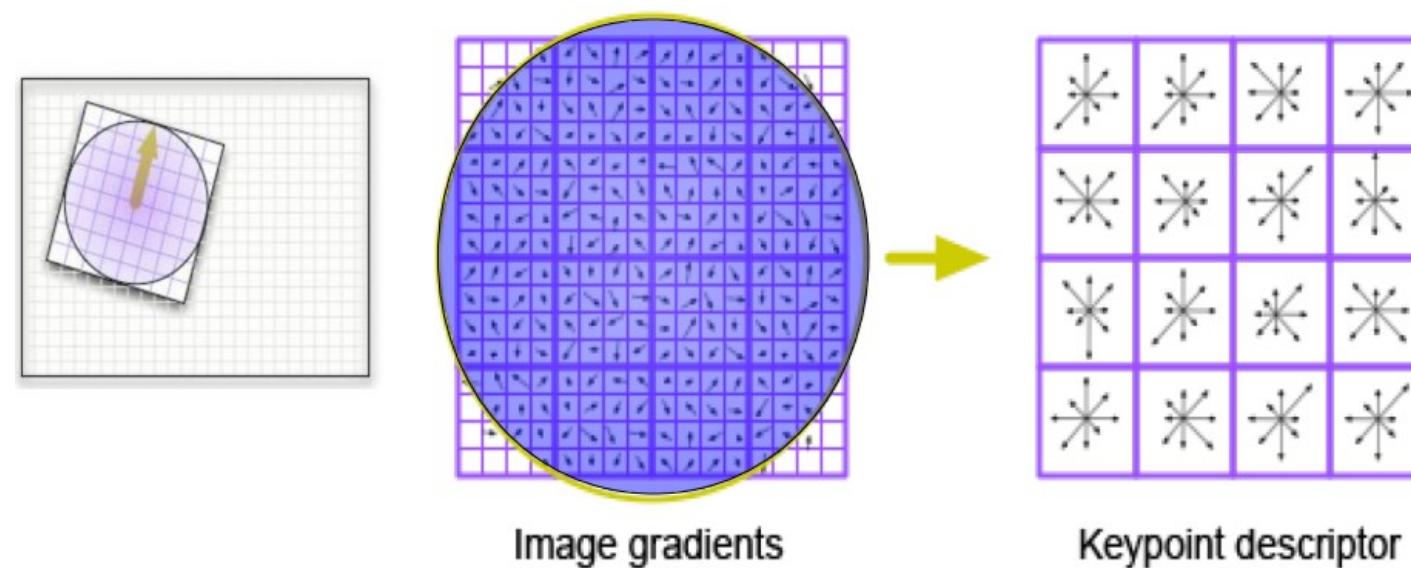
- Orientation is assigned to each keypoint to achieve rotation invariance
- Compute gradient magnitude and direction at the scale of key point (x,y)
 - in neighborhood of each keypoint
- Create gradient histogram (36 bins)
 - Weighted by magnitude and Gaussian window
- Select the peak as direction of the key point



Only 8 bins shown in image (out of 36)

Step 4: Keypoint Descriptor

- Compute **relative** orientation and magnitude in a 16×16 neighborhood at key point
- Form weighted histogram (8 bin) for 4×4 regions
 - Gaussian weighting around center
- $4 \times 4 \times 8 = 128$ dimensional feature vector



Source: Jonas Hurrelmann

Keypoint Descriptor - Lighting changes

- Store numbers in a vector
- Normalize to unit vector
 - Illumination invariance (affine changes)
- For non-linear intensity transforms
 - Bound unit vector values to maximum 0.2 (remove larger gradients)
- Renormalize to unit vector

Keypoint Matching

- Match the key points against a database of that obtained from training images.
- The distance metric used is Euclidean distance
- 3 options tried
 1. Threshold on distance
 - bad performance
 2. Nearest Neighbor
 - better
 3. Ratio Test
 - best performance

Keypoint Matching...

- Find two nearest neighbors i.e. key points with minimum Euclidean distances.
- To add robustness to matching, consider ratio of distances (*best match/2nd best match*)
 - If low, first match looks good.
 - If high, could be ambiguous match (reject it).
 - Threshold used in the paper=0.8

SIFT

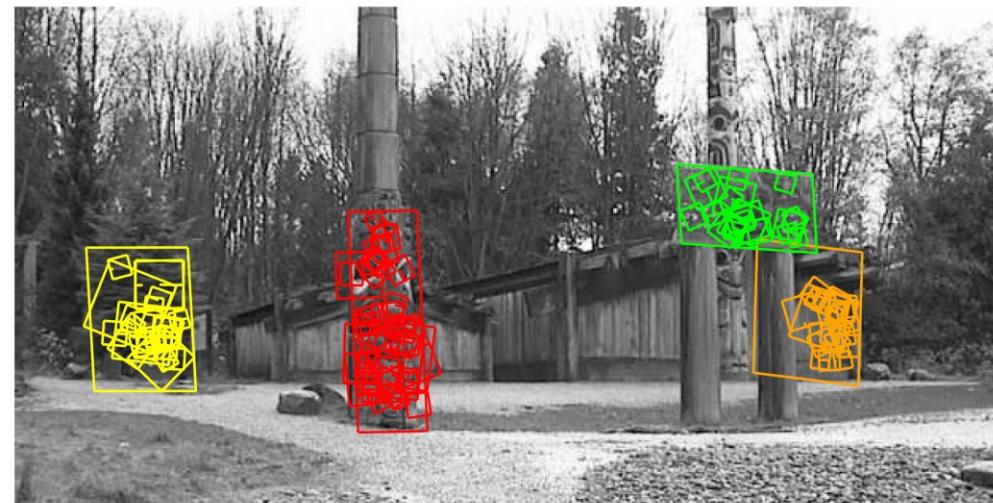
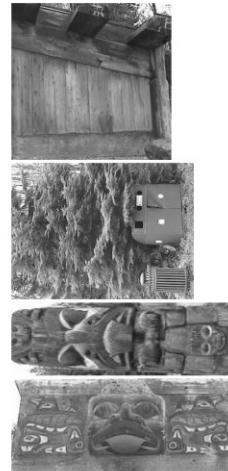
- Invariant to:
 - Scale
 - Rotation
- Partially invariant to:
 - Illumination changes (sometimes even day vs. night)
 - Camera viewpoint (up to about 60 degrees of out-of-plane rotation)
 - Occlusion, clutter
- Also important:
 - Fast and efficient - can run in real time
 - Lots of code available

Some Results



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

Some Results



Source: *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004

Summary

- In this lecture, we learned
 - Introduction
 - Harris Corner Detector
 - Scale Invariant Feature Transform (SIFT)

Resources and References

- Readings
 - *Distinctive Image Features from Scale-Invariant Keypoints*, David G. Lowe, IJCV, 2004.
- Slides sources
 - Fei-Fei Li
 - David Lowe
 - Mubarak Shah
 - Robert Collins
 - Chang Shu
 - Sanja Fidler
 - <http://weitz.de/sift/index.html>
 - <https://www.inf.fu-berlin.de/lehre/SS09/CV/uebungen/uebung09/SIFT.pdf>

Image Processing & Pattern Recognition

Image Segmentation and Clustering Part 1: Segmentation

Dr. Zia Ud Din

Outline

□ Segmentation

- Thresholding
- Edge Detection
- Region-based Segmentation

□ Clustering

- K-means
- Mean Shift

Contents

□ Segmentation

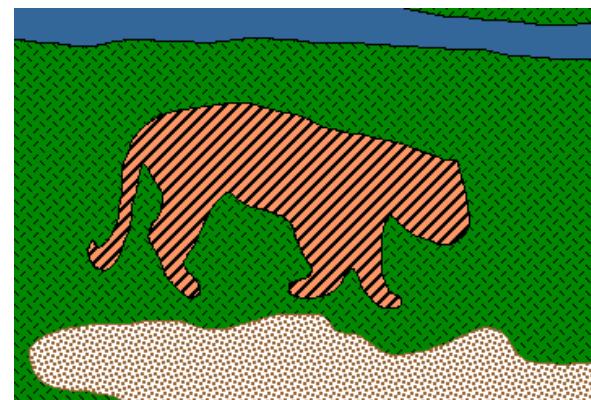
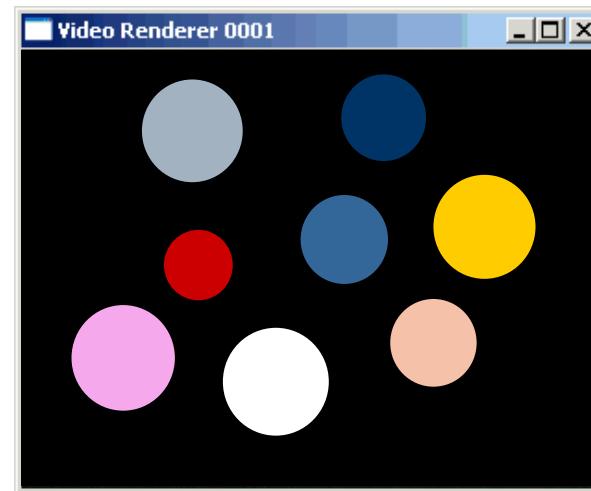
- Thresholding
- Edge Detection
- Region-based Segmentation

□ Clustering

- K-means
- Mean Shift

Segmentation

- Segmentation subdivides an image to regions or objects



Segmentation

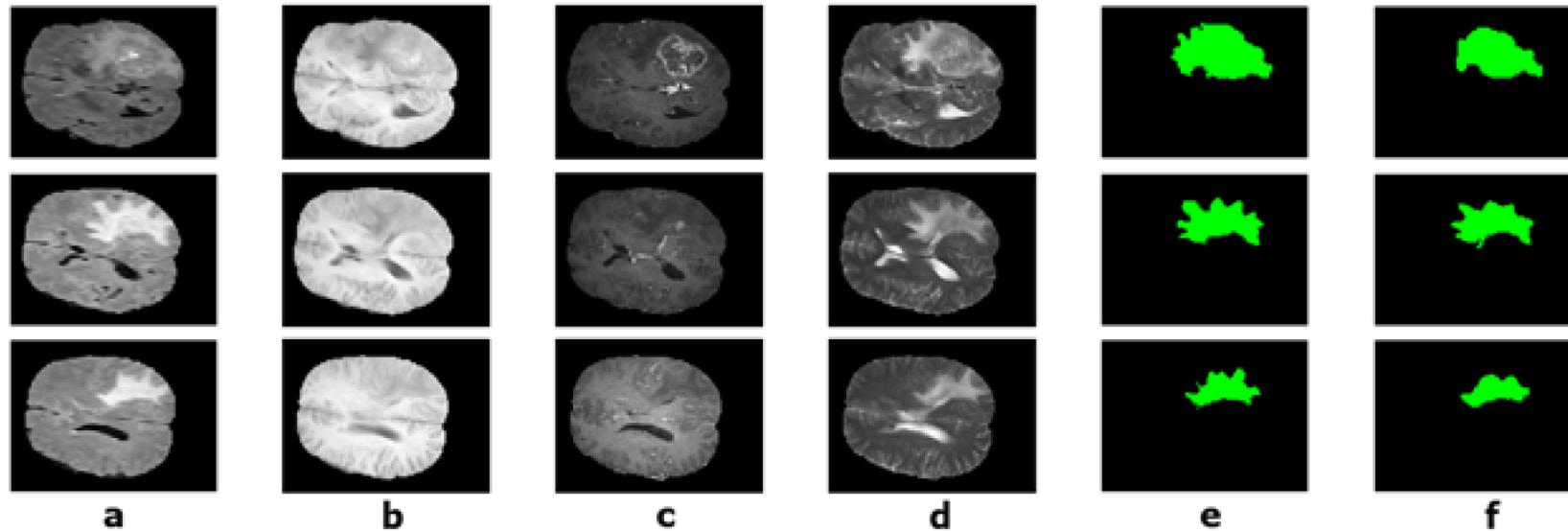
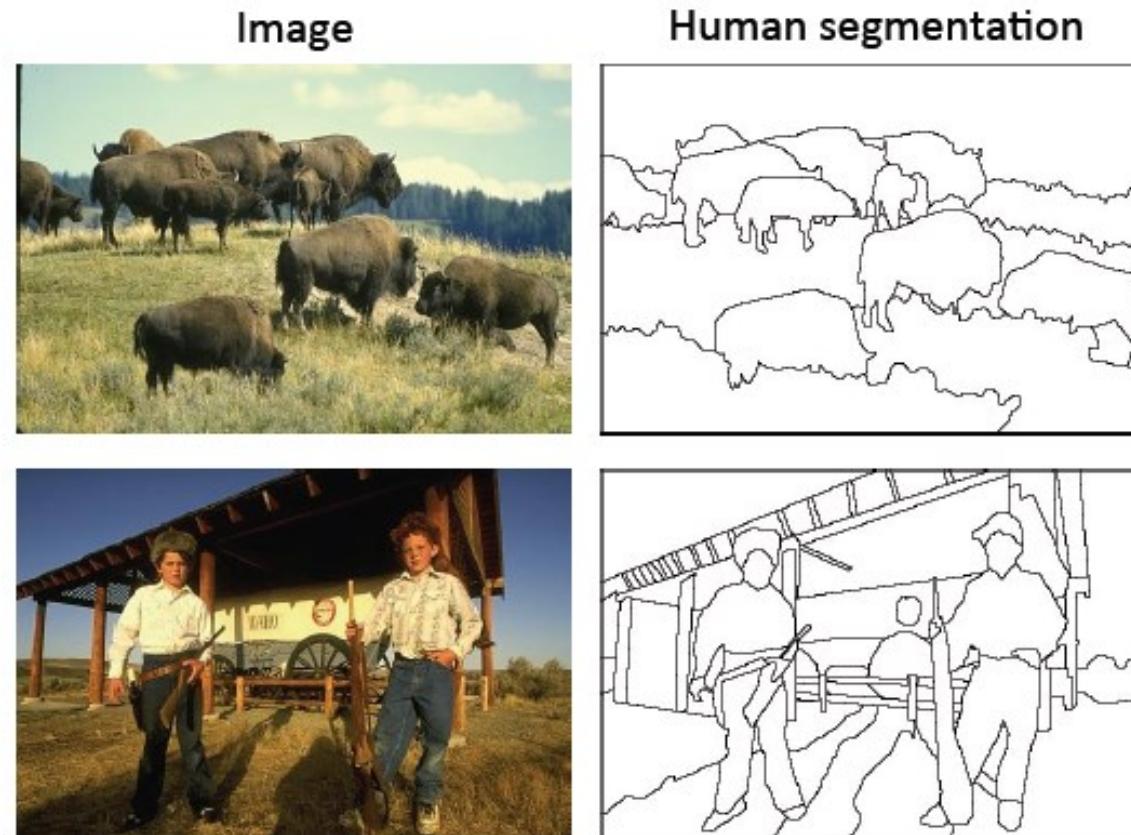
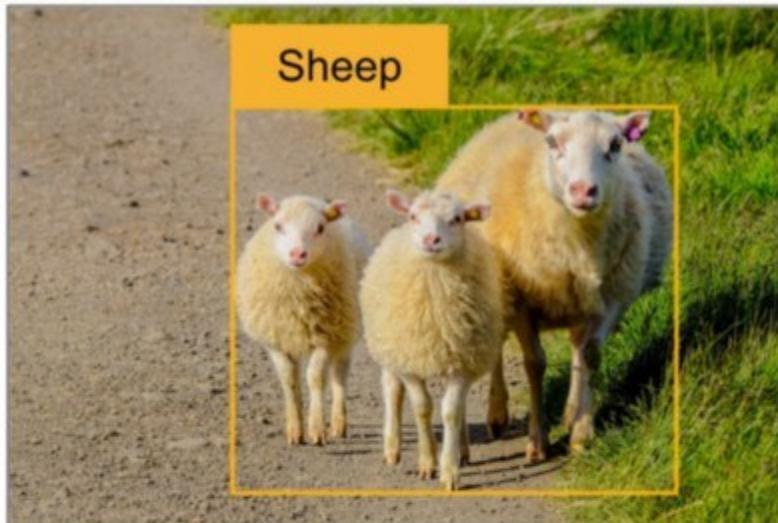


Fig. 2. Three examples (row) of Complete Tumor segmentation. (a) Flair, (b) T1, (c) T1-contrast, (d) T2, (e) Manual segmentation, (f) Automatic segmentation.

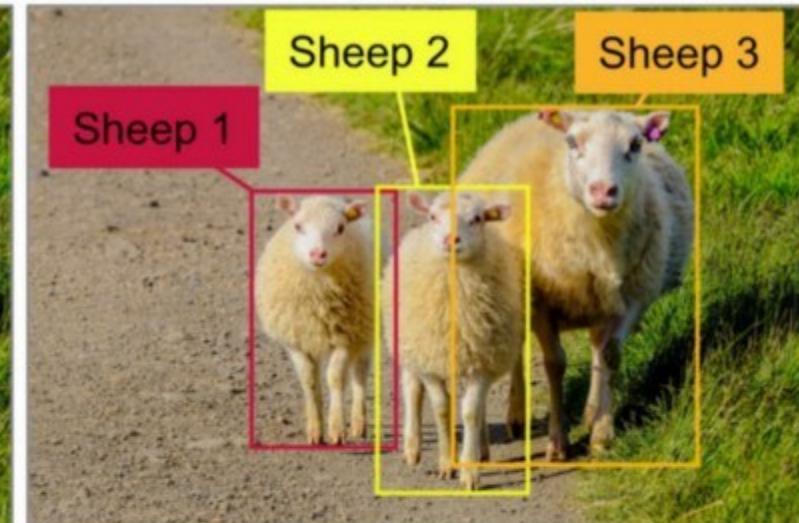
The Goal of Segmentation

- Separate image into coherent “objects”

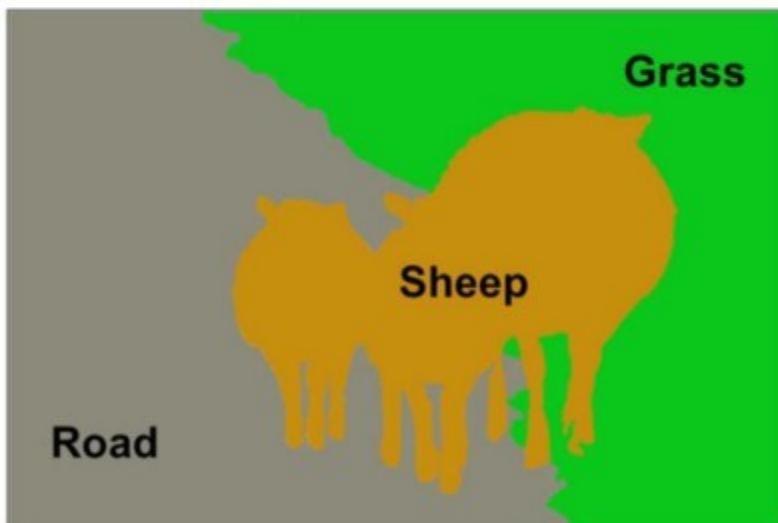




Classification + Localization



Object Detection



Semantic Segmentation



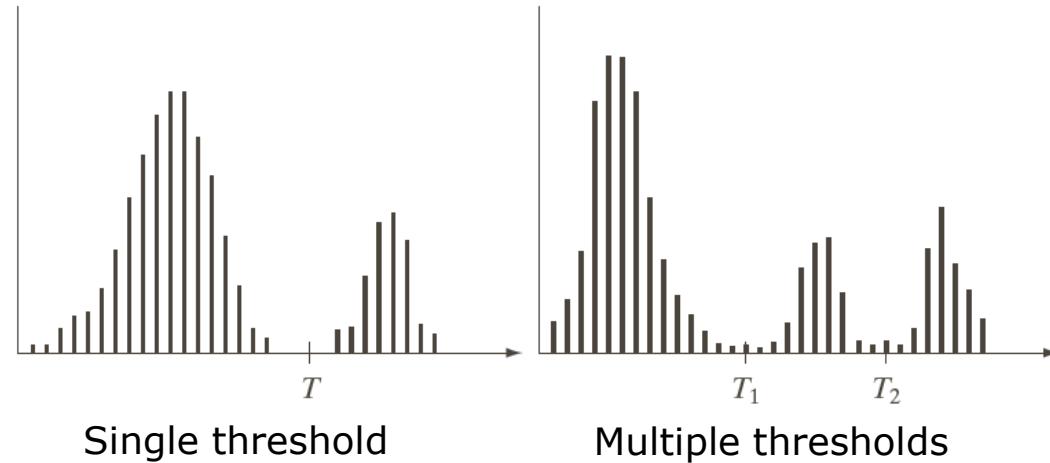
Instance Segmentation

Contents

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

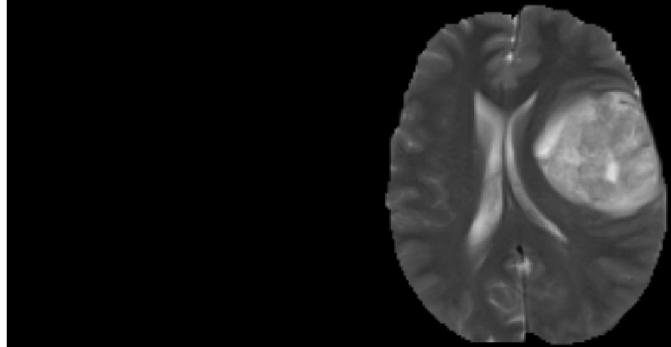
Thresholding

- Assumption: the range of intensity levels covered by objects of interest is different from the background.
- Basic Thresholding:
$$g(x, y) = \begin{cases} 0, & \text{if } I(x, y) < T \\ 1, & \text{otherwise} \end{cases}$$
- Multilevel Thresholding:
$$g(x, y) = \begin{cases} 0, & \text{if } I(x, y) < T_1 \text{ or } I(x, y) > T_2 \\ 1, & \text{otherwise} \end{cases}$$



Slice: 105

Window: 1836
Level: 918



VSD.Brain.XX.O.MR_T2.706

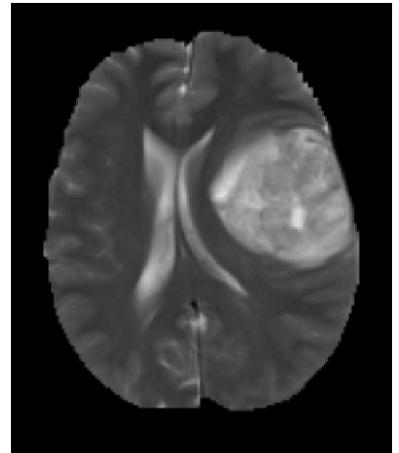
input

Window: 4
Level: 2



VSD.Brain_3more.XX.XX.OT.6566

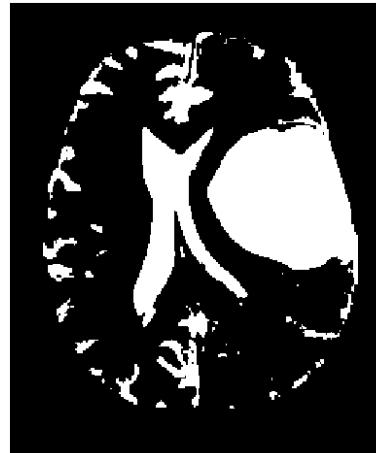
GT



cropped



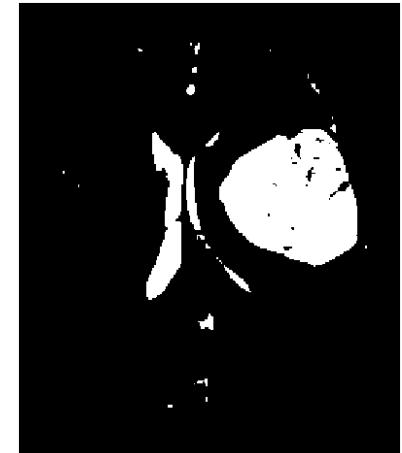
t=50



t=80



t=100



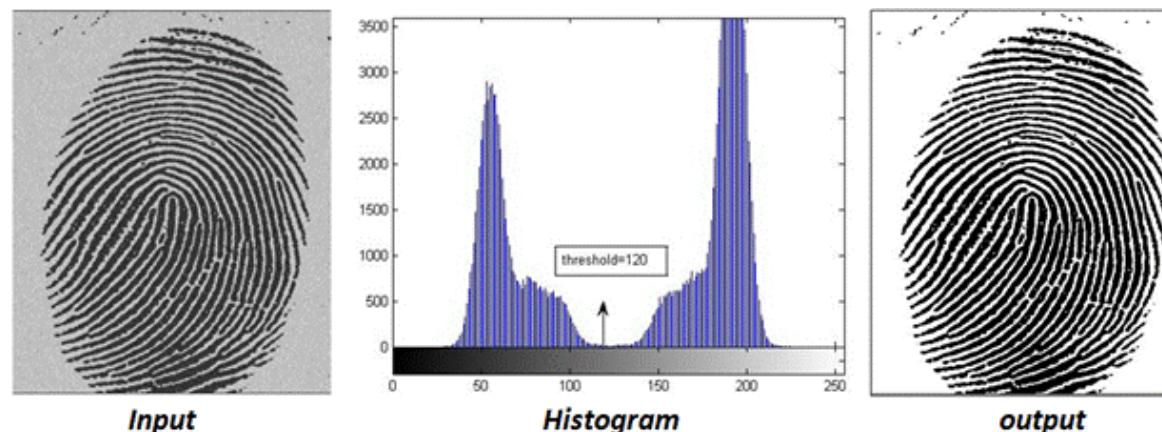
t=120



t=150

Automatic Threshold Determination (Iterative Thresholding)

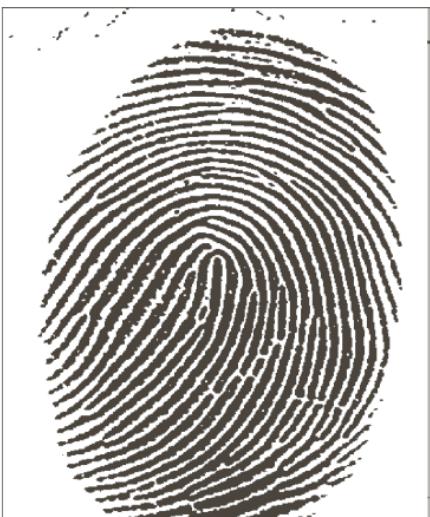
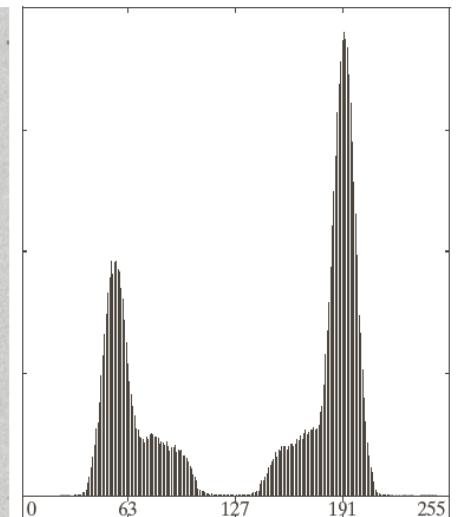
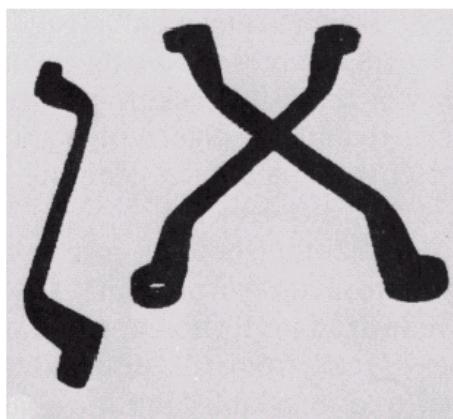
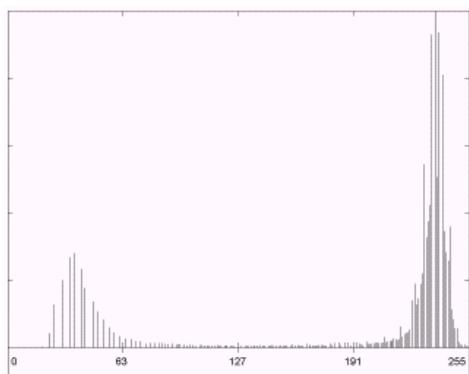
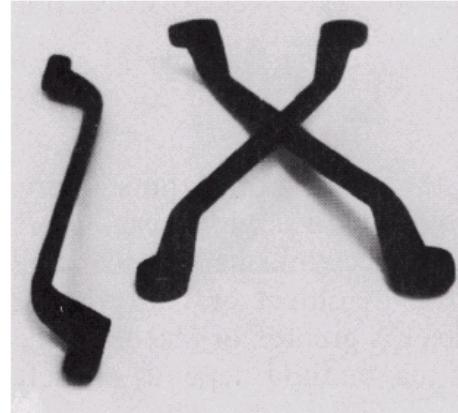
- The threshold value is repeatedly calculated based on:
 - the mean values of the foreground and background pixels
 - until the threshold converges, or until a maximum number of iterations is reached.
- Best for bimodal histogram



Iterative Thresholding Algorithm

- 1) Select an initial threshold t (e.g. mean or midway between maximum and minimum gray levels)
- 2) Segment image using t
- 3) Compute the average intensity m_1 and m_2 for the pixels in segmented regions R_1 and R_2 .
- 4) Compute a new threshold: $t = \frac{1}{2}(m_1 + m_2)$
- 5) Repeat steps 2-4 until the difference between successive values of t is very small

Thresholding Results



Effect of Noise on Thresholding

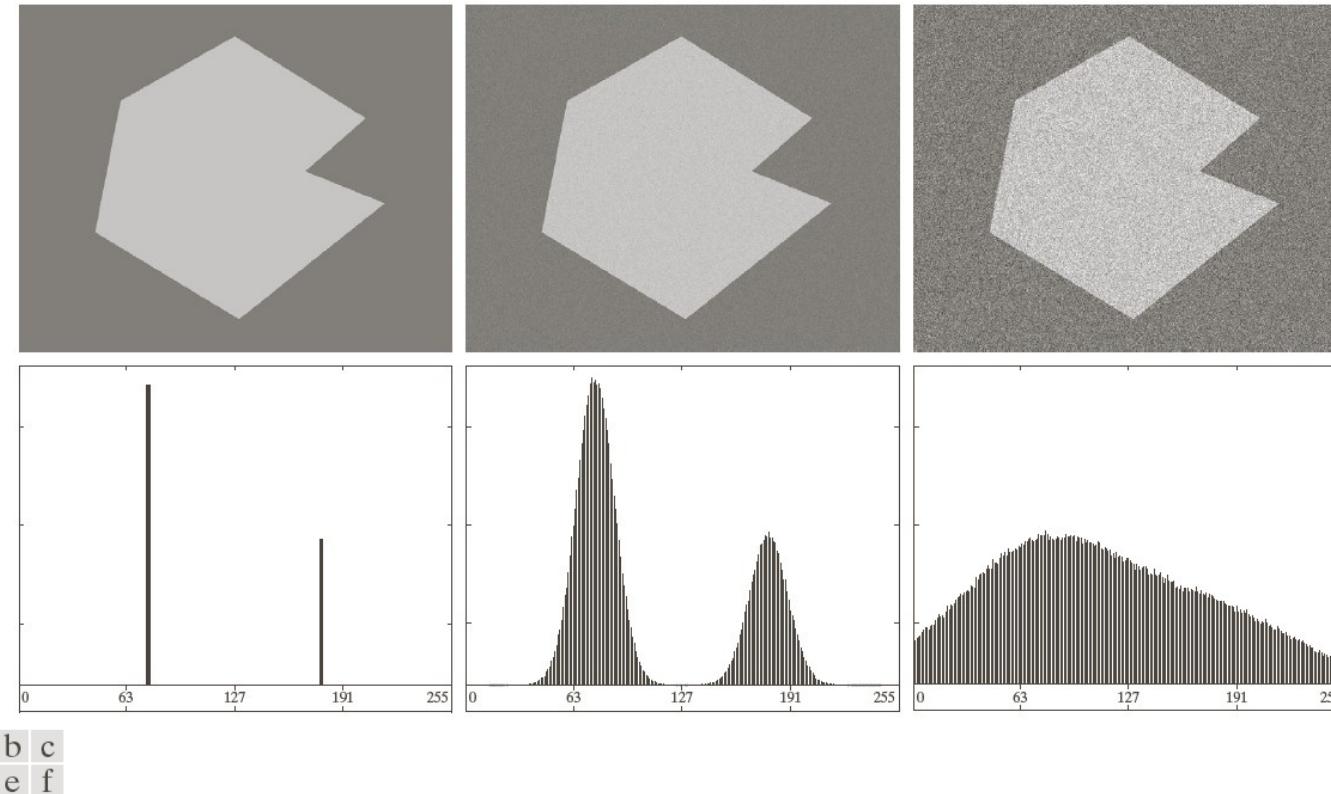
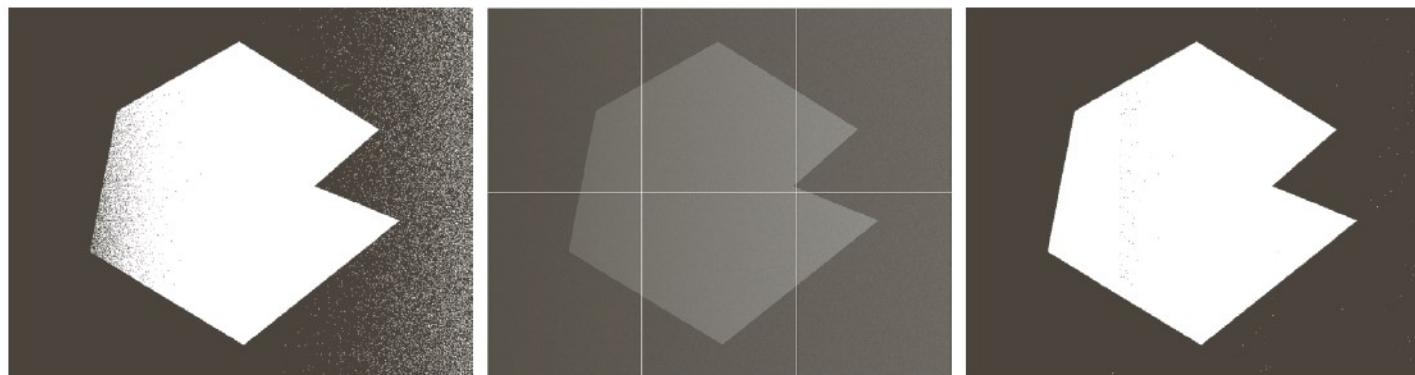
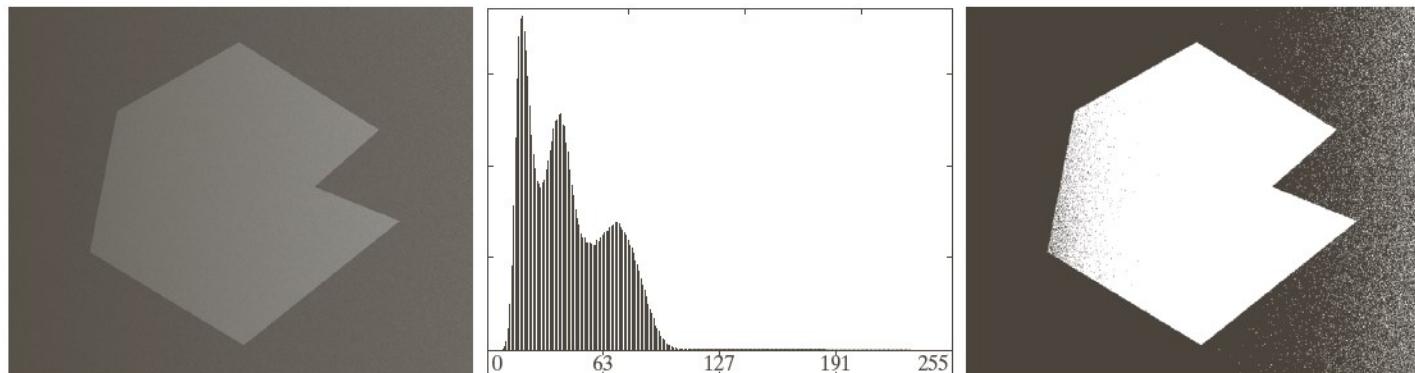


FIGURE 10.36 (a) Noiseless 8-bit image. (b) Image with additive Gaussian noise of mean 0 and standard deviation of 10 intensity levels. (c) Image with additive Gaussian noise of mean 0 and standard deviation of 50 intensity levels. (d)–(f) Corresponding histograms.

Adaptive/Local Thresholding



a b c
d e f

FIGURE 10.46 (a) Noisy, shaded image and (b) its histogram. (c) Segmentation of (a) using the iterative global algorithm from Section 10.3.2. (d) Result obtained using Otsu's method. (e) Image subdivided into six subimages. (f) Result of applying Otsu's method to each subimage individually.

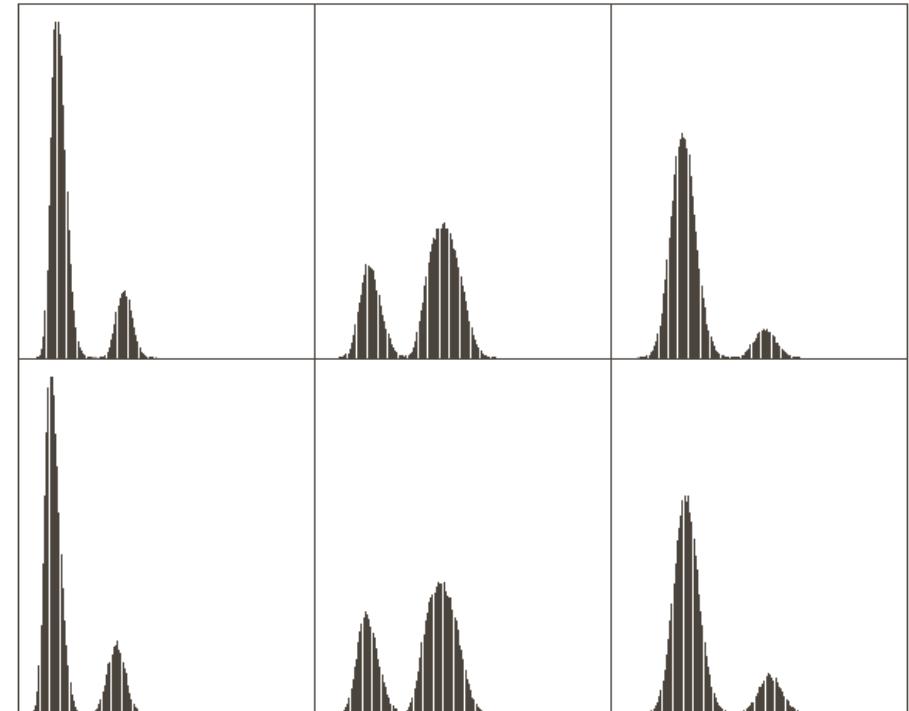
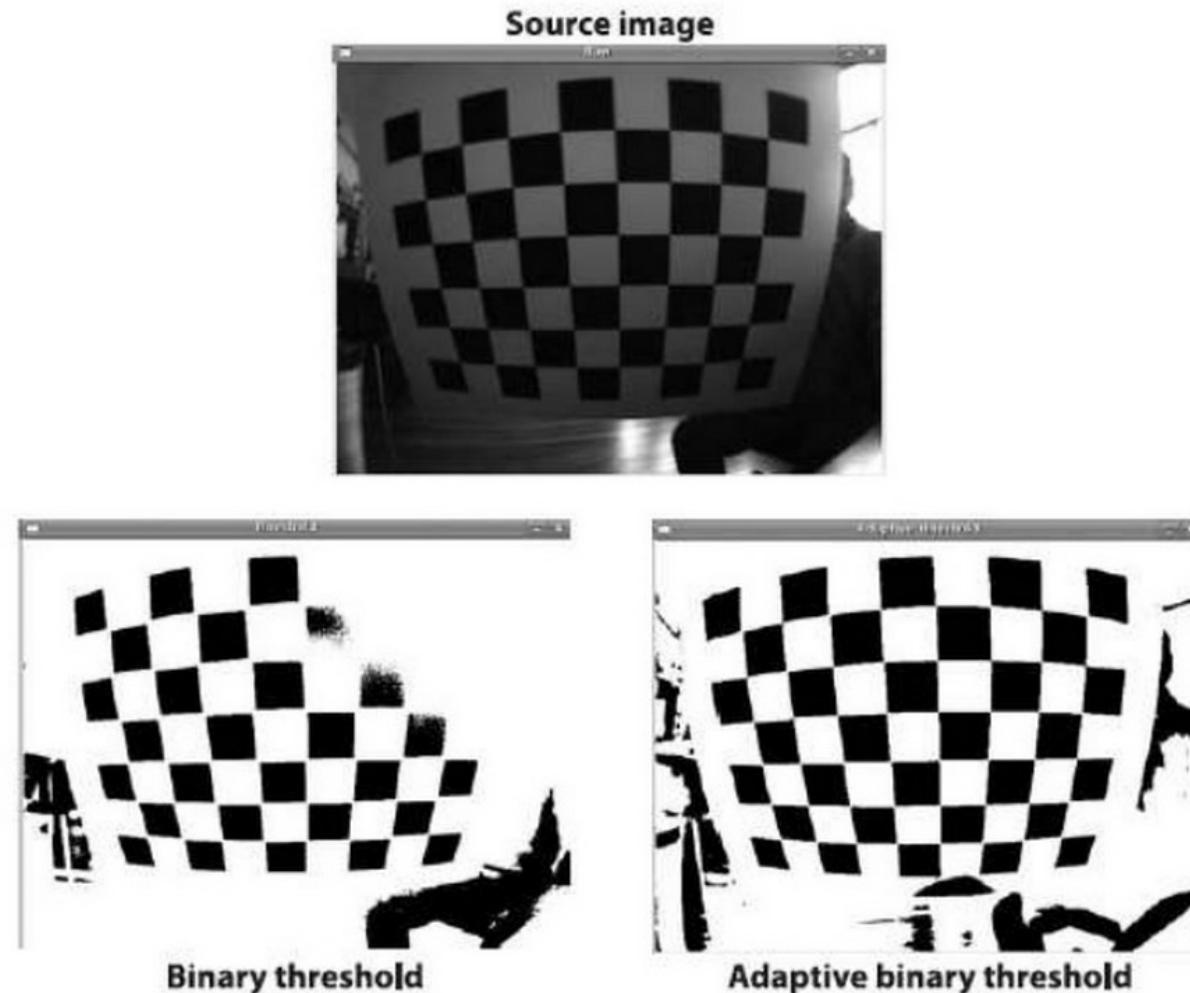


FIGURE 10.47
Histograms of the
six subimages in
Fig. 10.46(e).

Adaptive/Local Thresholding...



Contents

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

Edge Detection

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

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

Gradient

Prewitt

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

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

Sobel

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

Laplacian

0	-1	0
-1	4	-1
0	-1	0

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

Edge Detection

Marr Hildreth Edge Detector

1. Smooth image by Gaussian filter
2. Apply Laplacian to smoothed image



Combine in a single step
Laplacian of Gaussian (LoG)

3. Find zero crossings

- Scan along each row, record an edge point at the location of zero-crossing.
- Repeat above step along each column

Canny Edge Detector

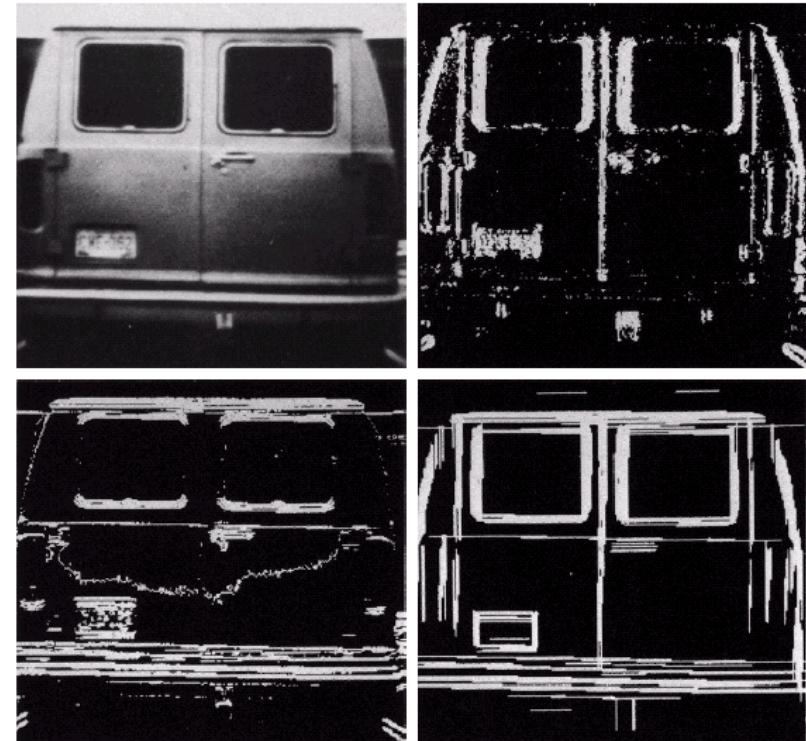
1. Smoothing
2. Differentiation
3. Non-Maximum Suppression
4. Hysteresis Thresholding



Combine in a single step:
Derivative of Gaussian (DoG)

Segmentation using Edge Detection

- Example: License Plate Segmentation
- Idea: Find rectangular shapes similar to license plate
 - 1. Find edges
 - 2. Connect edge points
 - 3. Find rectangular shapes
 - 4. Check horizontal-vertical proportion



Contents

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

Region-Based Segmentation

- The idea is to combine similar pixels in the same region
- We will discuss two important region-based segmentation techniques
 1. Region-Growing Segmentation
 2. Region Splitting and Merging Segmentation

Region-Growing Segmentation

- Region-Grouping
 - Group pixels or sub-regions into larger regions based on predefined criteria.
- Basic Formulation
 - Let I represent the entire image region. Segmentation is to partition I into n subregions, R_1, R_2, \dots, R_n such that
 1. Segmentation must be complete $\bigcup_{i=1}^n R_i = I$
 2. R_i is a connected region, $i = 1, 2, \dots, n$ in some predefined sense
 3. $R_i \cap R_j = \emptyset$ for all i and $j, i \neq j$
 4. $P(R_i) = \text{TRUE}$ for $i = 1, 2, \dots, n$
 - using some homogeneity criterion
 5. $P(R_i \cup R_j) = \text{FALSE}$ for adjacent regions R_i, R_j

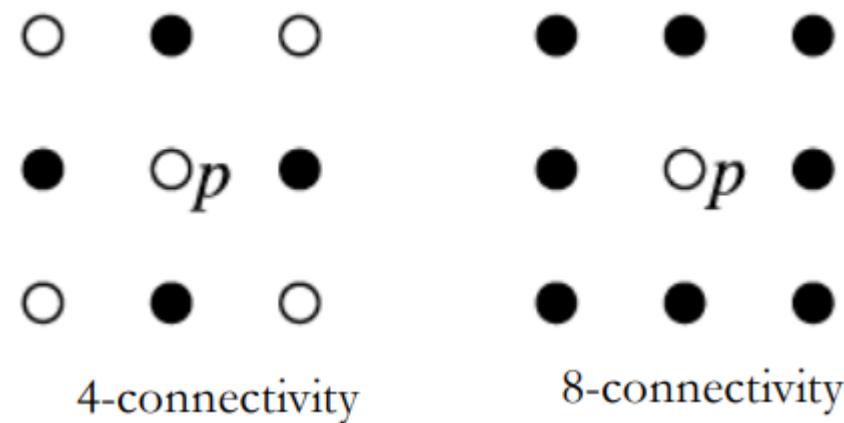
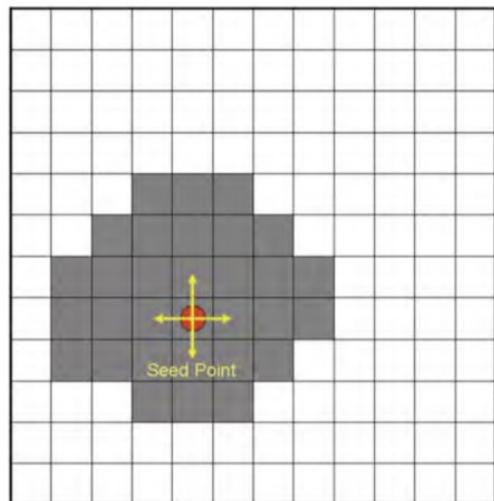
Region-Growing Segmentation

- Examples of homogeneity criteria for region R
 - diff. between max and min grey-values in R is small
 - diff. between any pixel and mean grey-value in R is small
 - variance of grey-values in R is small

Region-Growing Segmentation

□ Pseudocode

1. Choose the seed pixel(s)
2. Check the neighboring pixels and add them to the region if they are similar to the seed
3. Repeat step 2 for each of the newly added pixels; stop if no more pixels can be added.



Example 1

0	1	2	0
2	5	6	1
1	4	7	3
0	2	5	1

Seed Point = 7
T=2
4-connectivity
Homogeneity criterion: $(p_i - \text{seed}) \leq T$

0	1	2	0
2	5	6	1
1	4	7	3
0	2	5	1

Example 2

0	0	5	6	7
1	1	5	8	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

Image, 2 seeds

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

Result for T=4

a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a

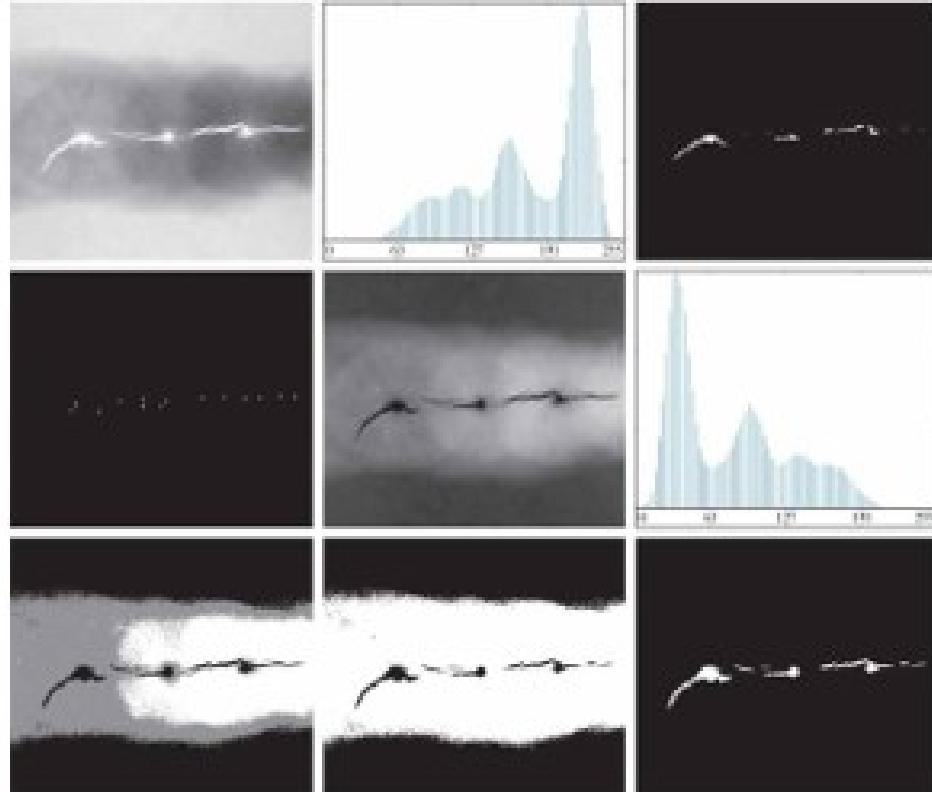
Result for T=8

- Homogeneity criterion: maximum allowed absolute difference T within region
- 8-connectivity

How do we choose the seed(s) in practice ?

- It depends on the nature of the problem.
 - If targets need to be detected using infrared images for example, choose the brightest pixel(s).
 - Without a-priori knowledge, compute the histogram and choose the gray-level values corresponding to the strongest peaks

Region Growing Example

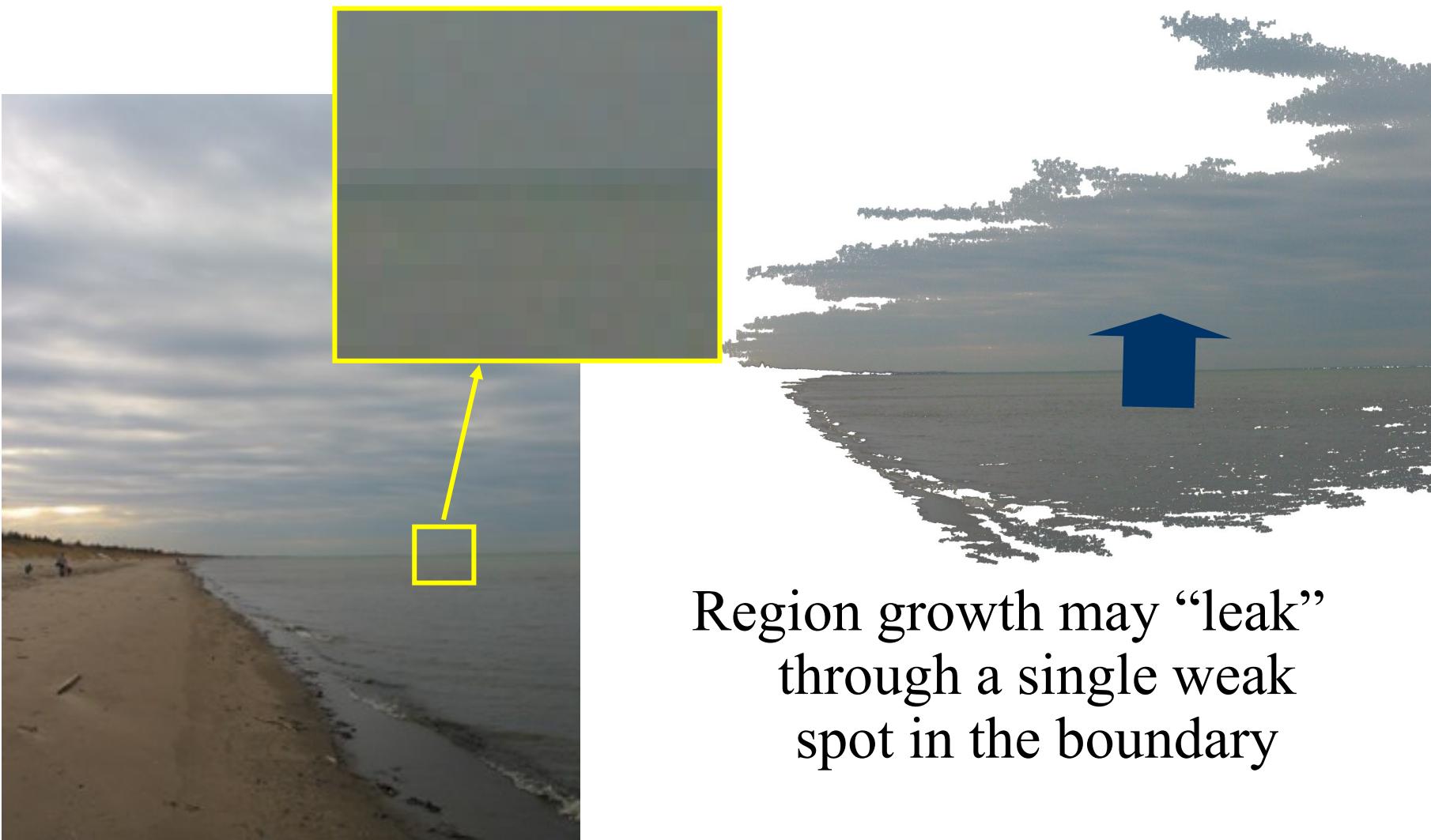


- (a) X-ray image of a defective weld.
- (b) Histogram.
- (c) Initial seed image.
- (d) Final seed image (the points were enlarged for clarity).
- (e) Absolute value of the difference between the seed value (255) and (a).
- (f) Histogram of (e).
- (g) Difference image thresholded using dual thresholds.
- (h) Difference image thresholded with the smallest of the dual thresholds.
- (i) Segmentation result obtained by region growing.

a	b	c
d	e	f
g	h	i

Region Growing Disadvantages

- Its not trivial to find good seed points
 - Different seed points will give different results
- Region growth may “leak” through a single weak spot in the boundary

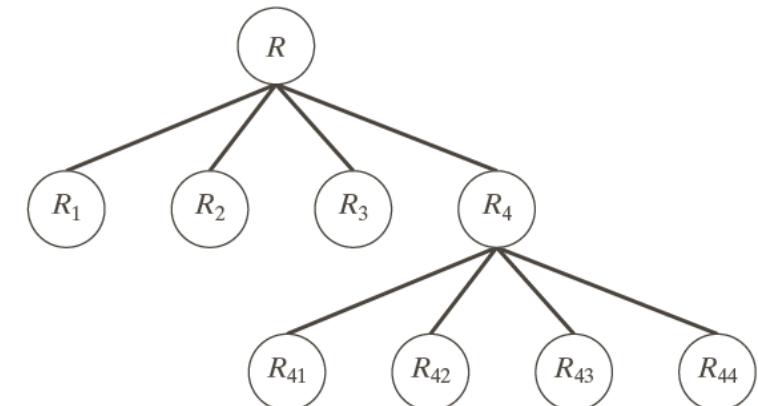


Region growth may “leak”
through a single weak
spot in the boundary

Region Splitting and Merging

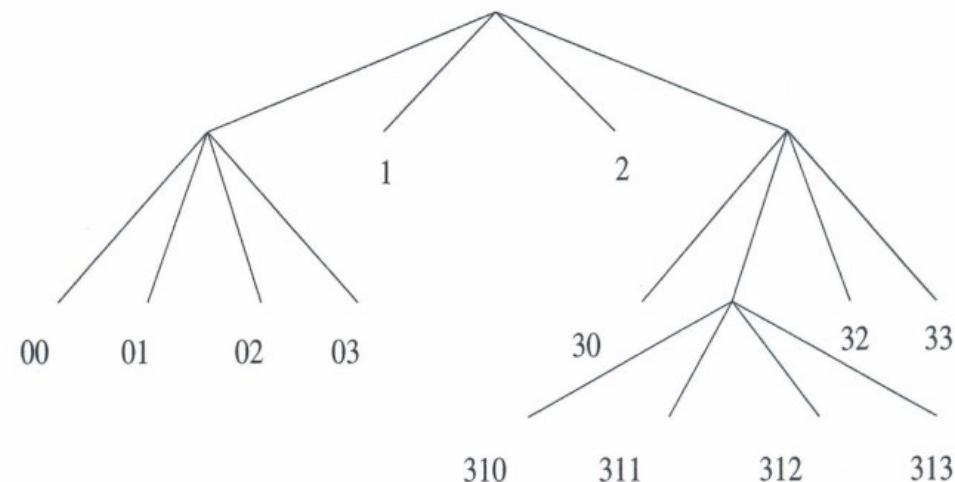
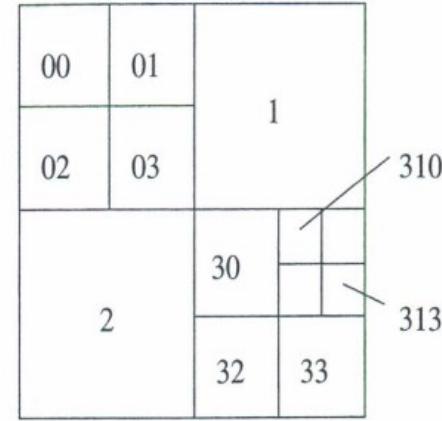
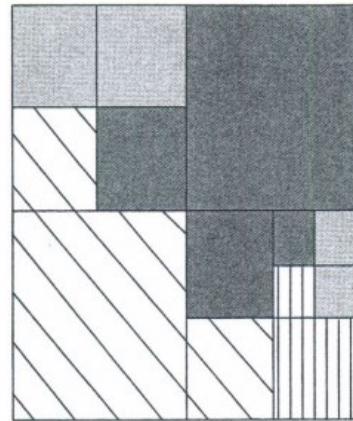
- Another important region-based segmentation technique
- START: consider entire image as one region
- 1. If region satisfies homogeneity criteria, leave it unmodified
- 2. If not, split it into four quadrants and recursively apply 1 and 2 to each newly generated region.
STOP when all regions in the **quadtree** satisfy the homogeneity criterion
- 3. If any two adjacent regions R_i, R_j can be merged into a homogeneous region, merge them. STOP when no merging is possible anymore.

R_1	R_2
R_3	R_{41} R_{42}
	R_{43} R_{44}



Quadtree formation

Segmentation Quadtree



Region Splitting and Merging

1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0	
1	1	8	8	8	4	1	0	
1	1	6	6	6	3	1	0	
1	1	5	6	6	3	1	0	
1	1	5	6	6	2	1	0	
1	1	1	1	1	1	0	0	

1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0	
1	1	8	8	8	4	1	0	
1	1	6	6	6	3	1	0	
1	1	5	6	6	3	1	0	
1	1	5	6	6	2	1	0	
1	1	1	1	1	1	0	0	

1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0	
1	1	8	8	8	4	1	0	
1	1	6	6	6	3	1	0	
1	1	5	6	6	3	1	0	
1	1	5	6	6	2	1	0	
1	1	1	1	1	1	1	0	0

1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0	
1	1	8	8	8	4	1	0	
1	1	6	6	6	3	1	0	
1	1	5	6	6	3	1	0	
1	1	5	6	6	2	1	0	
1	1	1	1	1	1	1	0	0

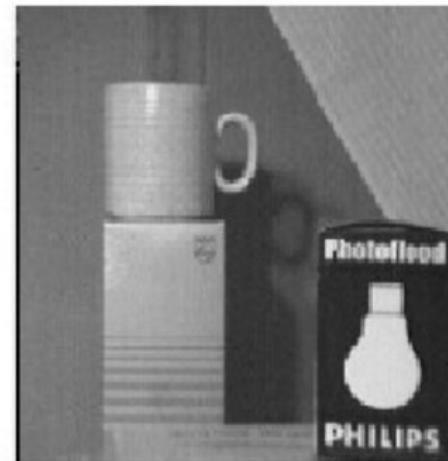
Split: t=1

1	1	1	1	1	1	1	1	2
1	1	1	1	1	1	1	1	0
3	1	4	9	9	8	1	0	
1	1	8	8	8	4	1	0	
1	1	6	6	6	3	1	0	
1	1	5	6	6	3	1	0	
1	1	5	6	6	2	1	0	
1	1	1	1	1	1	1	0	0

merge

Region Splitting and Merging

input



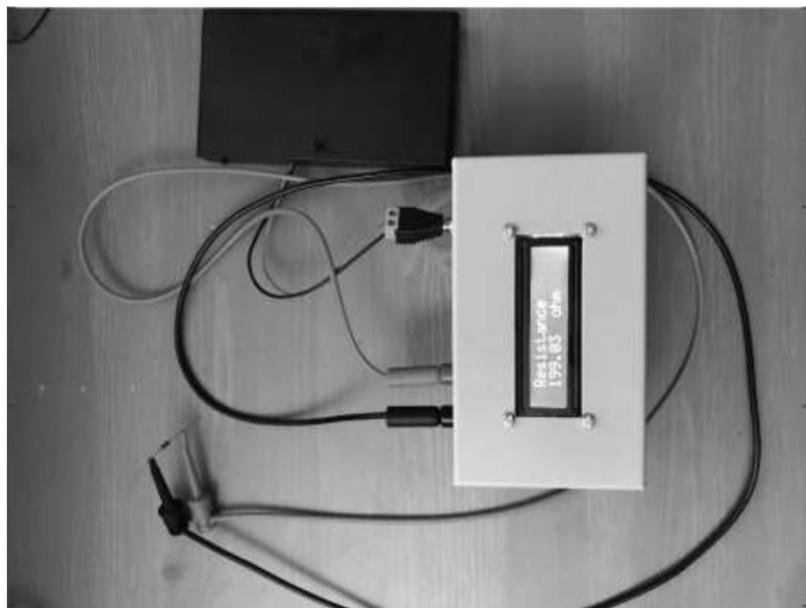
After split



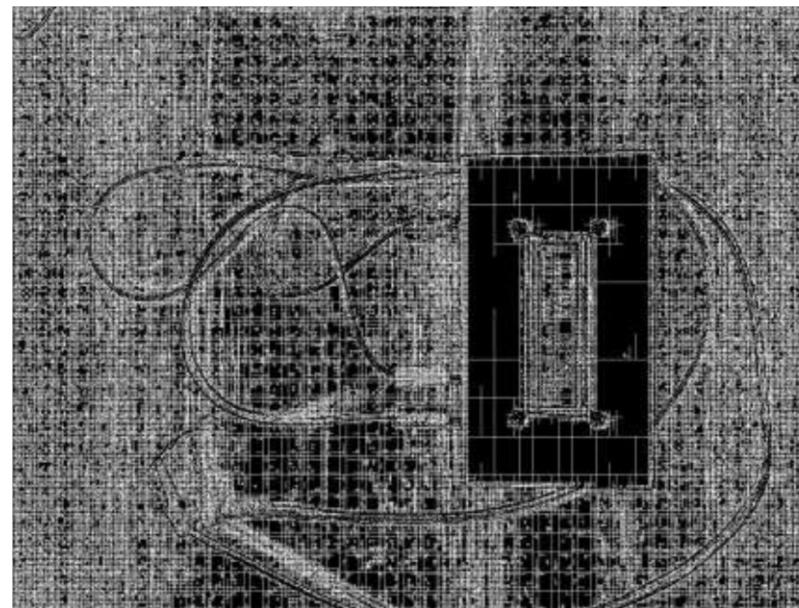
After merge



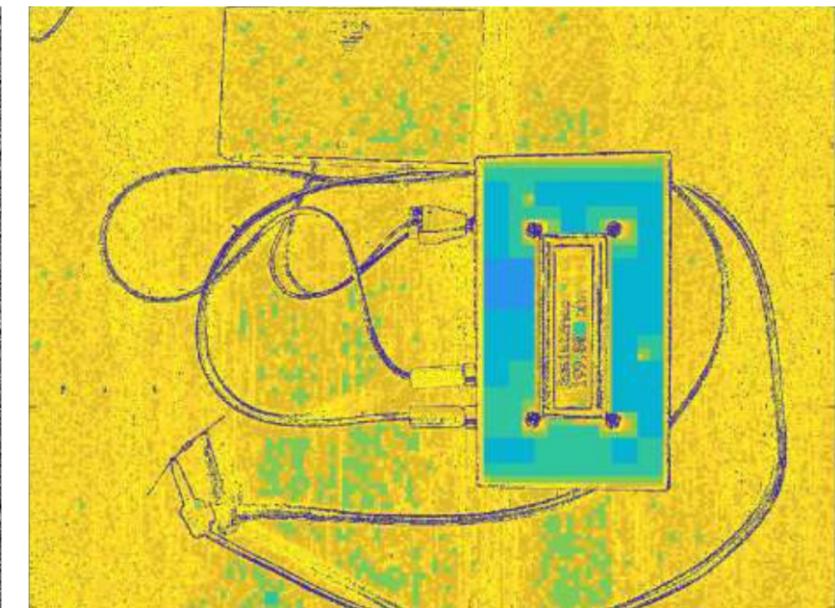
Region Splitting and Merging



input



After split



After merge

For Next Time...

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

Resources

- Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129-136, 1982
- D. Comaniciu, and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.
- Slides sources
 - Fei-Fei Li
(http://vision.stanford.edu/teaching/cs131_fall1617/lectures/lecture13_kmeans_mean_shift_cs131_2016)
 - Sergei Vassilvitskii and David Arthur
(<http://theory.stanford.edu/~sergei/slides/BATS-Means.pdf>)
 - Yaron Ukrainitz & Bernard Sarel
(http://www.wisdom.weizmann.ac.il/~vision/courses/2004_2/files/mean_shift/mean_shift.ppt)
 - <http://robot-develop.org/wp-content/uploads/2012/03/seg3.pdf>

Image Processing & Pattern Recognition

Image Segmentation and Clustering Part 2: Clustering

Dr. Zia Ud Din

Outline

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

Contents

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

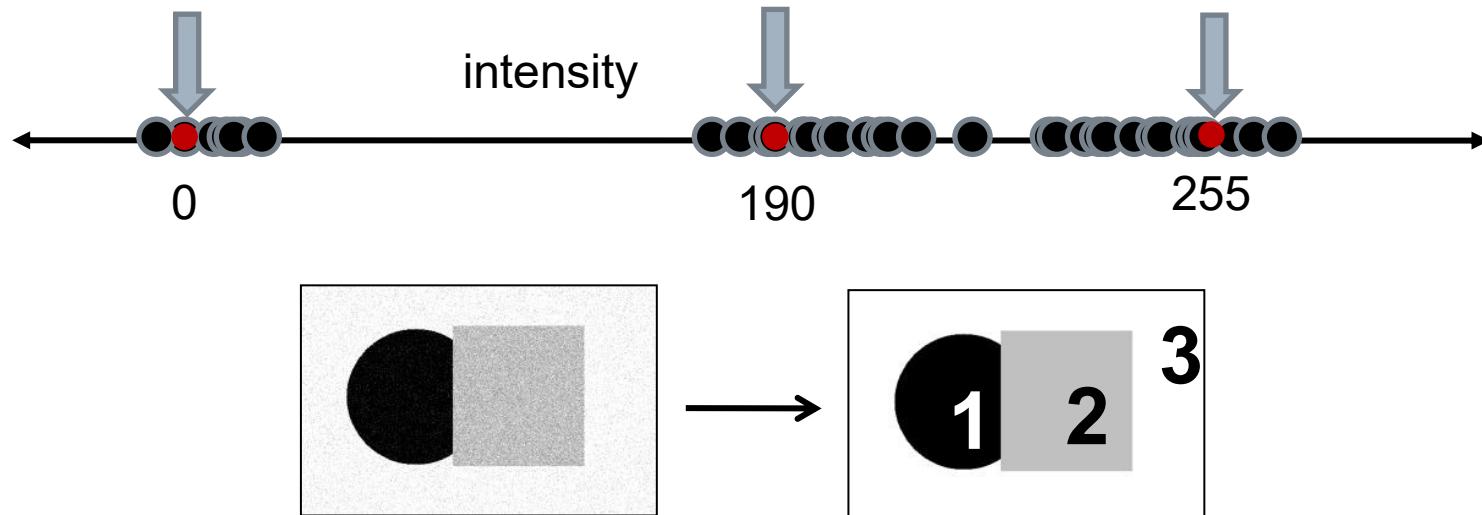
Clustering

- Clustering: grouping together similar points and representing them with a single token
- Criteria for forming a cluster depends on the application
- Some examples:
 - Intensity
 - Colour
 - Texture

Contents

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

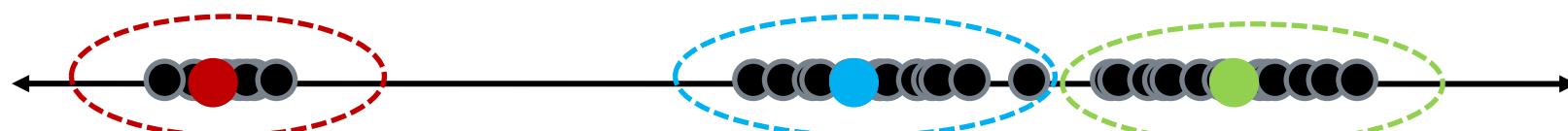
K-Means Clustering



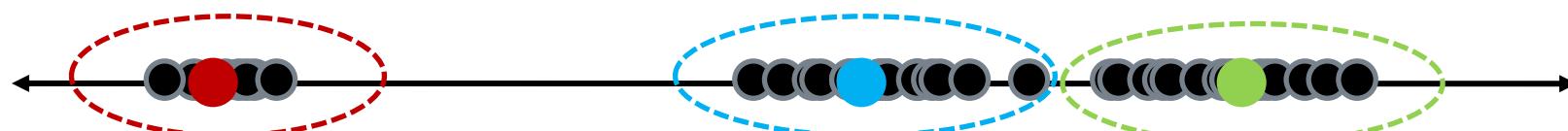
- Goal: choose three “**centers**” as the representative intensities, and label every pixel according to which of these centers it is **nearest** to.

K-Means Clustering

- “chicken and egg problem”
- If we knew the cluster centers, we could allocate points to groups by assigning each to its closest center



- If we knew the group memberships, we could get the centers by computing the mean per group

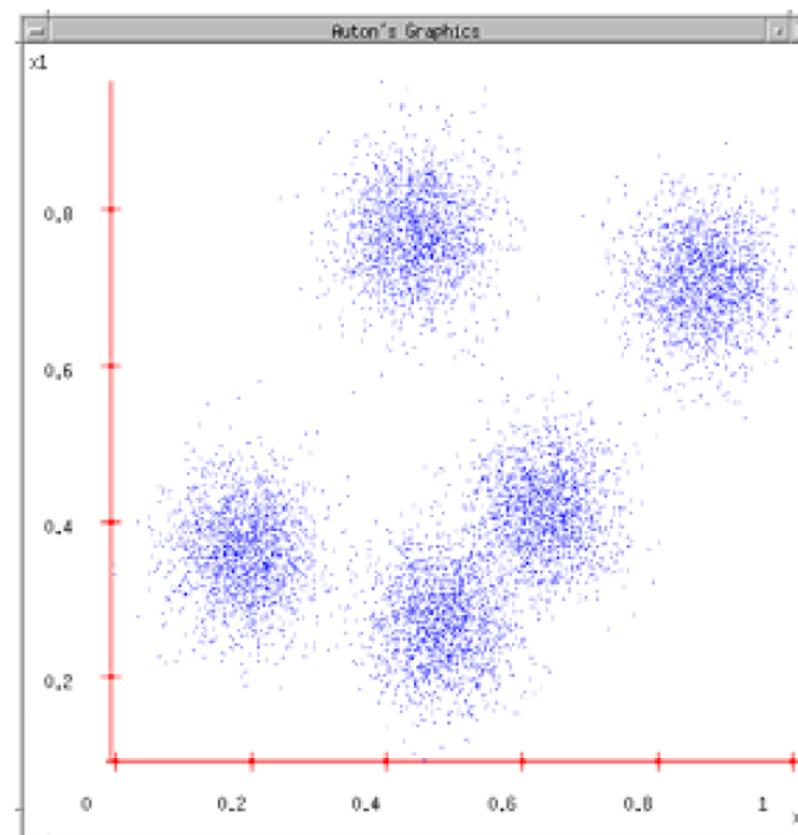


K-Means Clustering

Algorithm:

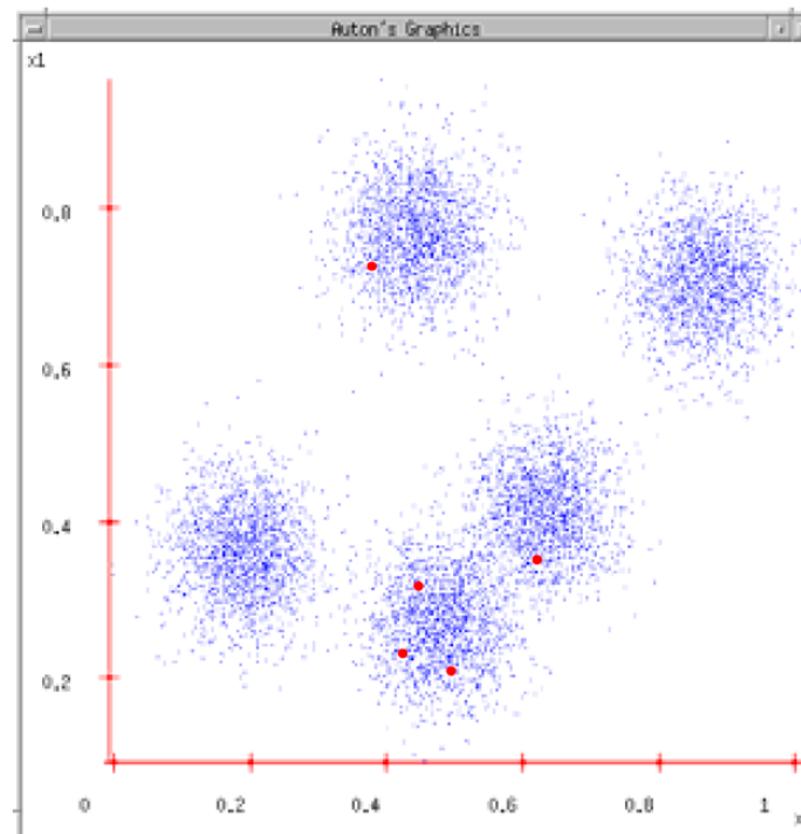
1. Randomly initialize k points to act as cluster centers (c_1, \dots, c_k)
2. Given cluster centers, determine points in each cluster
3. Given points in each cluster, solve for c_j
4. If c_j have changed, repeat from Step 2

Example: K-Means Clustering



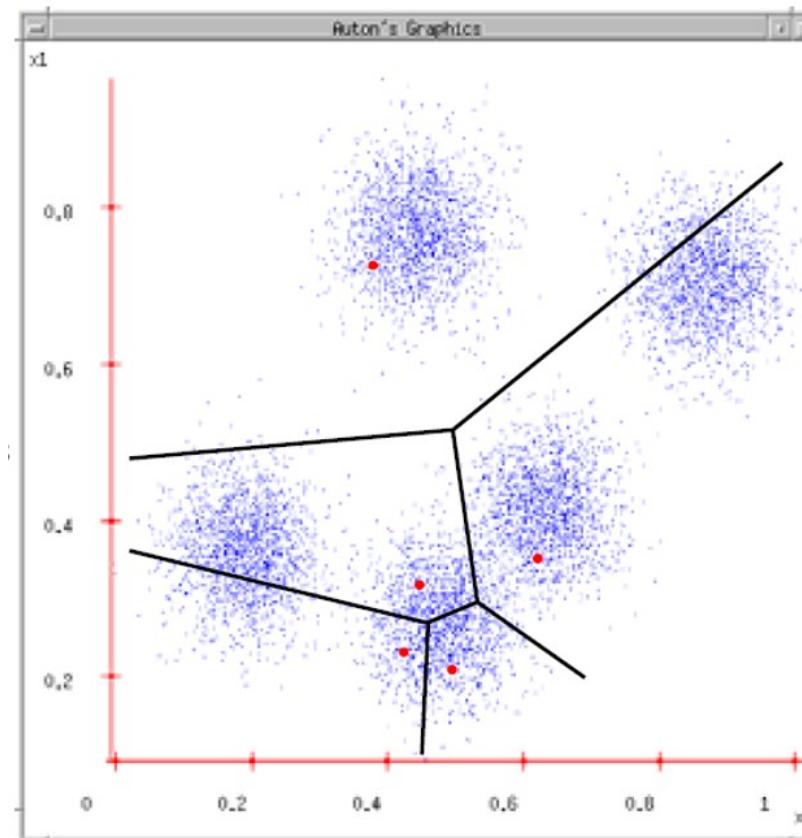
Example: K-Means Clustering

- Step 1: Randomly initialize k points to act as cluster centers



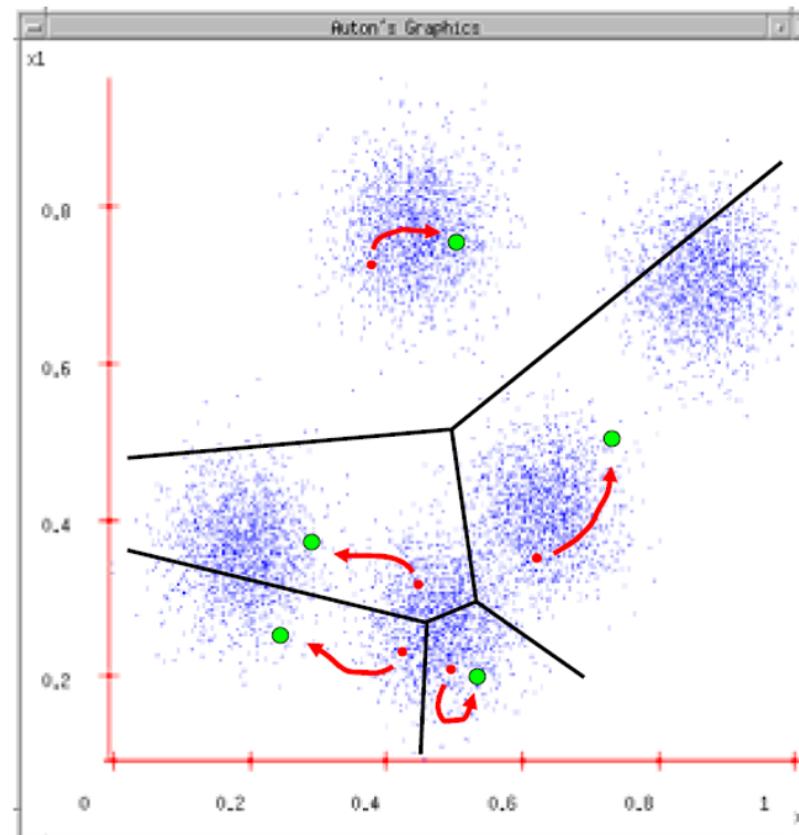
Example: K-Means Clustering

- Step 2: Given cluster centers, determine points in each cluster



Example: K-Means Clustering

- Step 3: Given points in each cluster, solve for c_j

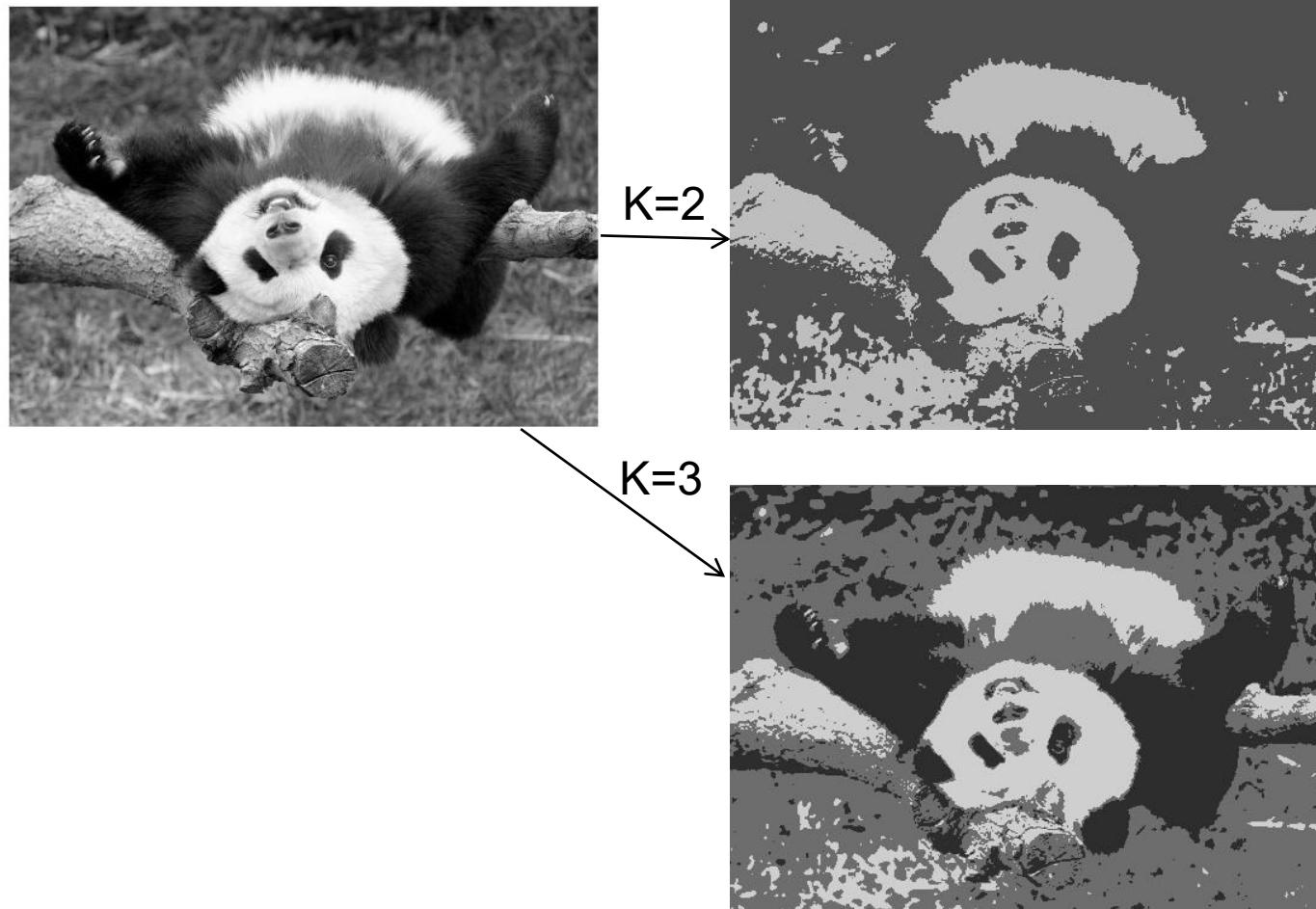


Step 4: If c_j have changed, repeat from Step 2

K-Means Clustering

- Online demo:
 - <https://www.naftaliharris.com/blog/visualizing-k-means-clustering/>

Segmentation as Clustering



Segmentation as Clustering

- Depending on what we choose as the **feature space**, the pixels can be grouped in different ways
- Grouping pixels based on **intensity** similarity:

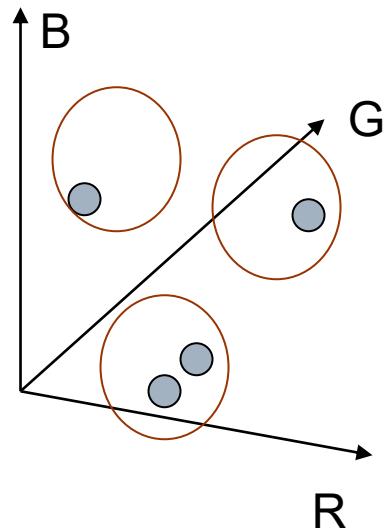


Feature space: intensity value (1-d)

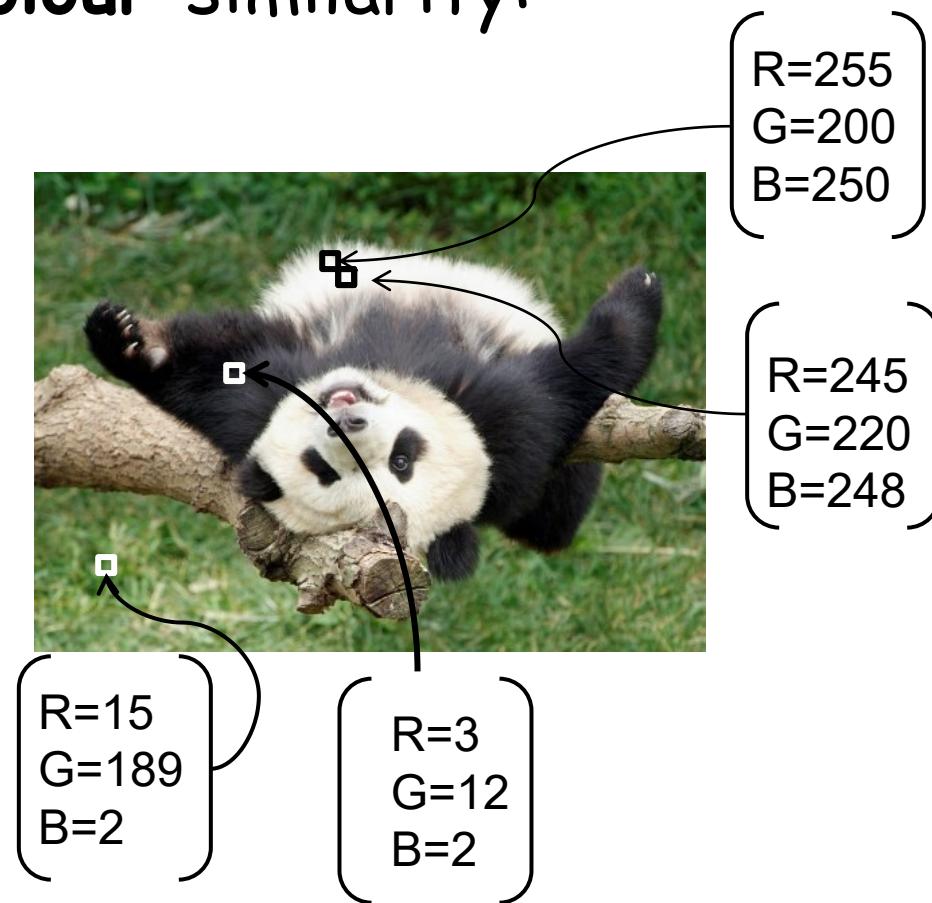


Segmentation as Clustering

- Grouping pixels based on colour similarity:



Feature space: intensity value (3-d)



Segmentation as Clustering

- Grouping pixels based on **texture** similarity



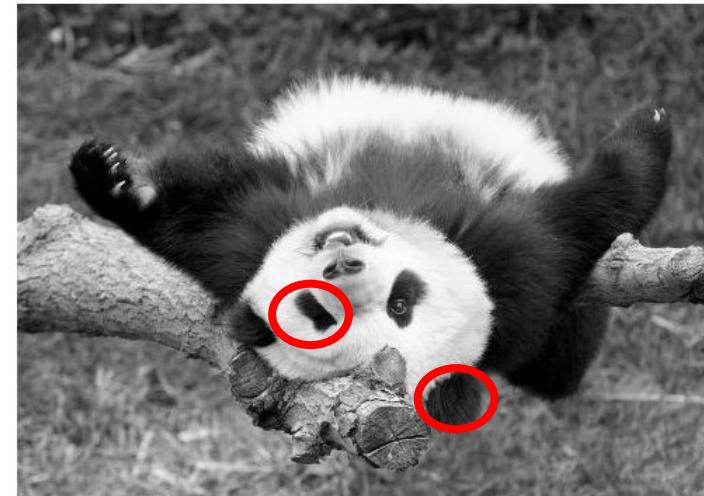
Feature space: filter bank responses (e.g., 24D)

Segmentation as Clustering

- Returning to grouping pixels based on **intensity** similarity
- In what case is intensity not enough for segmentation?



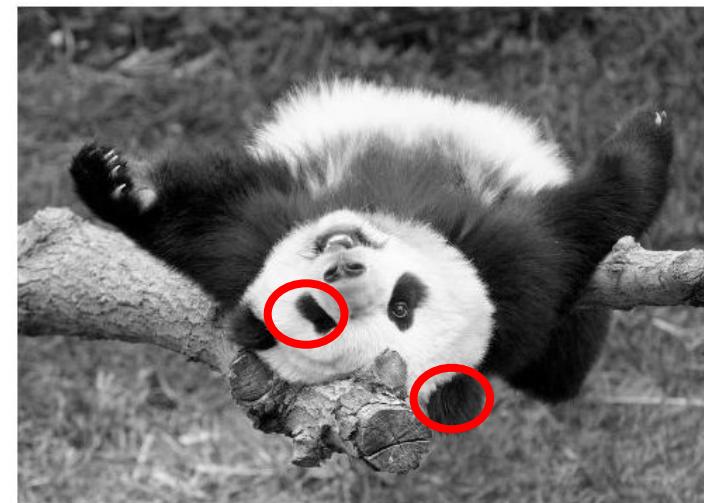
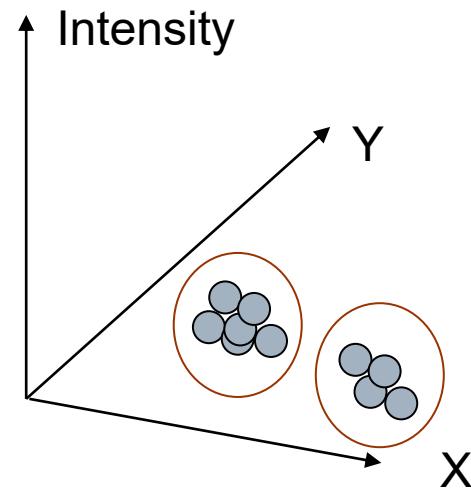
Feature space: intensity
value (1-d)



- Clusters based on intensity similarity don't have to be spatially coherent

Segmentation as Clustering

- Grouping pixels based on **intensity & position** similarity:



- Both regions are black
- Including **position (x,y)** information allows us to group the regions into two distinct segments

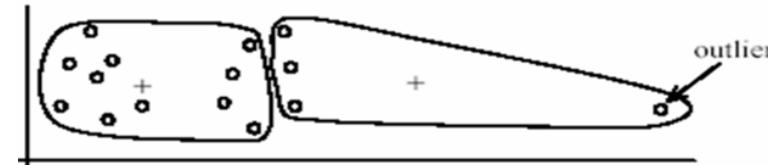
K-Means Clustering

□ Pros:

- Simple and fast
- Easy to implement
- Converges to local minimum of within-cluster squared error

□ Cons:

- Specifying k
- Sensitive to initial centers
- Sensitive to outliers
- Detects spherical clusters only



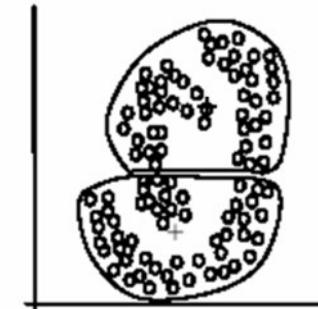
(A): Undesirable clusters



(B): Ideal clusters



(A): Two natural clusters



(B): k -means clusters

Contents

- Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
- Clustering
 - K-means
 - Mean Shift

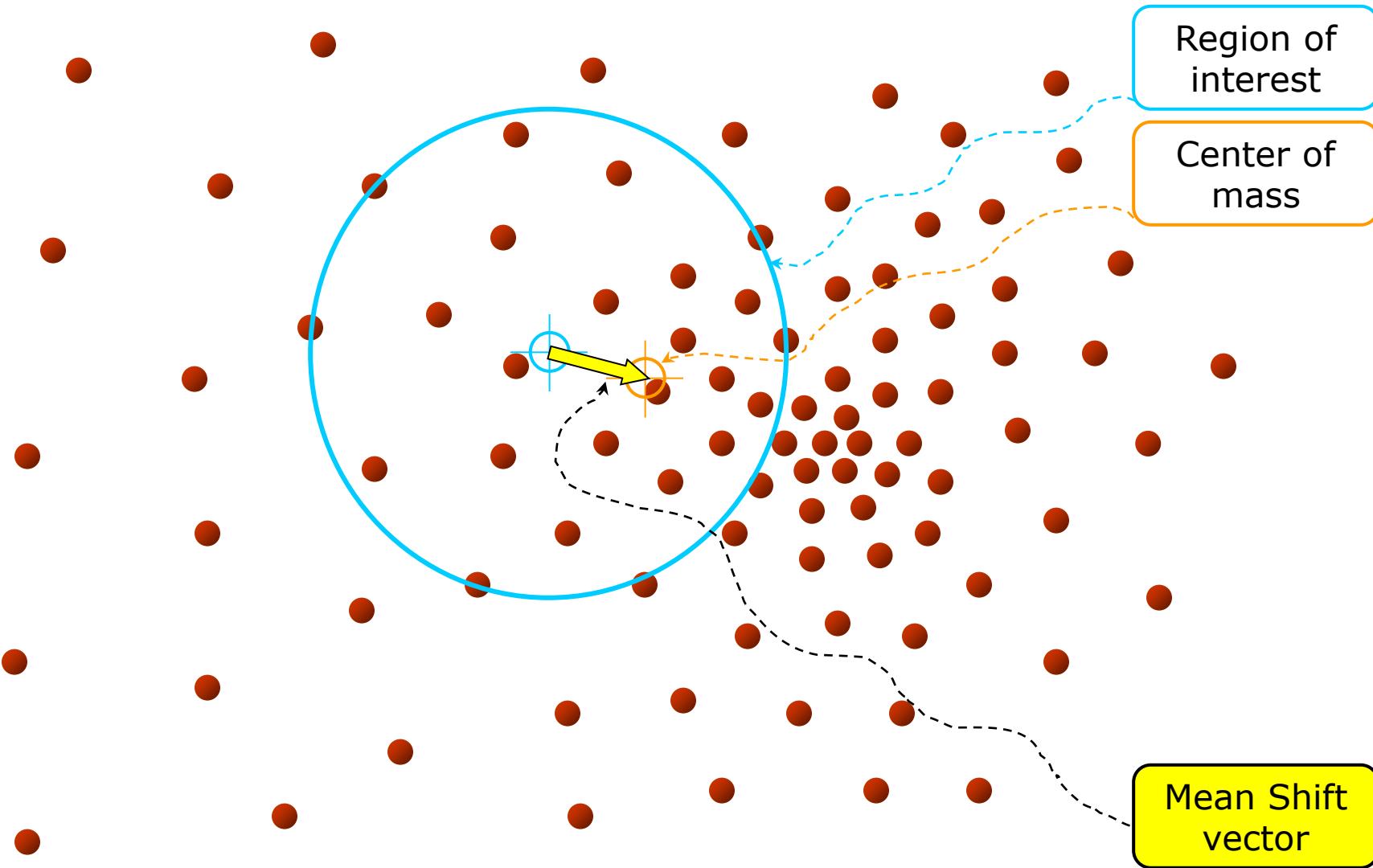
Mean Shift Clustering

- We saw that there were certain issues with k-means:
- 1. We need to know in advance how many clusters are there in the data
- 2. Sensitivity to initialization
- 3. Can not find non-spherical clusters

- Mean shift improves k-means on all these aspects.
- It is a versatile technique for clustering-based segmentation

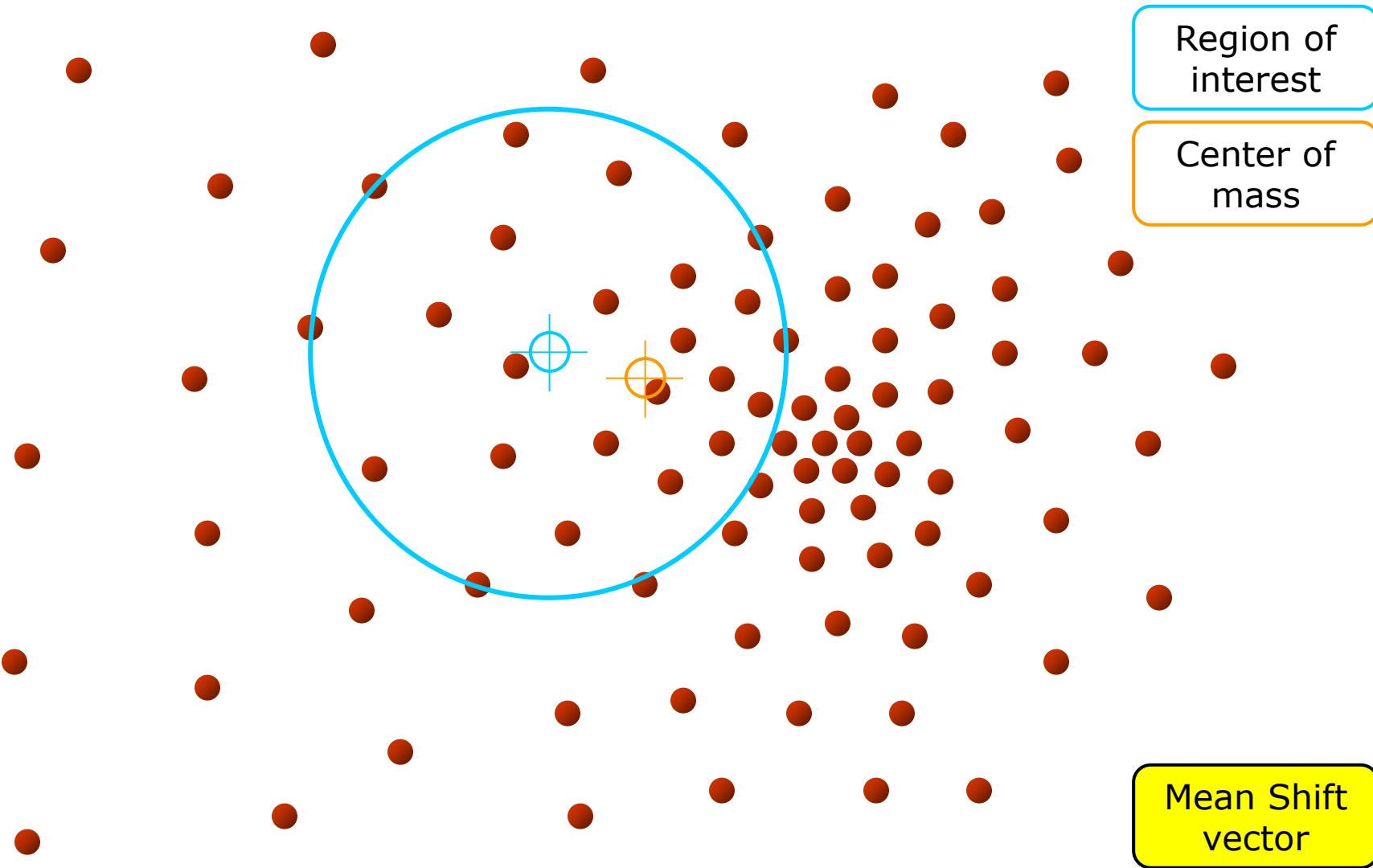
Mean Shift Clustering: Algorithm

- For each point:
 1. Center a window on that point
 2. Compute the mean of the data in the search window
 3. Center the search window at the new mean location
 4. Repeat (2,3) until convergence
 - The center of window at convergence is the mode for that point
- Assign points that lead to nearby modes to the same cluster



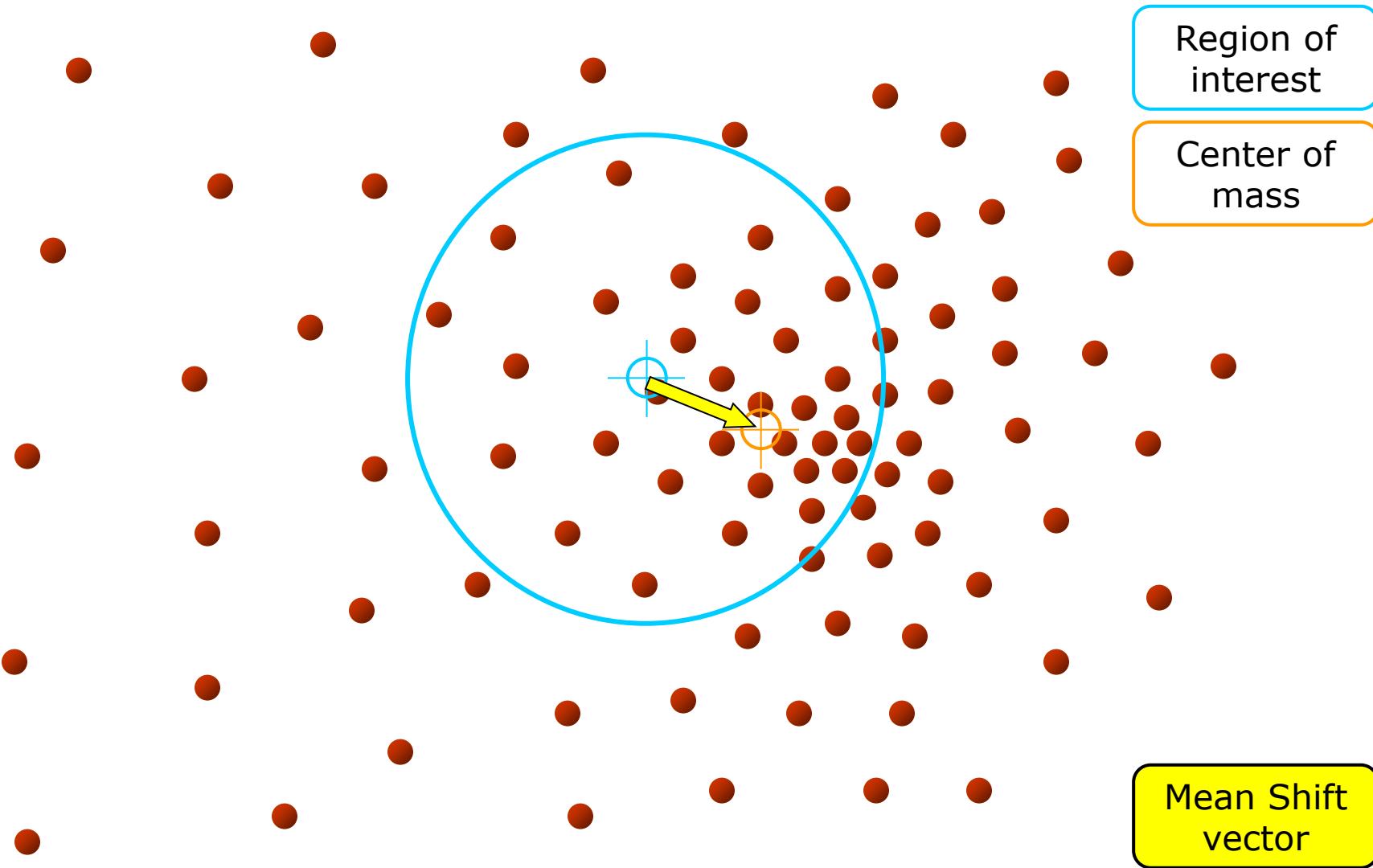
Objective : Find the densest region
Distribution of identical billiard balls

Slide by Y. Ukrainitz & B. Sarel



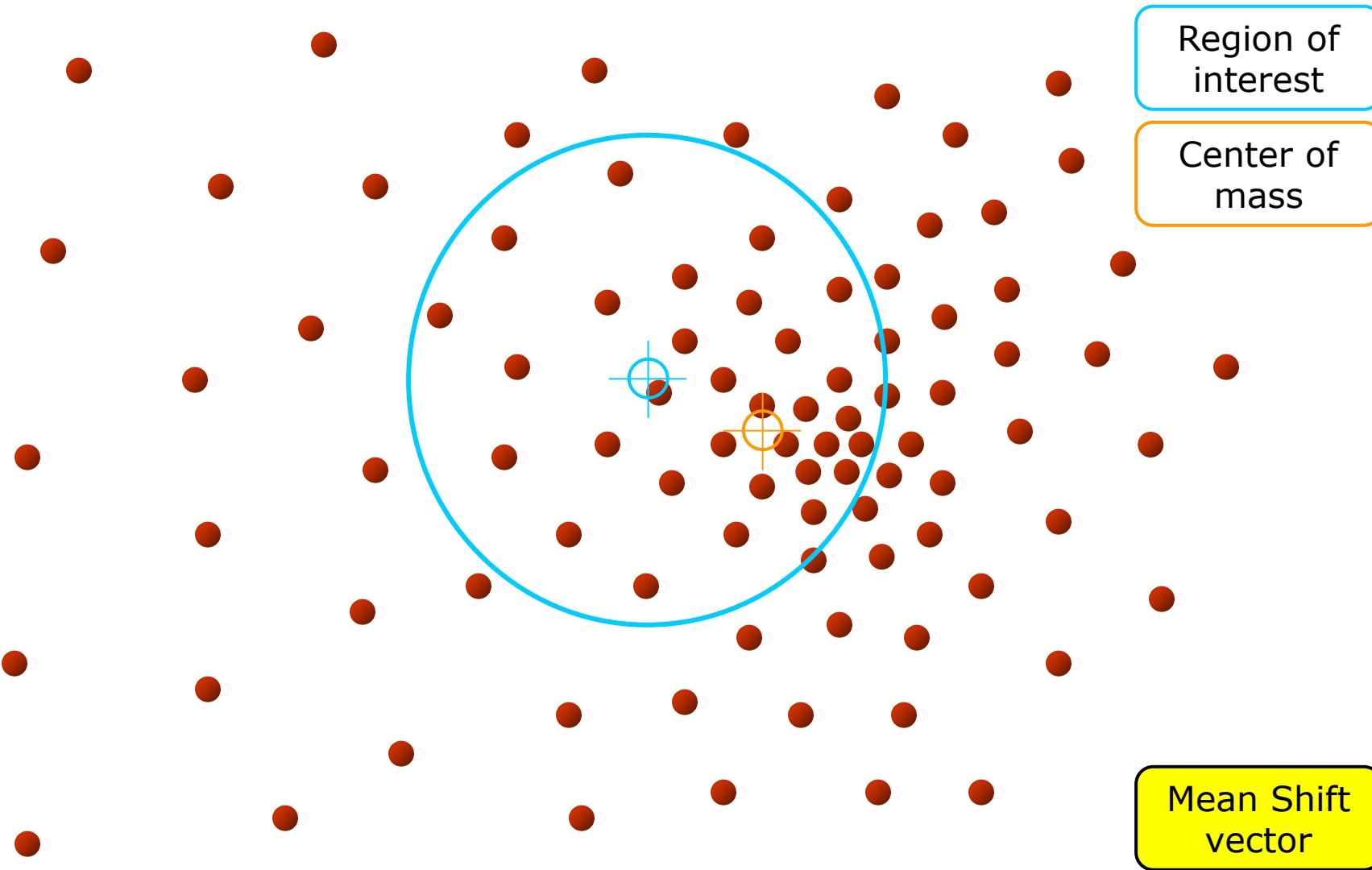
Objective : Find the densest region
Distribution of identical billiard balls

Slide by Y. Ukrainitz & B. Sarel



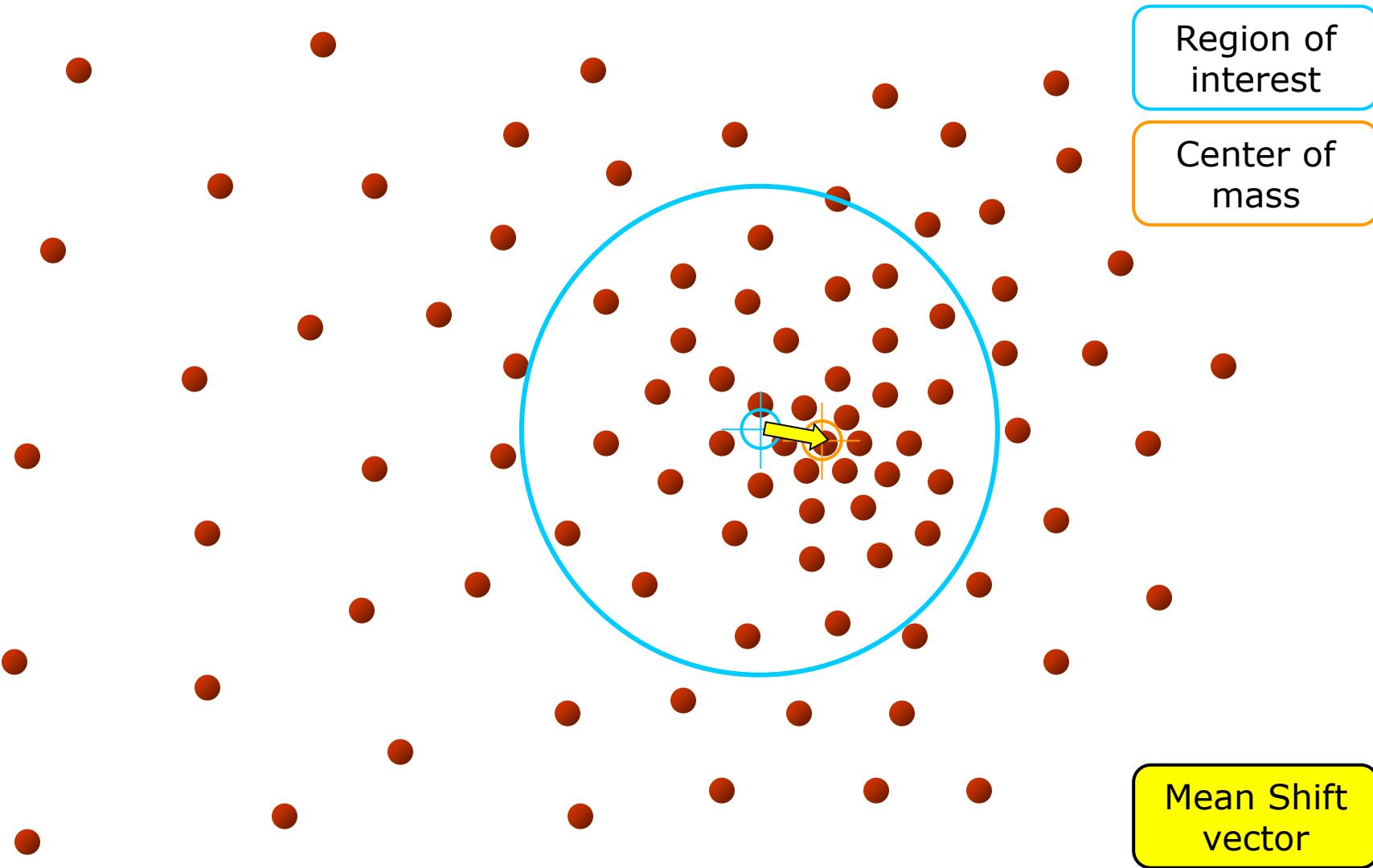
Objective : Find the densest region
Distribution of identical billiard balls

Slide by Y. Ukrainitz & B. Sarel



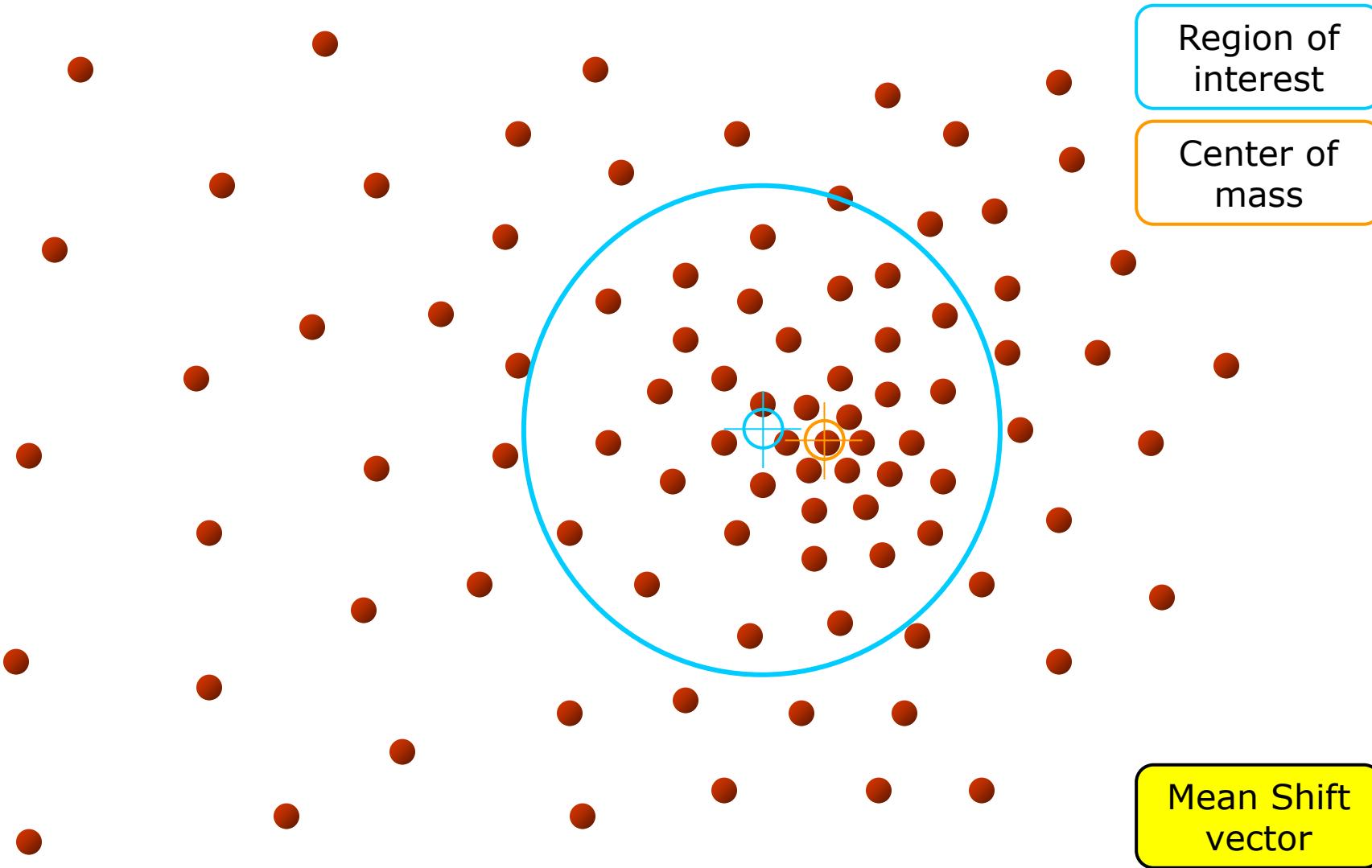
Objective : Find the densest region
Distribution of identical billiard balls

Slide by Y. Ukrainitz & B. Sarel



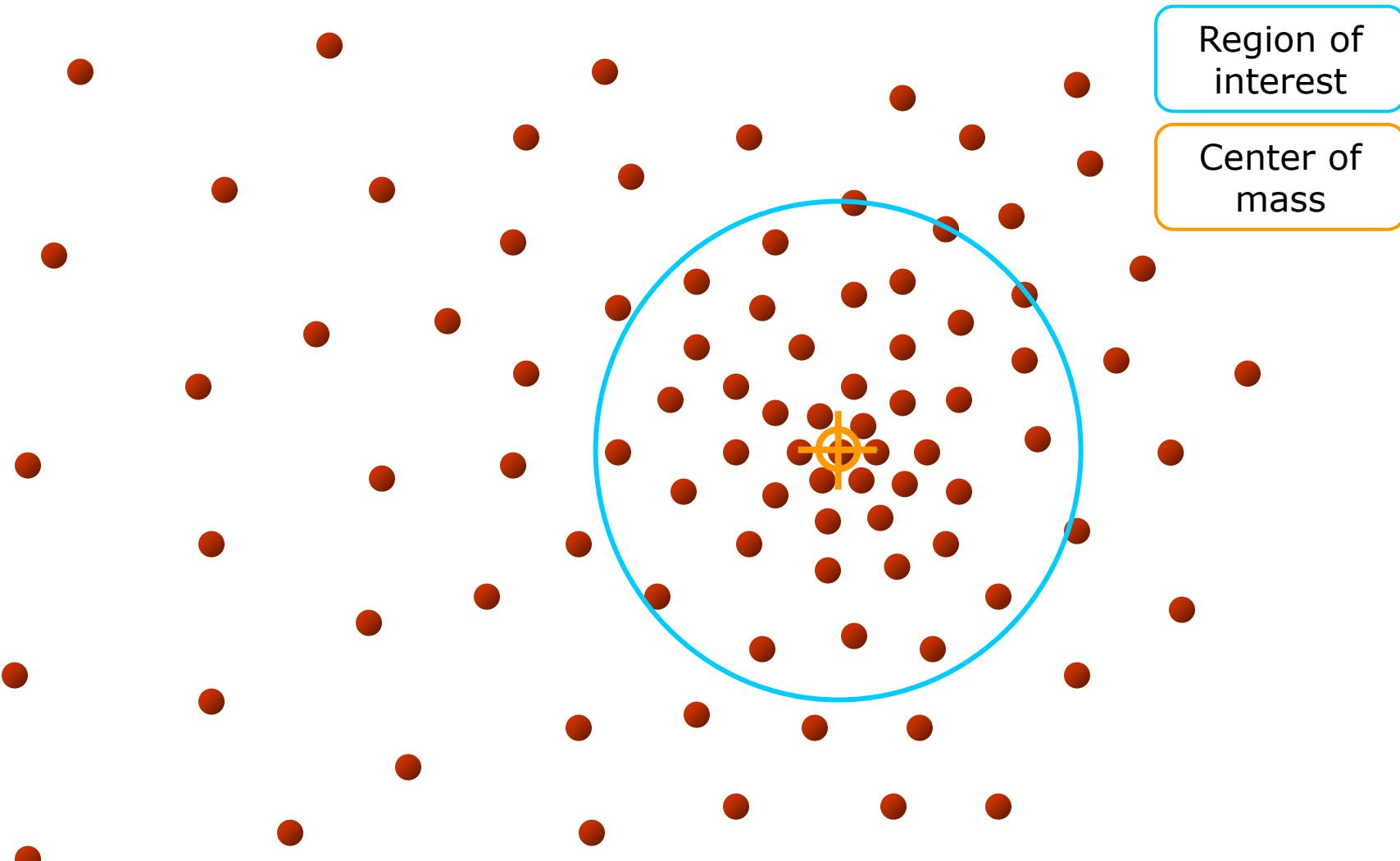
Objective : Find the densest region
Distribution of identical billiard balls

Slide by Y. Ukrainitz & B. Sarel



Objective : Find the densest region
Distribution of identical billiard balls

Slide by Y. Ukrainitz & B. Sarel

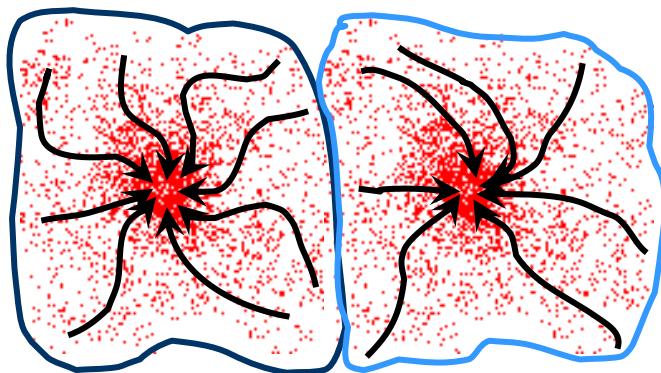


Objective : Find the densest region
Distribution of identical billiard balls

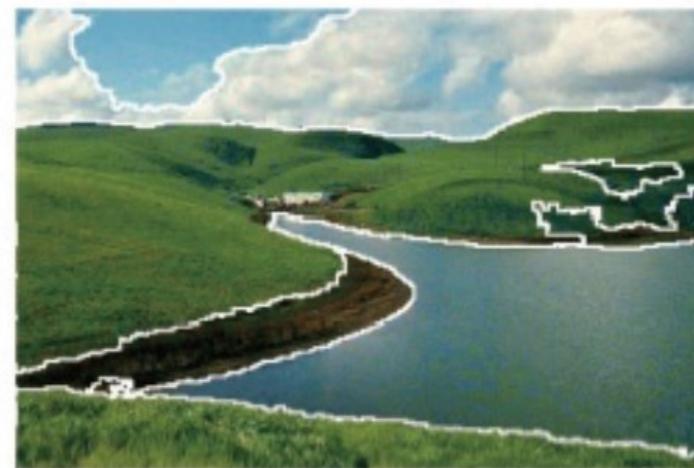
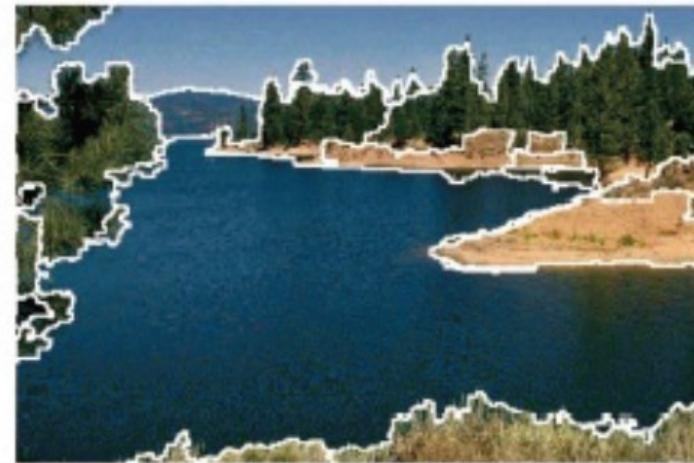
Slide by Y. Ukrainitz & B. Sarel

Mean Shift Clustering

- Cluster: all data points in the attraction basin of a mode
- Attraction basin: the region for which all trajectories lead to the same mode.



Mean Shift Segmentation Results



Mean Shift Strengths & Weaknesses

Strengths

- Generic technique
- Does not assume any prior shape (e.g. elliptical) on data clusters
- Can handle arbitrary feature spaces
- Only ONE parameter to choose
- h (window size) has a physical meaning, unlike K-Means
- Robust to outliers

Weaknesses

- The window size (bandwidth selection) is not trivial
- Does not scale well with dimension of feature space

Summary

- In this lecture, we learned
 - Segmentation
 - Thresholding
 - Edge Detection
 - Region-based Segmentation
 - Clustering
 - K-means
 - Mean Shift

Resources

- Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129-136, 1982
- D. Comaniciu, and P. Meer, "Mean Shift: A Robust Approach Toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002.
- Slides sources
 - Fei-Fei Li
(http://vision.stanford.edu/teaching/cs131_fall1617/lectures/lecture13_kmeans_mean_shift_cs131_2016)
 - Sergei Vassilvitskii and David Arthur
(<http://theory.stanford.edu/~sergei/slides/BATS-Means.pdf>)
 - Yaron Ukrainitz & Bernard Sarel
(http://www.wisdom.weizmann.ac.il/~vision/courses/2004_2/files/mean_shift/mean_shift.ppt)
 - <http://robot-develop.org/wp-content/uploads/2012/03/seg3.pdf>

Image Processing & Pattern Recognition

Image Classification with Convolutional Neural Networks

Dr. Zia Ud Din

Outline

- Introduction to Image Classification
- Neural Networks
- Convolutional Neural Networks
- Datasets for Image Classification
- Practical Example with CIFAR Dataset
- Further Resources

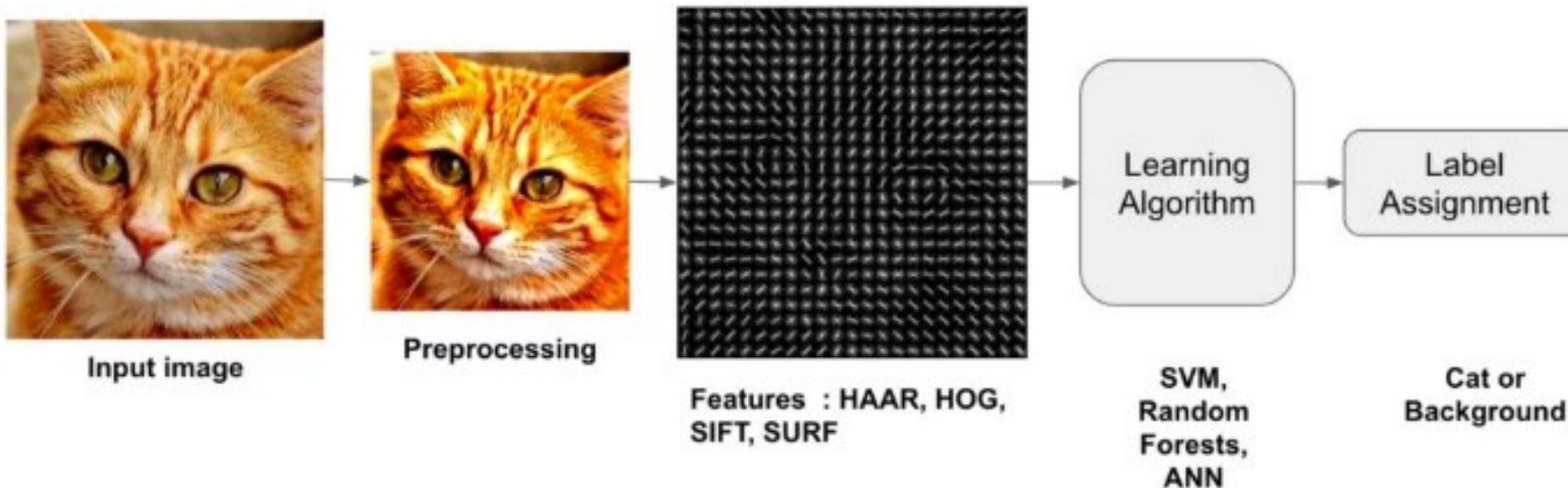
Outline

- **Introduction to Image Classification**
- Neural Networks
- Convolutional Neural Networks
- Datasets for Image Classification
- Practical Example with CIFAR Dataset
- Further Resources

Introduction to Image Classification

- **Image Classification** is the task of assigning an input image one label from a fixed set of categories.
- Many Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.
- Data-driven approaches for Image Classification include the following steps:
 1. Collect a dataset of images and labels
 2. Use Machine Learning to train a classifier
 3. Evaluate the classifier on new images

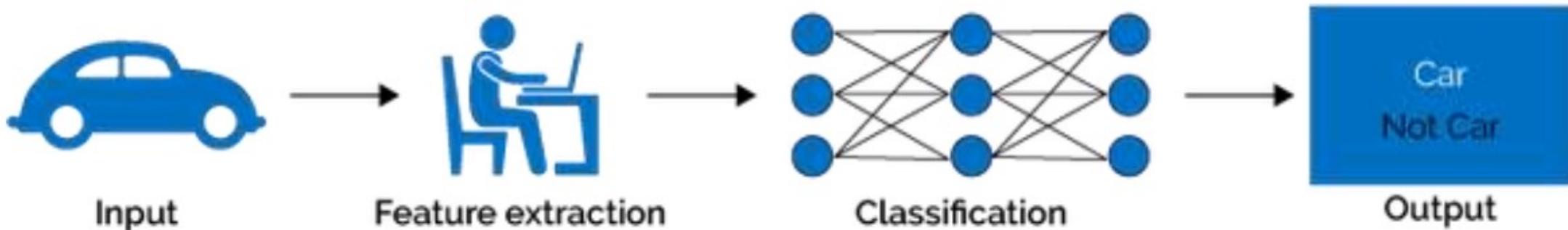
Anatomy of Traditional Image Classifiers



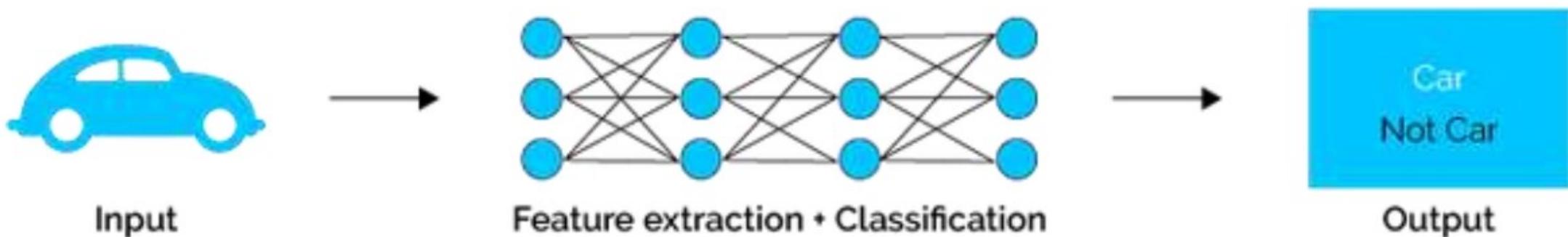
(Source)

- Deep Learning based algorithms bypass the feature extraction step completely.

Shallow learning



Deep learning



Outline

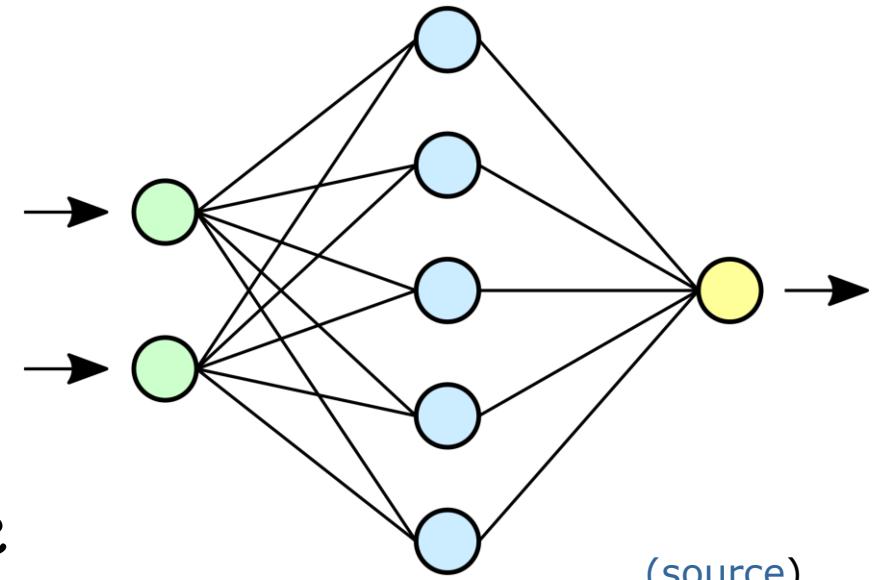
- Introduction to Image Classification
- **Neural Networks**
- Convolutional Neural Networks
- Datasets for Image Classification
- Practical Example with CIFAR Dataset
- Further Resources

Neural Networks

- A **Neural Network** is a computational model inspired by the way human brains operate. It consists of interconnected units or nodes called **neurons**, which process information.
- Neural Networks are a part of the broader field of machine learning and are particularly effective at recognizing patterns and making predictions based on complex data.
- These networks 'learn' from large amounts of data by adjusting the connections between neurons, mimicking the learning process in the human brain.

Neural Networks...

- Neural Networks are structured in layers: **the input layer**, **hidden layers**, and **the output layer**.
- The input layer receives the initial data, while the output layer produces the final prediction or classification.
- Between them are hidden layers, where the majority of processing occurs. The complexity and capacity of a neural network depend largely on the number and size of these hidden layers.



[\(source\)](#)

Perceptron

- A **perceptron** is a single-layer neural network and can be thought of as a single neuron. It is the basic unit for more complex neural networks.
- A perceptron takes a vector of real-valued inputs, calculates a **linear combination** of these inputs, then outputs “1” if the result is greater than some threshold and “-1” otherwise.

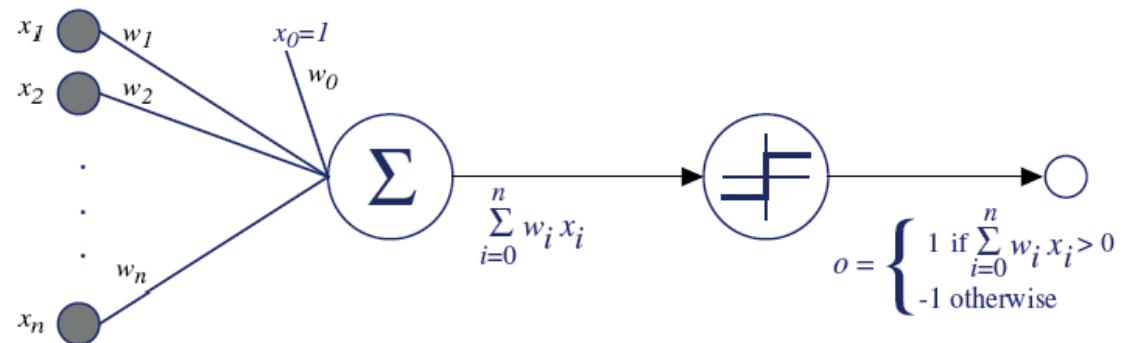
Understanding Perceptrons

- Given inputs $x_1 \dots x_n$, the output $o(x_1 \dots x_n)$ computed by the perceptron is:

$$o(x_1, \dots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \dots + w_n x_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

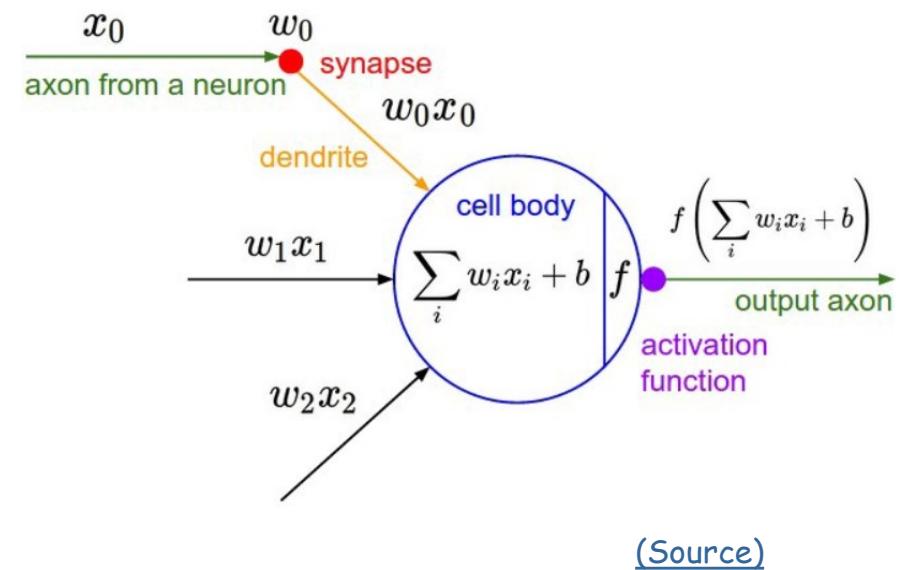
OR

$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise} \end{cases}$$



Activation Functions

- The activation function is used as a decision-making body at the output of a neuron.
- The neuron learns Linear or Non-linear decision boundaries based on the activation function.
- It also has a normalizing effect on the neuron output which prevents the output of neurons after several layers to become very large, due to the cascading effect.

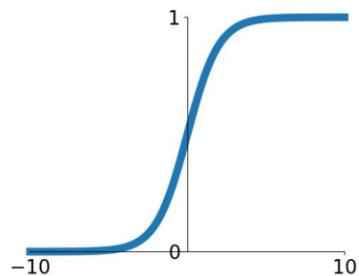


(Source)

Activation Functions...

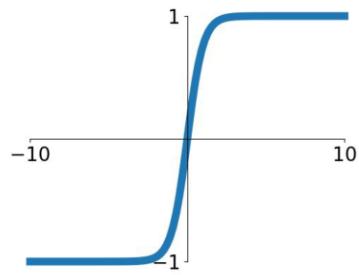
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



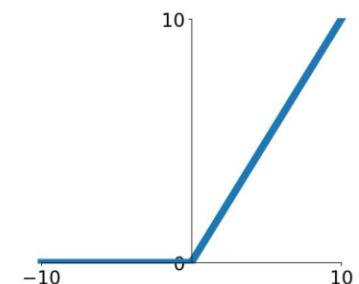
tanh

$$\tanh(x)$$



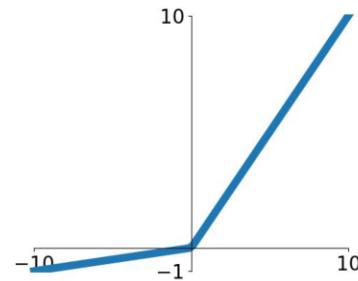
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

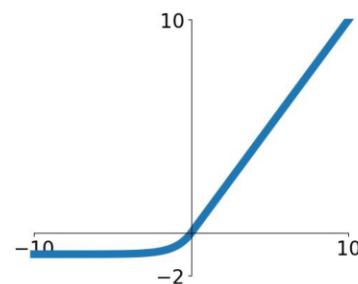


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Understanding Weights and Biases

- Weights and biases are fundamental to how neural networks learn. They are parameters that the network adjusts through the learning process.
- Weights control the strength of the connection between neurons. Biases allow the network to shift the activation function to fit the data better.
- During training, the network adjusts these weights and biases to minimize the difference between its predictions and the actual data, a process known as 'training the model'.

Training a Neural Network

- Training a neural network involves feeding it data and allowing it to adjust its weights and biases to minimize prediction errors.
- This process uses algorithms like **backpropagation** and **optimization techniques** like gradient descent to find the best values for weights and biases.
- As the network trains, it '**learns**' from the data, improving its ability to make predictions or classify data accurately.

Outline

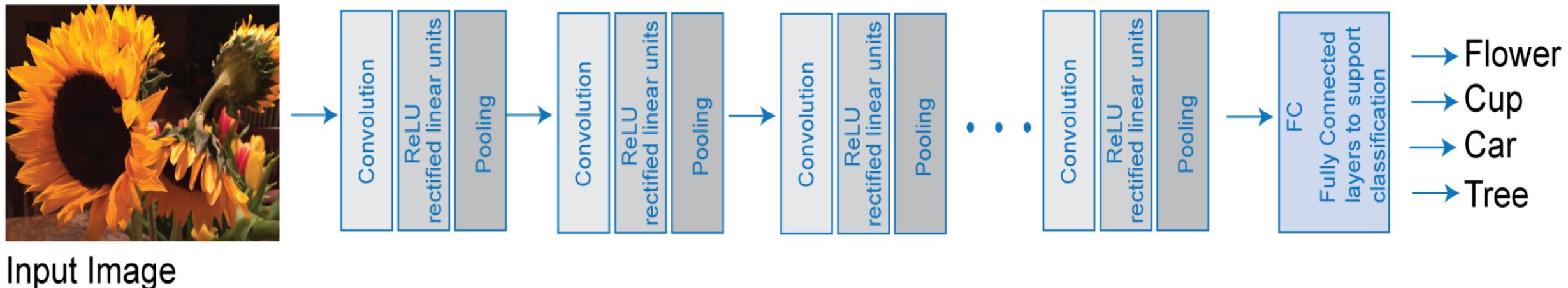
- Introduction to Image Classification
- Neural Networks
- **Convolutional Neural Networks**
- Datasets for Image Classification
- Practical Example with CIFAR Dataset
- Further Resources

Why CNNs?

- Regular Neural Networks face several challenges when dealing with images, leading to the adoption of Convolutional Neural Networks (CNNs) for image processing tasks.
 - High Dimensionality of Images
 - A modest 256x256 color image has 196,608 individual values (256x256 pixels, each with Red, Green, and Blue channels). In a regular NN, each pixel value would need to be connected to neurons in the next layer, resulting in an enormous number of parameters.
 - Loss of Spatial Hierarchy
 - When an image is fed into a regular NN, it's typically converted into a one-dimensional vector, losing the spatial relationships between pixels. This spatial information is crucial for understanding and identifying features in images.

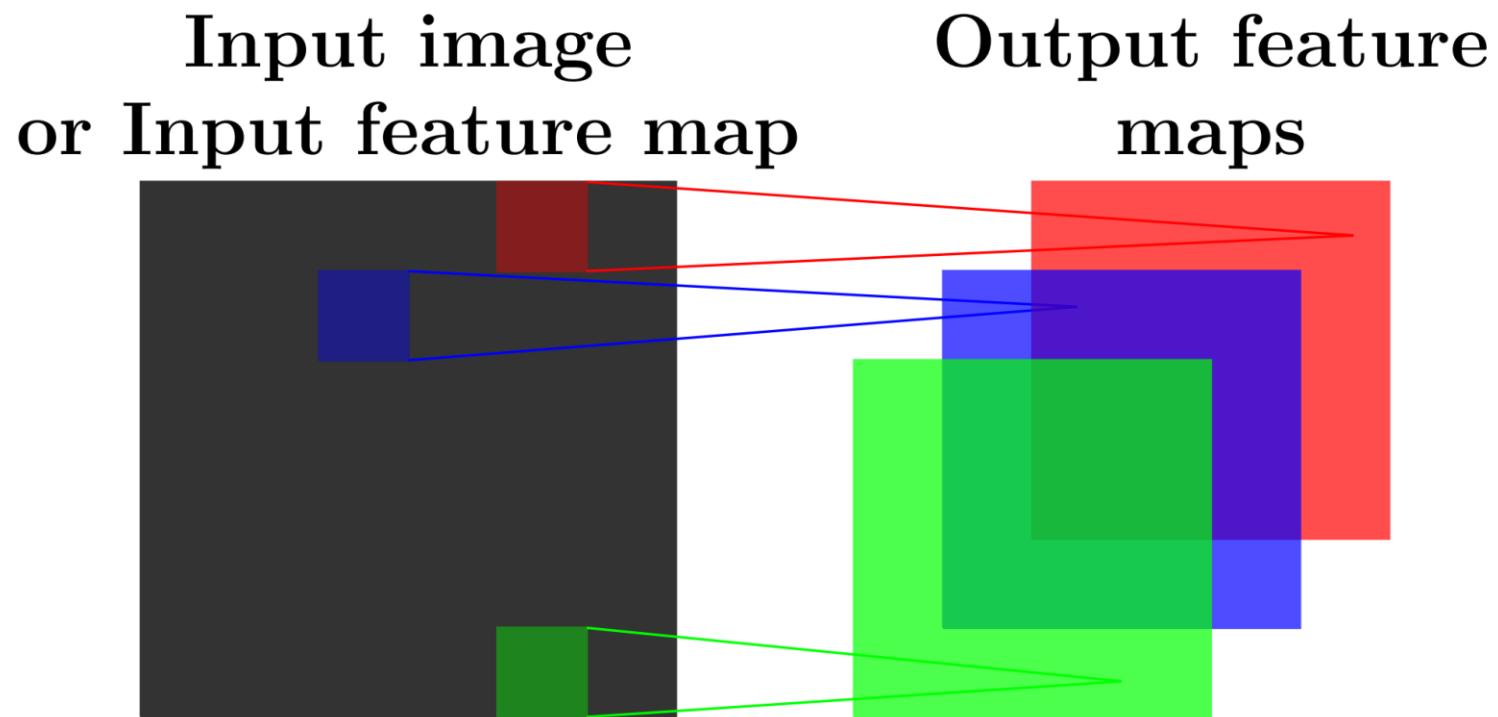
CNN

- Convolutional Neural Network (CNN) is a multi layer neural network
- Convolutional Neural Network is comprised of one or more convolutional layers (often with a pooling layers and then followed by one or more fully connected layers



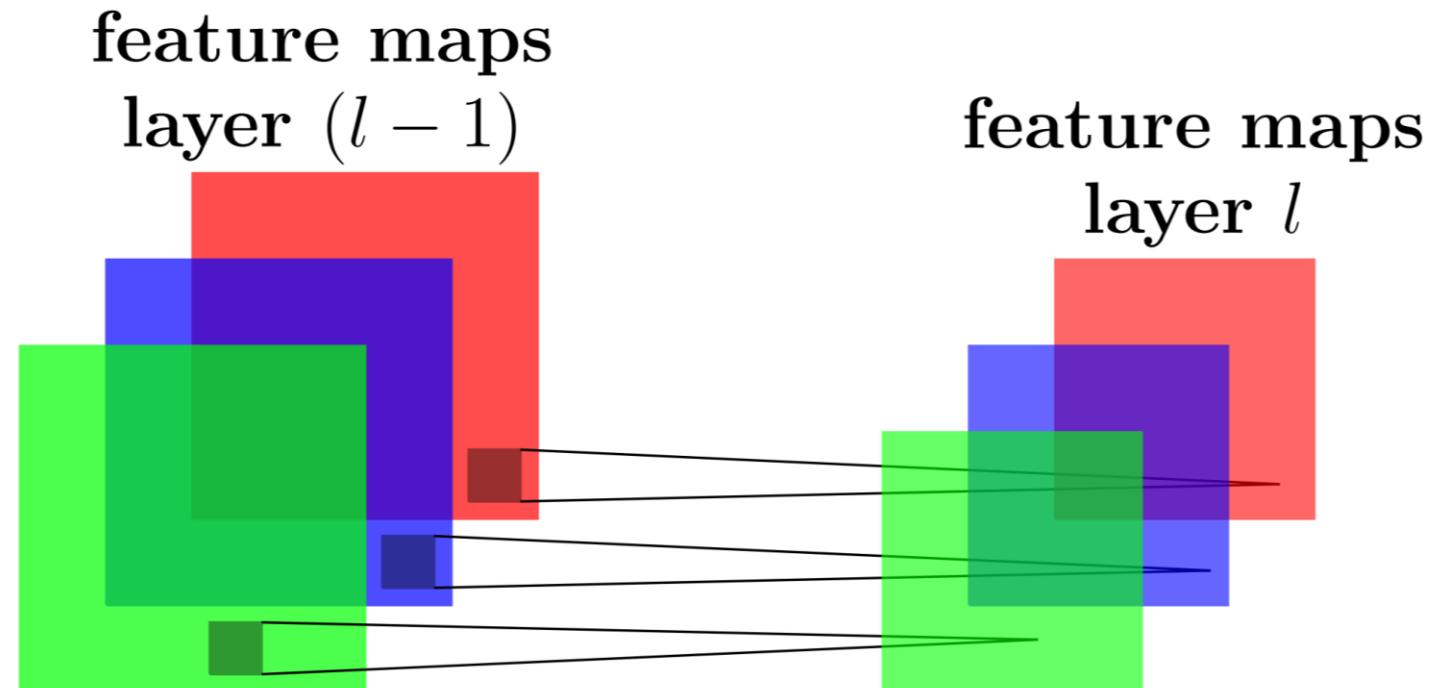
Convolutional Layer

- Convolutional layer acts as a feature extractor that extracts features of the inputs such as edges, corners endpoints



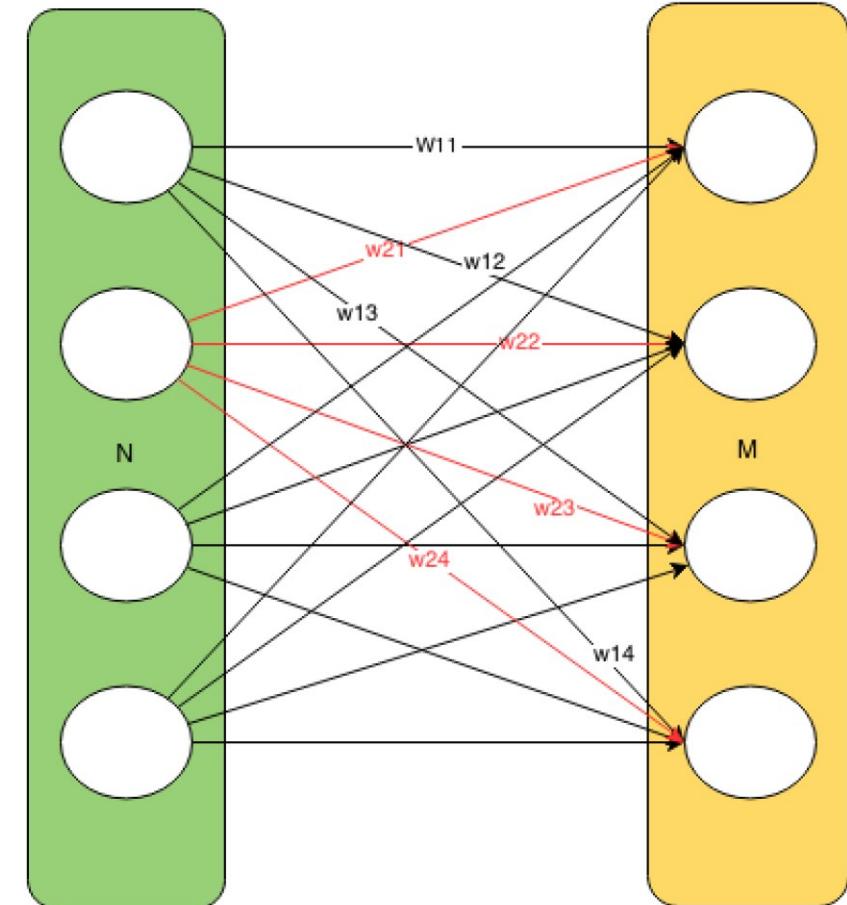
Pooling Layer

- Pooling reduces the resolution of the input volume for the subsequent layers, so the network has fewer parameters to learn and less computational work to perform.



Fully Connected Layer

- Fully connected layer have full connections to all activations in the previous layer.
- Fully connect layer act as classifier.



Convolutional Layer Examples

$$O = \frac{N-F+2P}{S} + 1$$

Where O = output image size, N = input image size,
 P = padding, F = filter size, S = stride

- Conv 3×3 with stride= 1 , padding= 0

1	2	1	1	1	1
1	1	5	3	9	1
2	4	4	7	5	3
3	6	7	5	6	2
1	6	5	3	1	2
2	3	1	1	1	1

6x6 Image



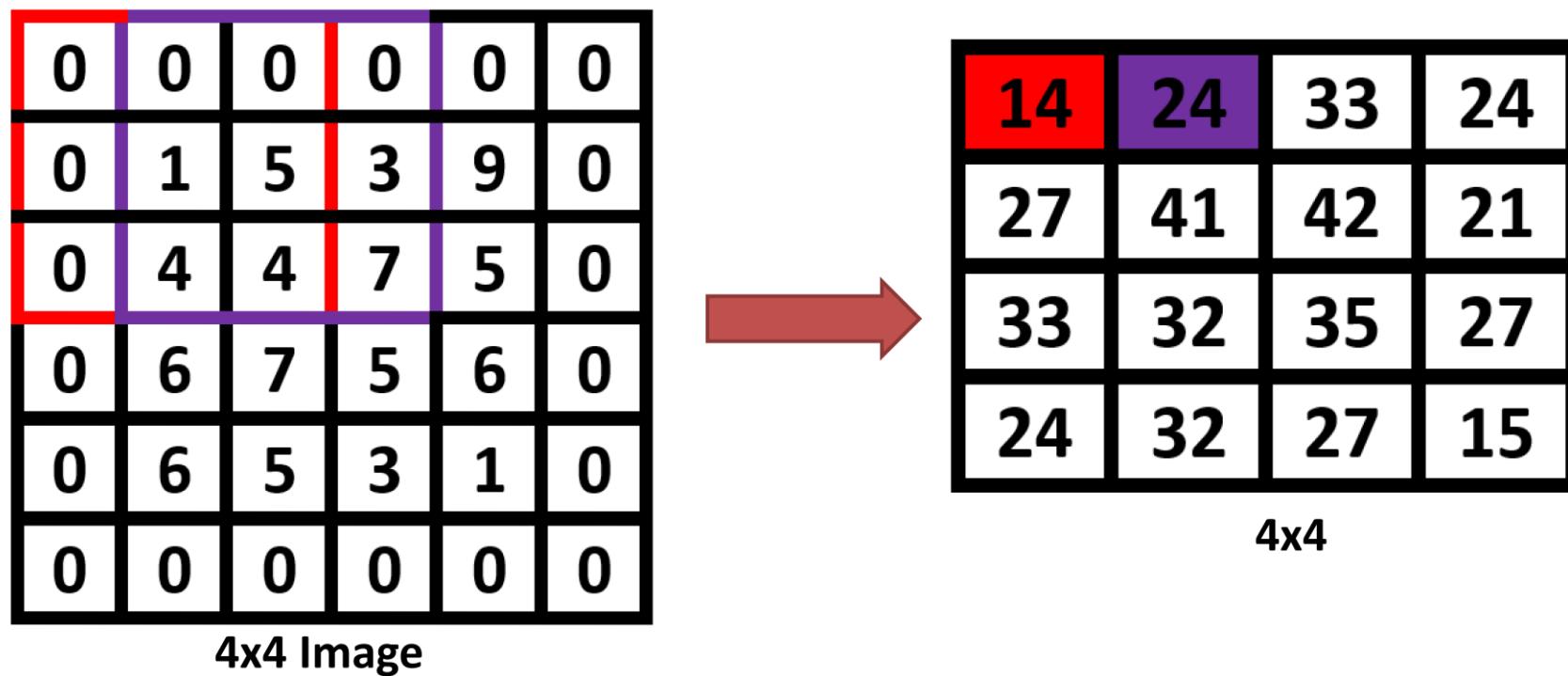
21	28	36	31
33	42	51	41
38	47	43	34
34	37	30	22

4x4

Convolutional Layer Examples...

- Conv 3×3 with stride= 1, padding= 1

$$O = \frac{N-F+2P}{S} + 1$$



Convolutional Layer Examples...

- Conv 3×3 with stride= 2 , padding=0

$$O = \frac{P-F+2P}{S} + 1$$

1	2	1	1	1	1	1
1	1	5	3	9	1	1
2	4	4	7	5	3	1
3	6	7	5	6	2	2
1	6	5	3	1	2	1
2	3	1	1	1	1	1
1	1	1	3	2	2	1

7x7 Image



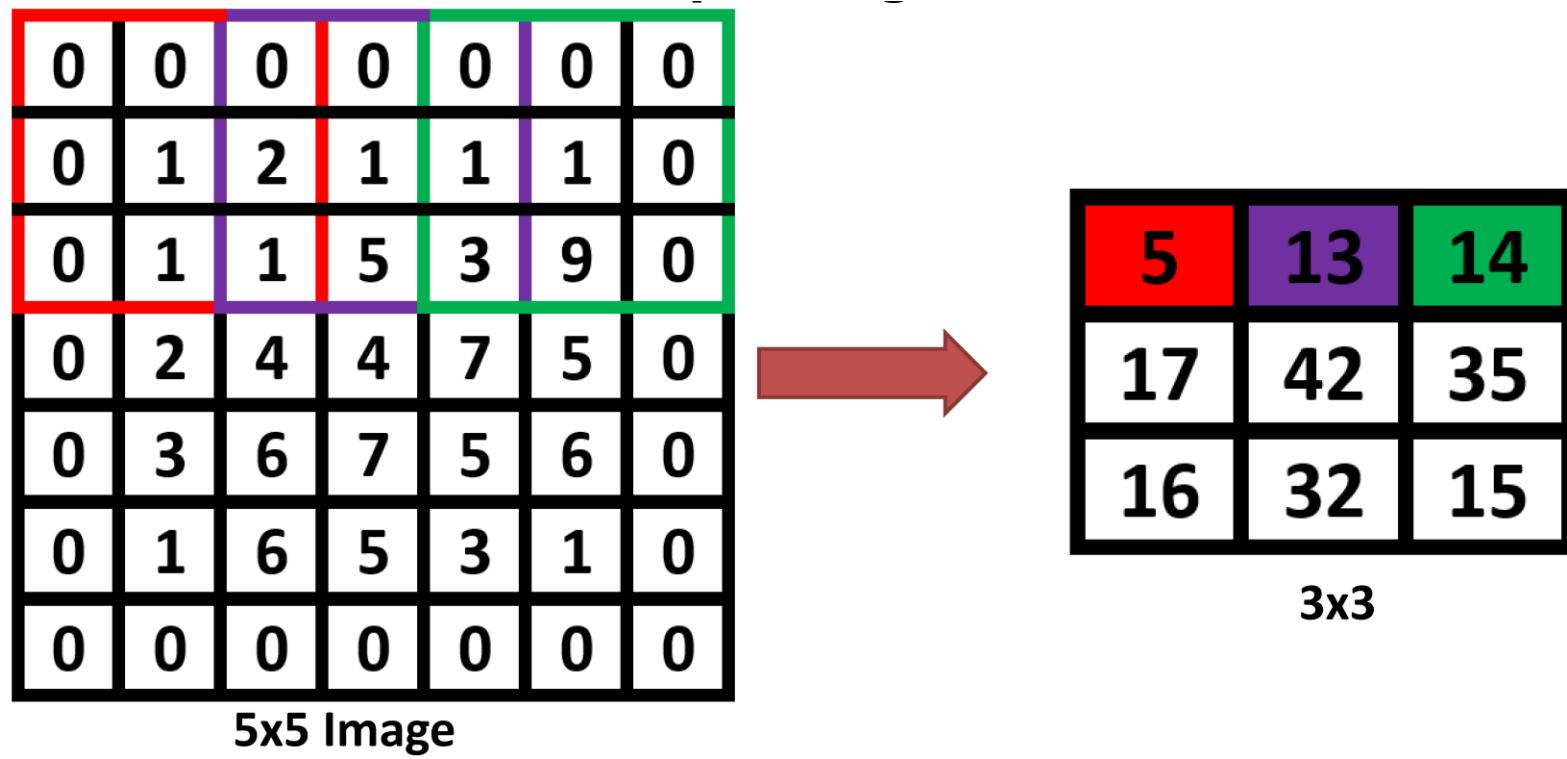
21		
14	36	23
38	43	23
21	18	12

3x3

Convolutional Layer Examples...

- Conv 3×3 with **stride= 2 , padding=1**

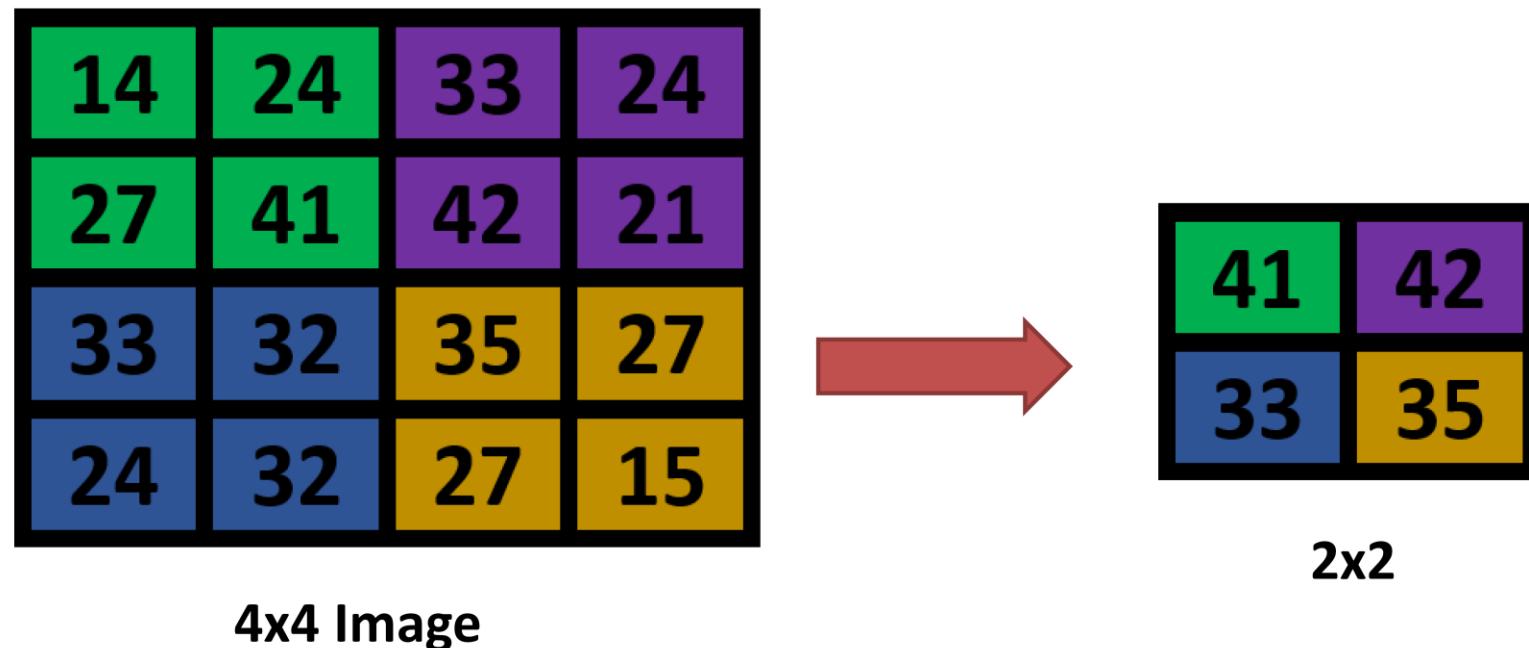
$$O = \frac{N-F+2P}{S} + 1$$



MaxPooling Layer Examples...

- MaxPooling 2×2 with stride= 2

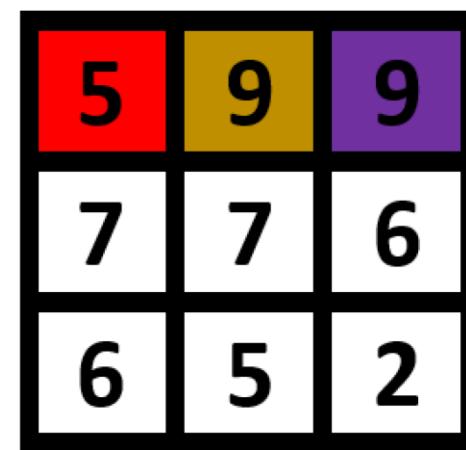
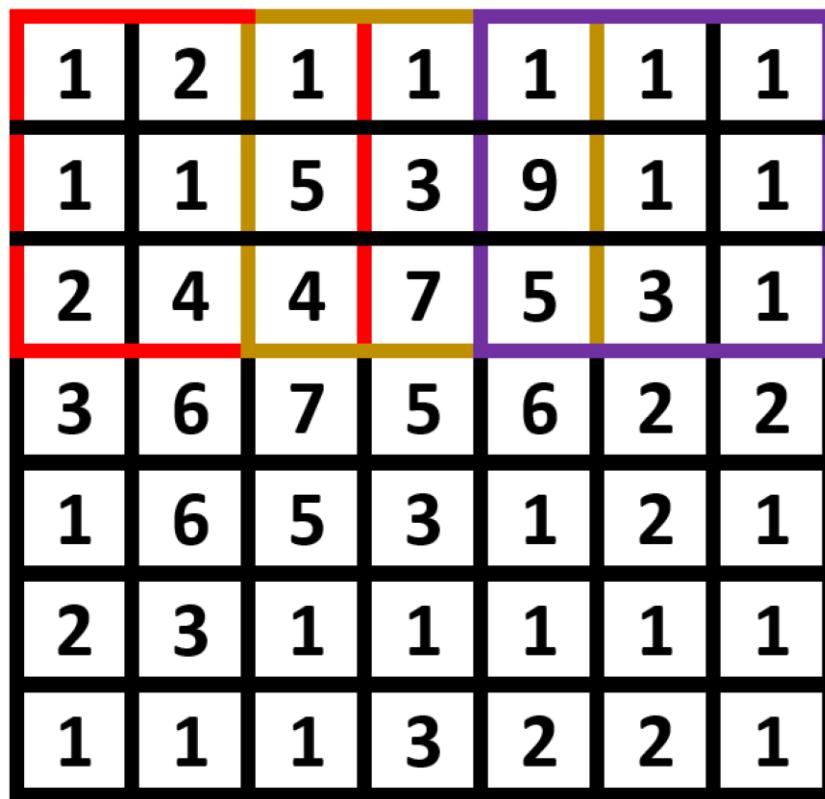
$$O = \frac{P-F+2P}{S} + 1$$



MaxPooling Layer Examples...

- MaxPooling 3×3 with stride= 2

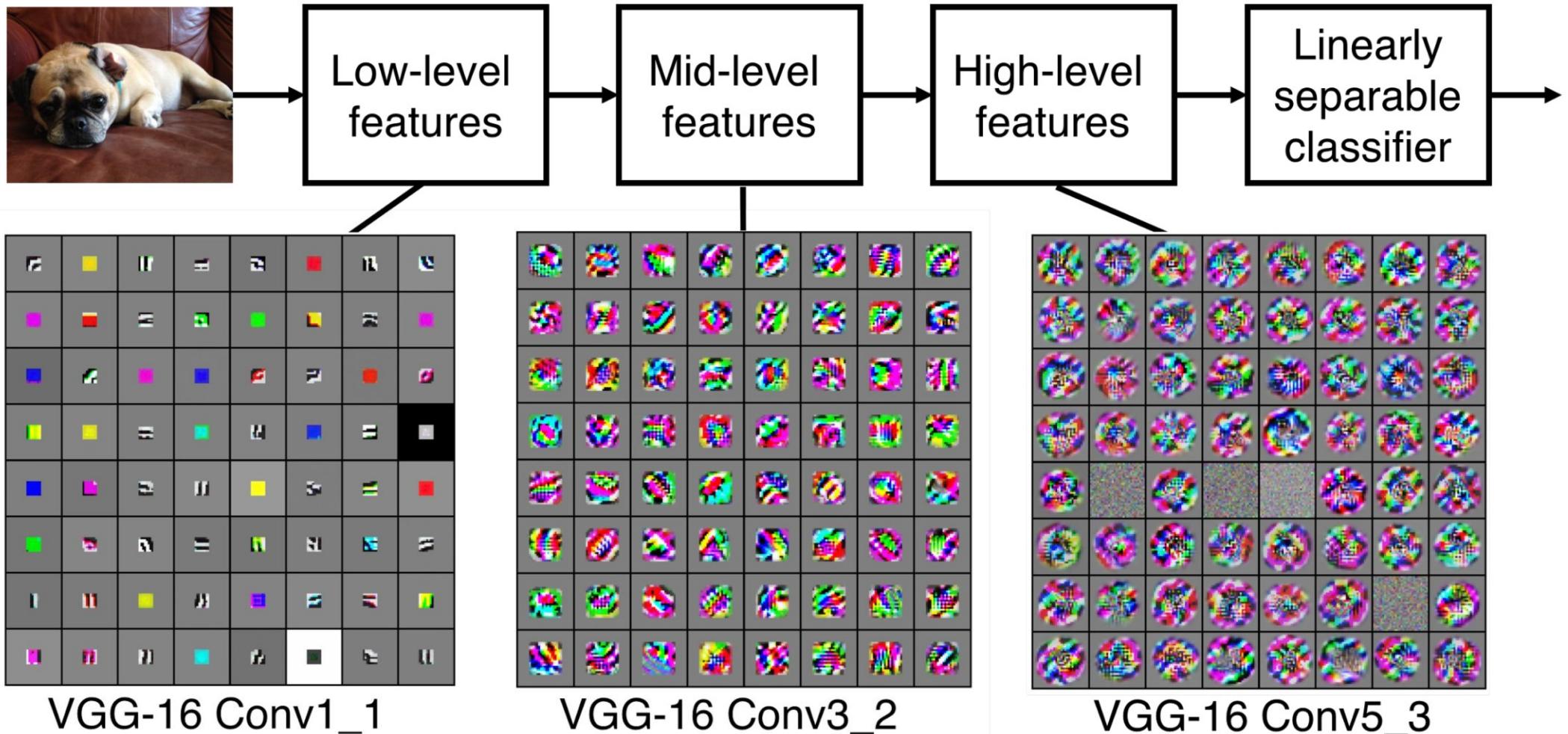
$$O = \frac{P-F+2P}{S} + 1$$



Layered Feature Learning

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



CNN Architectures

- To understand CNN architectures, we will use [AlexNet](#) as a case study.

ILSVRC

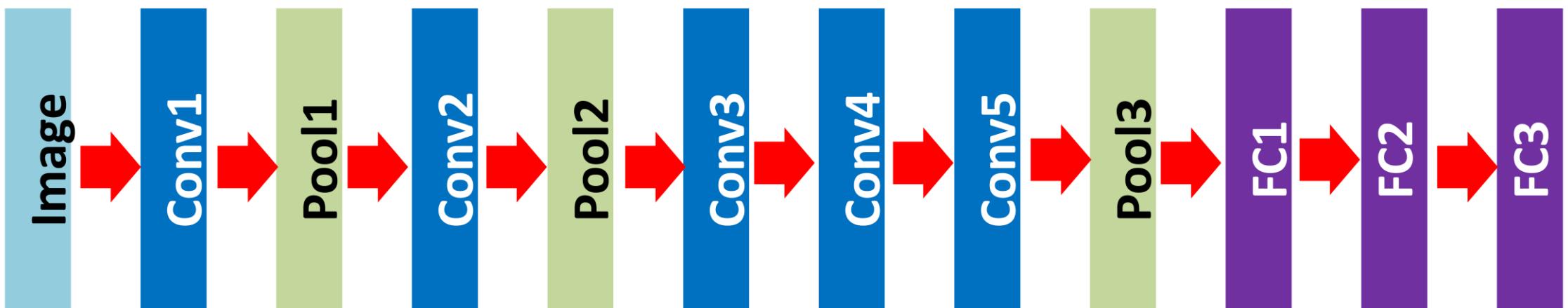
- **ImageNet Large Scale Visual Recognition Challenge** is image classification challenge to create model that can correctly classify an input image into **1,000 separate object categories**
- Models are trained on **1.2 million training images** with another 50,000 images for validation and 150,000 images for testing

AlexNet (2012)

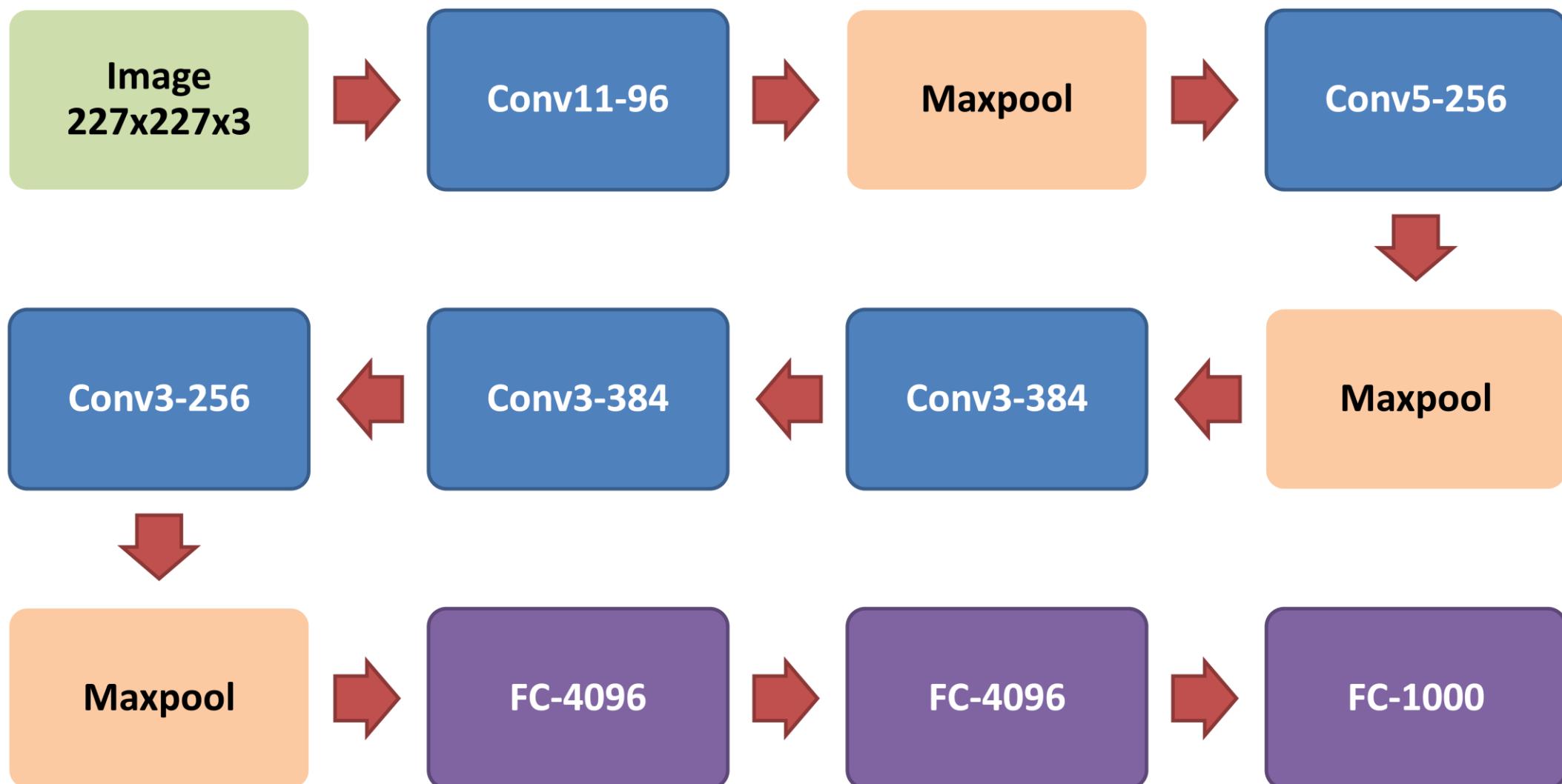
- AlexNet achieve on ILSVRC 2012 competition 15.3% Top-5 error rate compared to 26.2% achieved by the second best entry.
- AlexNet using batch stochastic gradient descent on training, with specific values for momentum and weight decay.
- AlexNet implement dropout layers in order to combat the problem of overfitting to the training data.

AlexNet Architecture

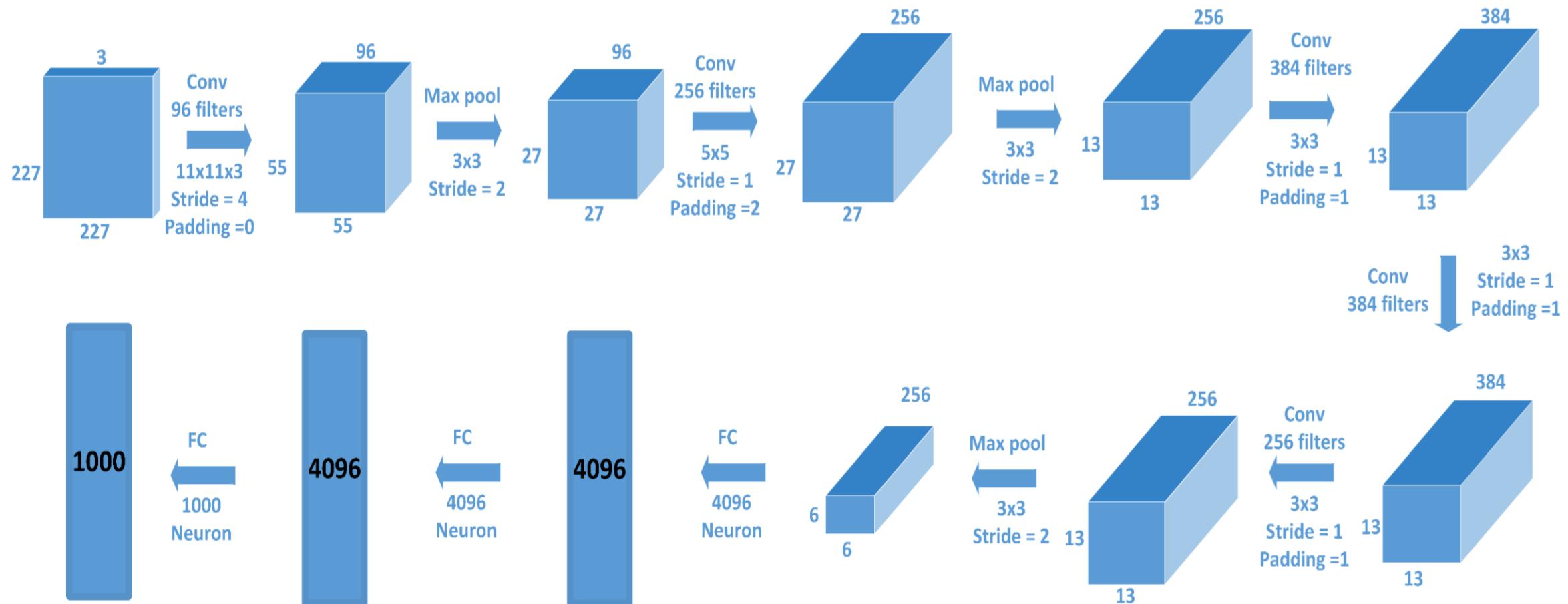
- AlexNet has 8 layers (5 conv. + 3 fc) without counting pooling layers.
- AlexNet use ReLU for the nonlinearity functions.
- AlexNet trained on two GTX 580 GPUs for five to six days.



AlexNet Architecture...

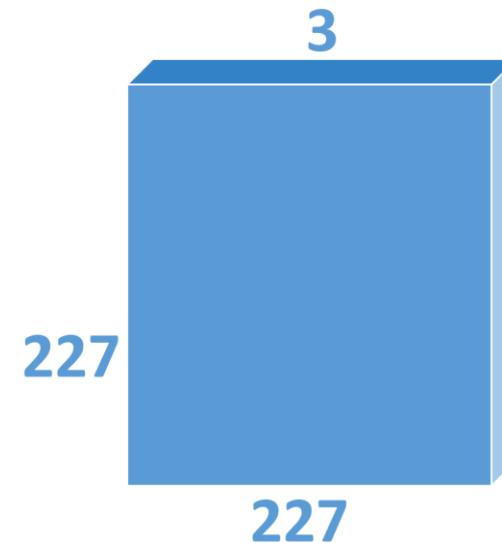


AlexNet Architecture...



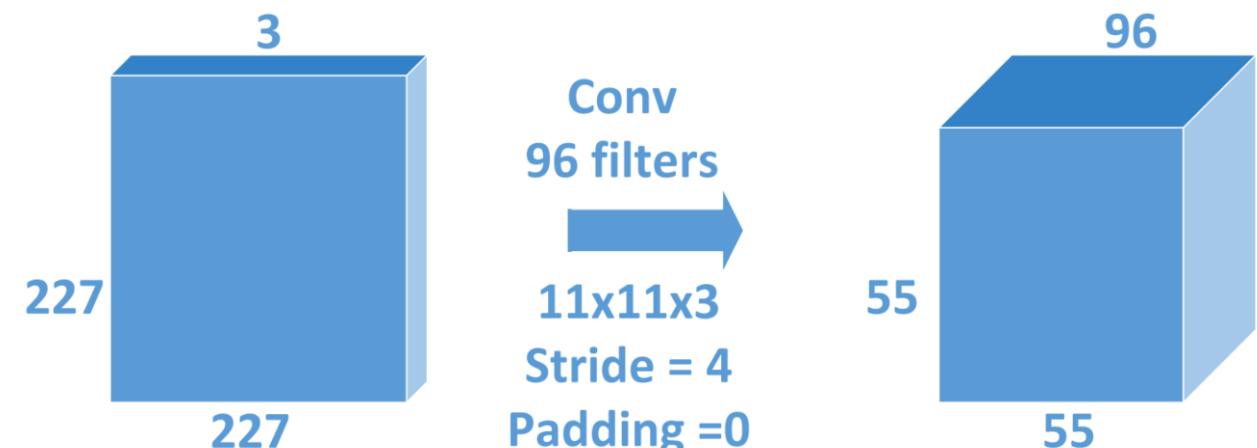
AlexNet Architecture...

- Layer 0: Input image
- Size: $227 \times 227 \times 3$
- Memory: $227 \times 227 \times 3$



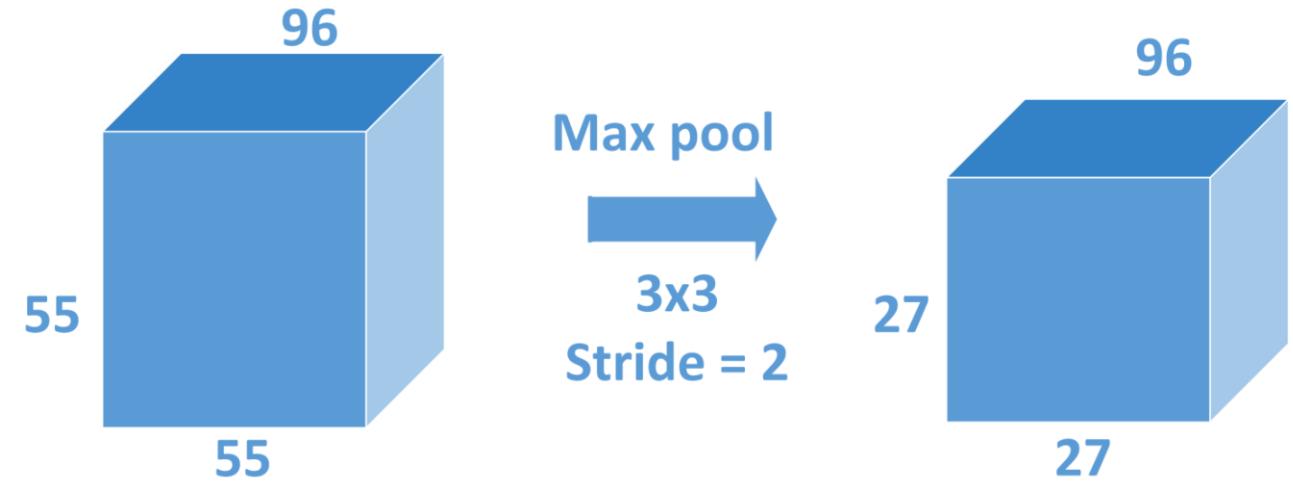
AlexNet Architecture...

- Layer 0: $227 \times 227 \times 3$
- Layer 1: Convolution with 96 filters, size 11×11 , stride=4, padding 0
- Outcome Size: $55 \times 55 \times 96$
- $(227-11)/4+1=55$ is size of outcome
- Memory: $55 \times 55 \times 96 \times 3$ (because of ReLU & LRN (Local Response Normalization))
- Weights (parameters): $11 \times 11 \times 3 \times 96$



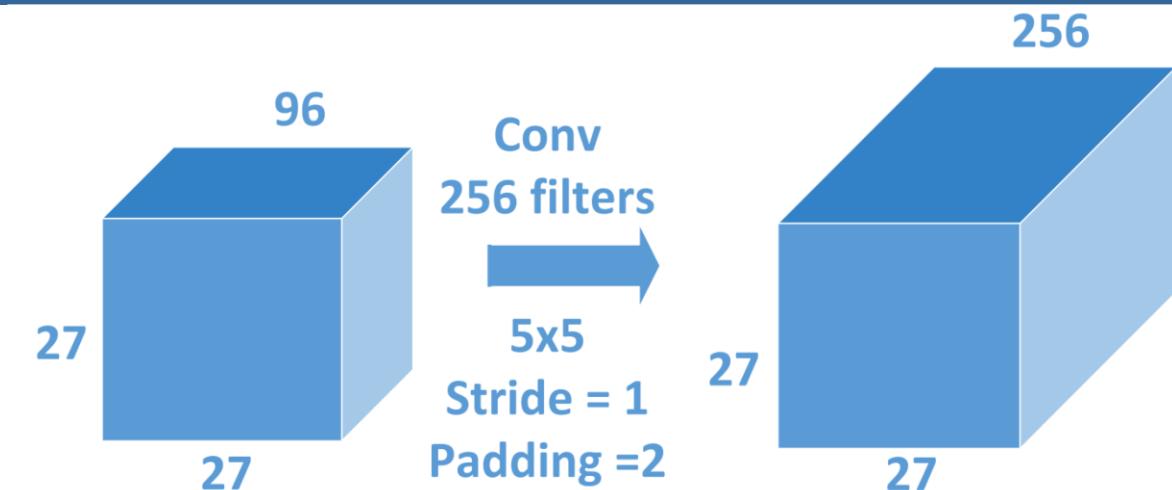
AlexNet Architecture

- Layer 1: $55 \times 55 \times 96$
- Layer 2: Max Pooling with 3×3 filter, stride 2
- Outcome Size: $27 \times 27 \times 96$
- $(55-3)/2+1=27$ is size of outcome
- Memory: $27 \times 27 \times 96$



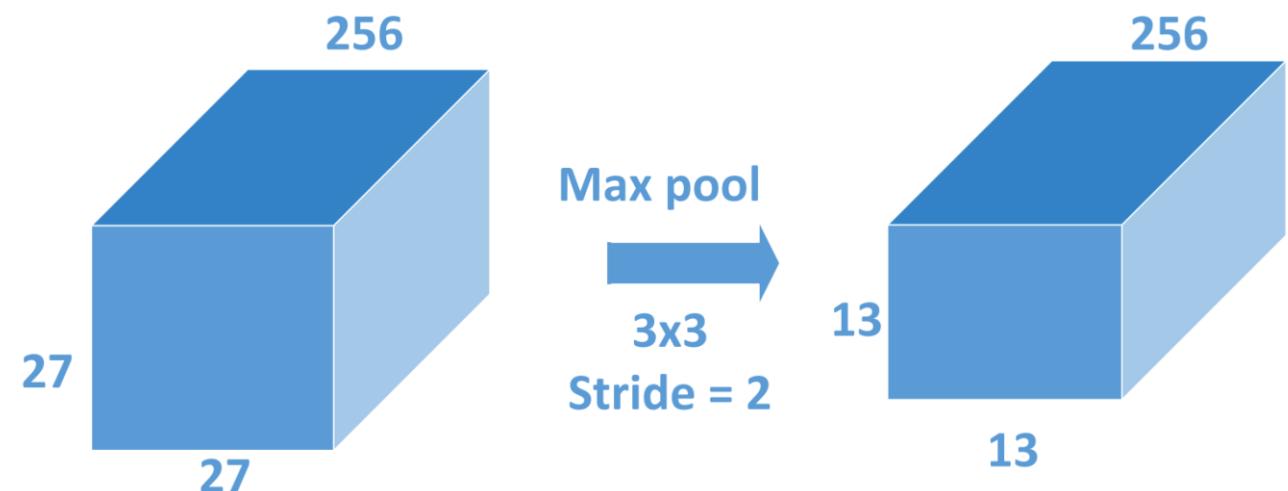
AlexNet Architecture...

- Layer 2: $27 \times 27 \times 96$
- Layer 3: Convolution with 256 filters, size 5×5 , stride 1, padding 2
- Outcome Size: $27 \times 27 \times 256$
- Original size is restored because of padding
- Memory: $27 \times 27 \times 256 \times 3$ (because of ReLU and LRN)
- Weights: $5 \times 5 \times 96 \times 256$



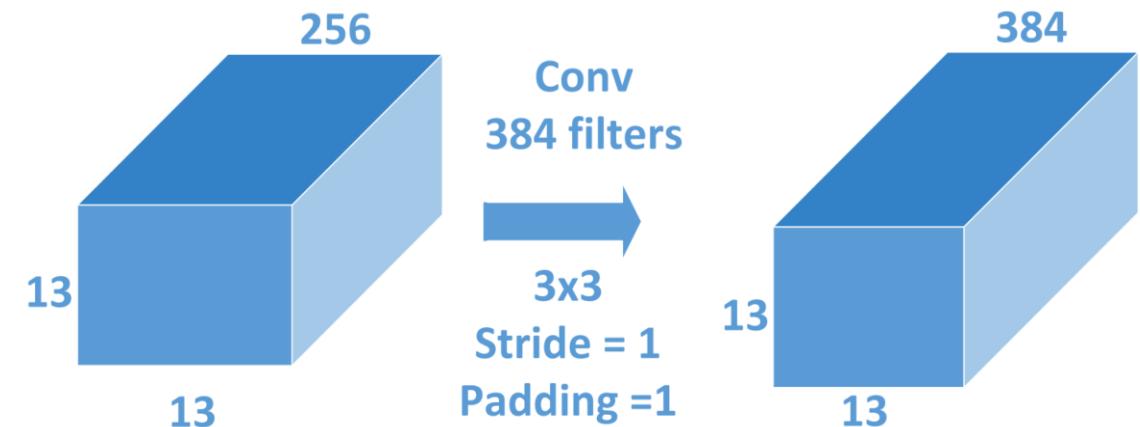
AlexNet Architecture

- Layer 3: $27 \times 27 \times 256$
- Layer 4: Max Pooling with 3×3 filter, stride 2
- Outcome Size: $13 \times 13 \times 256$
- $(27-3)/2+1=13$ is size of outcome
- Memory: $13 \times 13 \times 256$



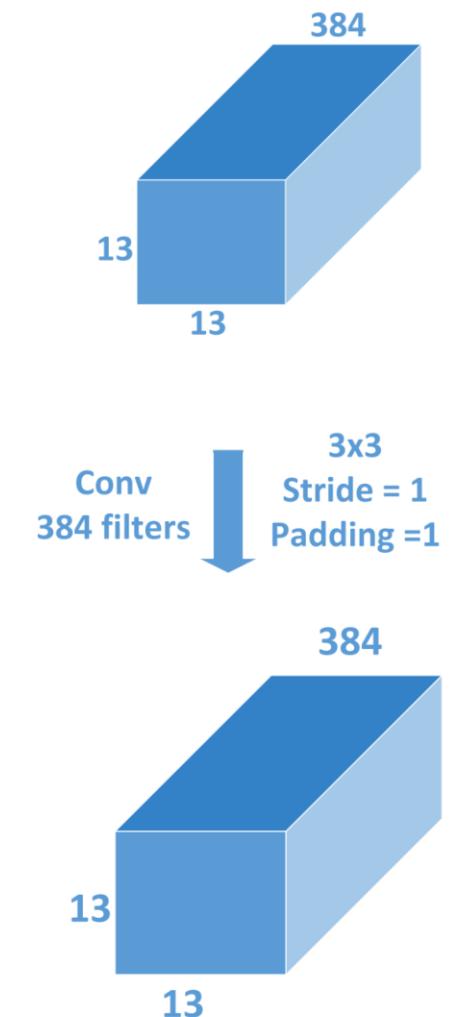
AlexNet Architecture...

- Layer 4: $13 \times 13 \times 256$
- Layer 5: Convolution with 384 filters, size 3×3 stride , padding 1
- Outcome Size: $13 \times 13 \times 384$
- The original size is restored because of padding $(13=2-3)/1+1=13$
- Memory: $13 \times 13 \times 384 \times 2$ (because of ReLU)
- Weights: $3 \times 3 \times 256 \times 384$



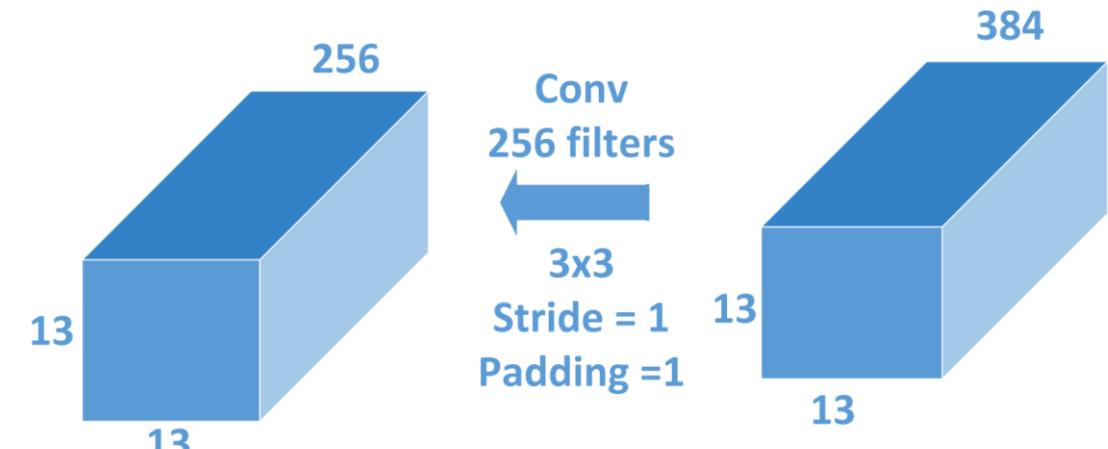
AlexNet Architecture...

- Layer 5: $13 \times 13 \times 384$
- Layer 6 Convolution with 384 filters, size 3×3 , stride 1, padding 1
- Outcome Size: $13 \times 13 \times 384$
- The original size is restored because of padding
- Memory: $13 \times 13 \times 384 \times 2$ (because of ReLU)
- Weights: $3 \times 3 \times 384 \times 384$



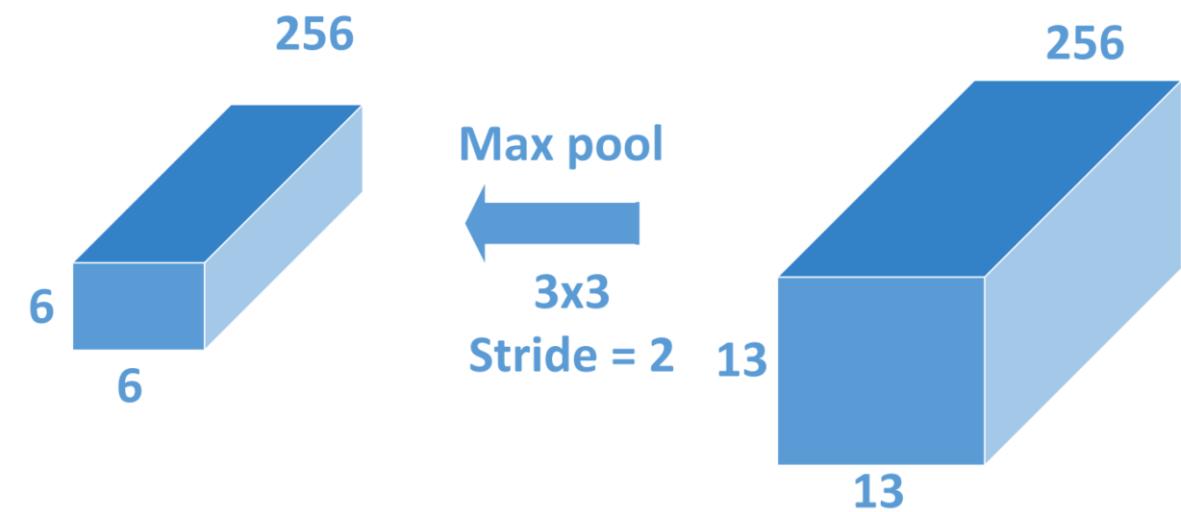
AlexNet Architecture...

- Layer 6: $13 \times 13 \times 384$
- Layer 7: Convolution with 256 filters, size 3×3 , stride 1, padding 1
- Outcome Size: $13 \times 13 \times 256$
- The original size is restored because of padding
- Memory: $13 \times 13 \times 256 \times 2$ (because of ReLU)
- Weights: $3 \times 3 \times 384 \times 256$



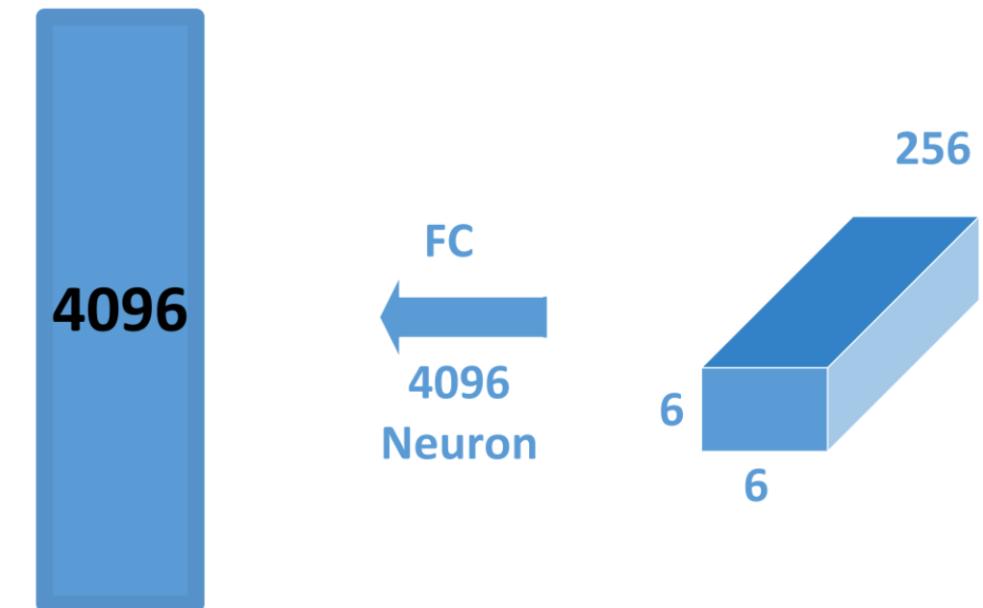
AlexNet Architecture

- Layer 7: $13 \times 13 \times 256$
- Layer 8: Max Pooling with 3×3 filter, stride 2
- Outcome Size: $6 \times 6 \times 256$
- $(13-3)/2+1=6$ is size of outcome
- Memory: $6 \times 6 \times 256$



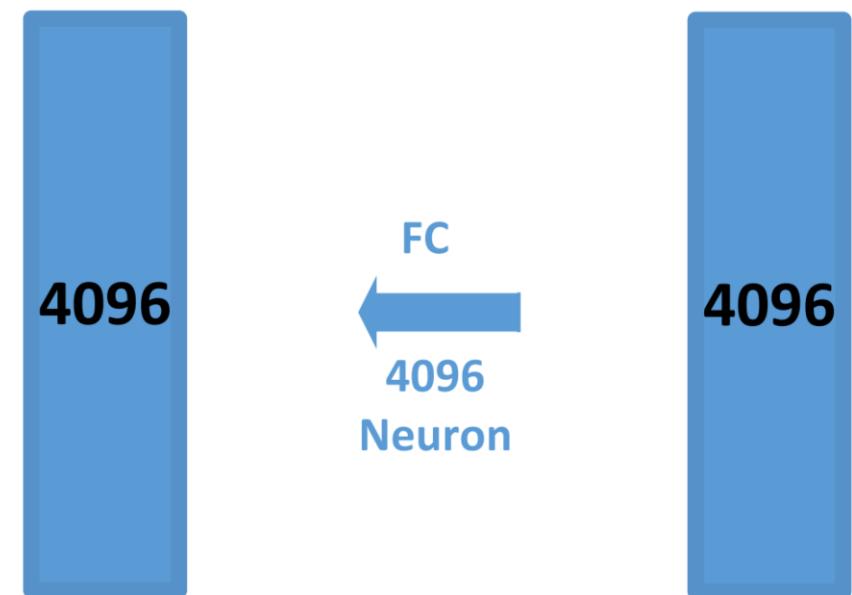
AlexNet Architecture...

- Layer 8: $6 \times 6 \times 256$ 9216 pixels are fed to FC
- Layer 9: Fully Connected with 4096 neuron
- Memory: 4096×3 (because of ReLU and Dropout)
- Weights: $4096 \times (6 \times 6 \times 256)$



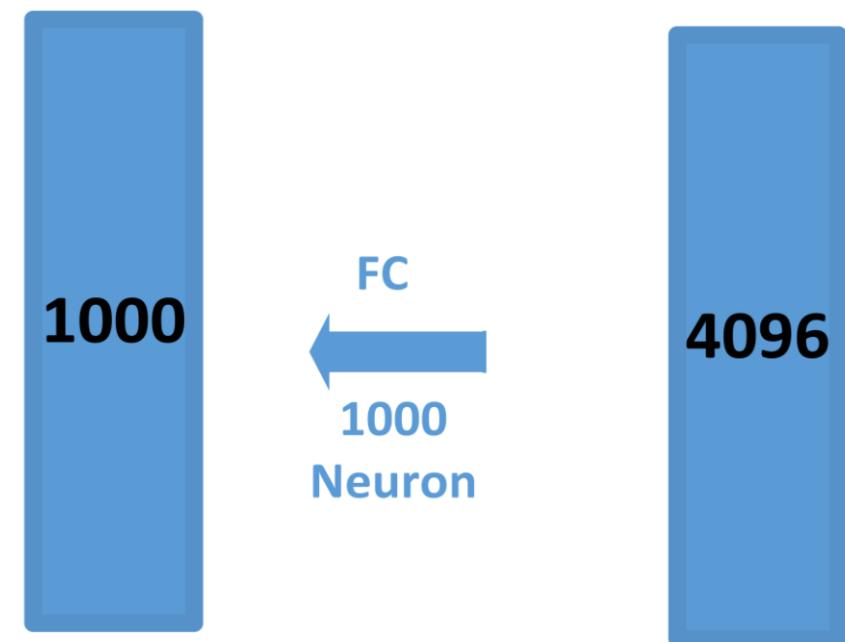
AlexNet Architecture...

- Layer 9: Fully Connected with 4096 neuron
- Layer 10: Fully Connected with 4096 neuron
- Memory: 4096×3 (because of ReLU and Dropout)
- Weights: 4096×4096



AlexNet Architecture...

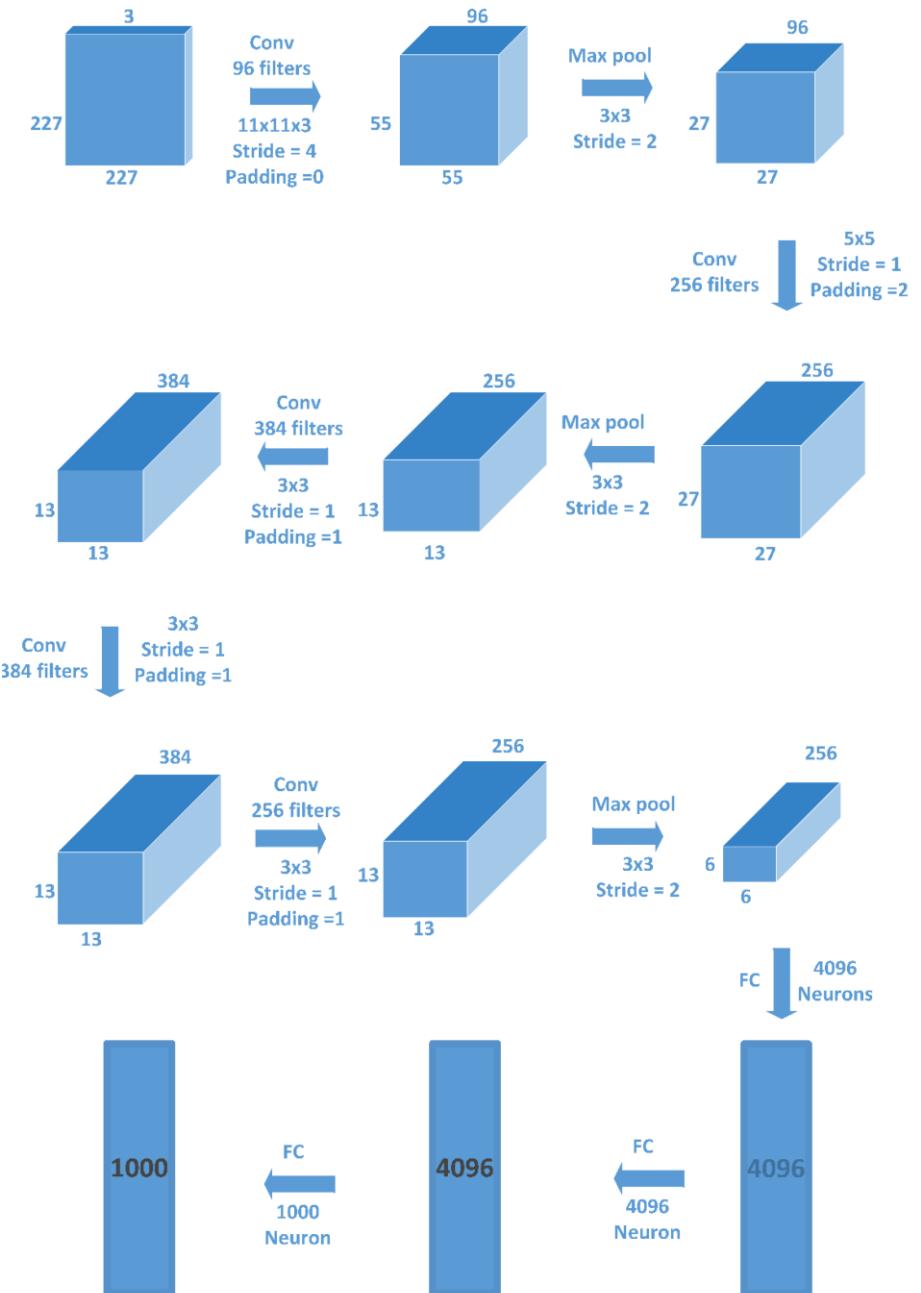
- Layer 10: Fully Connected with 4096 neuron
- Layer 11: Fully Connected with 1000 neurons
- Memory: 1000
- Weights: 4096×1000



AlexNet Architecture...

□ Total

- Memory: 2.24 million
- Weights: 62.37 million
- (bias, label and softmax not included)



AlexNet Architecture...

- First use of ReLU
- Alexnet heavily data augmentation
- Alexnet used dropout 0.5
- Alexnet batch size is 128
- Alexnet used SGD Momentum 0.9
- Alexnet used learning rate 0.01 reduced by a factor of 10

AlexNet Architecture...

[227x227x3] INPUT

[55x55x96] CONV1 : 96 11x11 filters at stride 4, pad 0

27x27x96] MAX POOL1 : 3x3 filters at stride 2

[27x27x96] NORM1: Normalization layer

[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2

[13x13x256] MAX POOL2: 3x3 filters at stride 2

[13x13x256] NORM2: Normalization layer

AlexNet Architecture...

[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1

[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1

[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1

[6x6x256] MAX POOL3: 3x3 filters at stride 2

[4096] FC6: 4096 neurons

[4096] FC7: 4096 neurons

[1000] FC8: 1000 neurons

Outline

- Introduction to Image Classification
- Neural Networks
- Convolutional Neural Networks
- **Datasets for Image Classification**
- Practical Example with CIFAR Dataset
- Further Resources

Most Commonly Used Image Classification Datasets

1. **MNIST** (Modified National Institute of Standards and Technology) dataset.
 - Benchmark dataset for evaluating machine learning and computer vision models, especially those dealing with handwritten digit recognition.
 - Collection of 70,000 grayscale images of handwritten digits (0 through 9).
 - Image Specifications:
 - Size: 28x28 pixels.
 - Color: Grayscale (8-bit, 0-255 pixel values).
 - Dataset Split:
 - Training Set: 60,000 images.
 - Test Set: 10,000 images.

Sample Images from MNIST Dataset

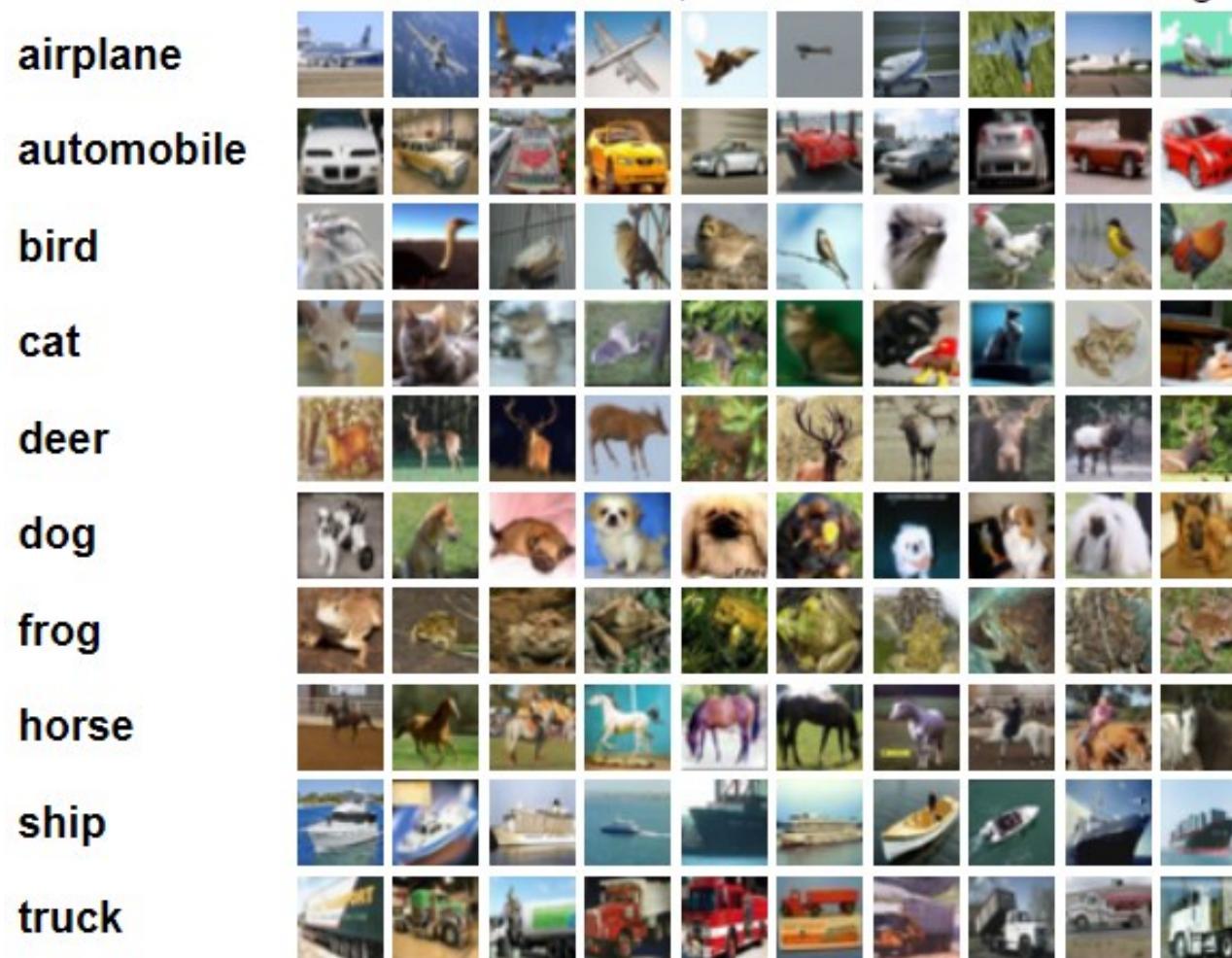
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
7 7 7 7 7 7 7 7 7 7 7 7 7 7 7
8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
9 9 9 9 9 9 9 9 9 9 9 9 9 9 9

Most Commonly Used Image Classification Datasets

2. **CIFAR** (Canadian Institute For Advanced Research) datasets.

- Variants: Two primary versions - **CIFAR-10** and **CIFAR-100**.
- Benchmark datasets for image recognition algorithms.
- Content:
 - CIFAR-10: 60,000 32x32 color images in 10 classes, with 6,000 images per class.
 - Classes: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck.
 - CIFAR-100: 60,000 32x32 color images in 100 classes, with 600 images per class.
- Image Specifications:
 - Size: 32x32 pixels.
 - Color: RGB (3-channel color images).
- Dataset Split:
 - Training Set: 50,000 images.
 - Test Set: 10,000 images.

Sample Images from CIFAR-10 Dataset

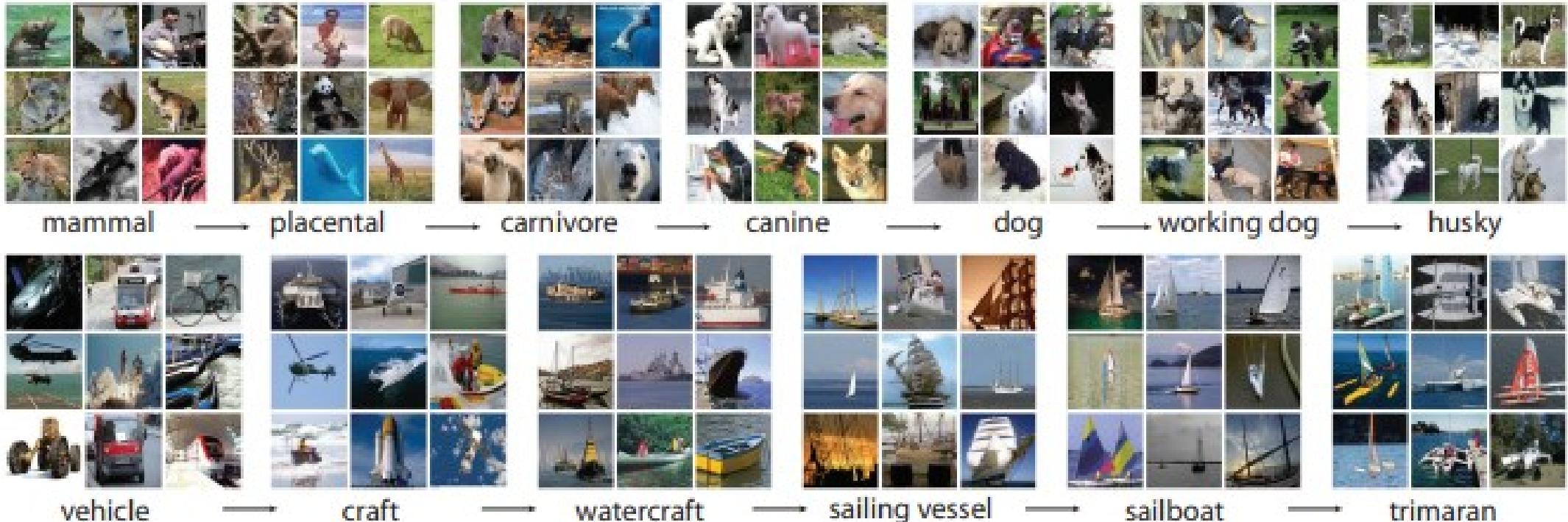


Most Commonly Used Image Classification Datasets

3. ImageNet

- A large-scale dataset used for training and benchmarking in computer vision and deep learning tasks, particularly image classification and object recognition.
- Over 14 million labeled images.
- Classes: Organized according to the WordNet hierarchy, with each meaningful concept in WordNet, possibly described by multiple words or word phrases, having an average of 500-1000 images.
- Image Specifications:
 - Size: Varies; high resolution.
 - Color: Full-color (RGB).
- Dataset Split: Varies by challenge and usage, but typically includes training, validation, and testing sets.

Sample Images from ImageNet Dataset



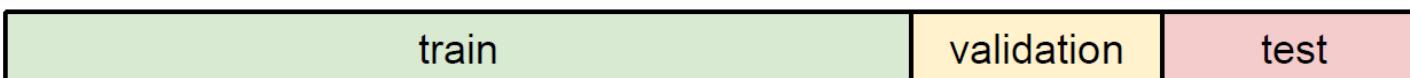
https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf

Hyperparameter Selection

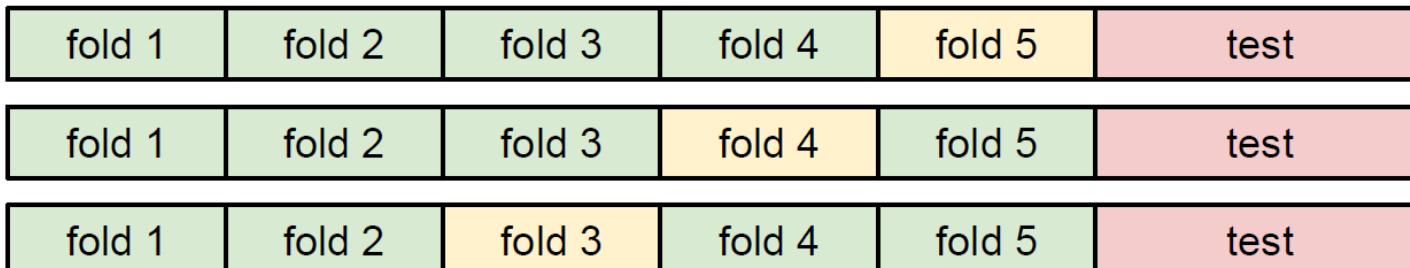
- Split data into **train** and **test**, choose hyperparameters that work best on test data
 - **BAD:** No idea how algorithm will perform on new data



- Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test
 - **Better!**



- **Cross-Validation:** Split data into **folds**, try each fold as validation and average the results



- Useful for small datasets, but not used too frequently in deep learning

Outline

- Introduction to Image Classification
- Neural Networks
- Convolutional Neural Networks
- Datasets for Image Classification
- Practical Example with CIFAR Dataset
- Further Resources

CNN Implementation

- There are multiple frameworks to implement CNNs. The most popular being [PyTorch](#) and [TensorFlow](#).
- The sample code provided here uses PyTorch. In addition, you will need [TorchVision](#).
- To install:
 - `pip install torch torchvision`

Steps for Image Classification using a CNN

1. Define the Model
 - We will develop [AlexNet](#) (from Scratch)
2. Define Loss Function and Optimizer
3. Load and Prepare Dataset
 - We will use [CIFAR-10](#) for this example
4. Train the Model
5. Test the Model

```
# Imports
import torch
import torchvision

import torch.nn as nn
from torch.utils.data.dataset import random_split
from torch.utils.data import DataLoader
import torch.optim as optim

import torchvision.models as models
import torchvision.transforms as transforms
```

1. Define the AlexNet Model

```
class AlexNet(nn.Module):
    def __init__(self, num_classes=1000):
        super(AlexNet, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=11, stride=4, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(64, 192, kernel_size=5, padding=2),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
            nn.Conv2d(192, 384, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(384, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=3, stride=2),
        )
        self.avgpool = nn.AdaptiveAvgPool2d((6, 6))
        self.classifier = nn.Sequential(
            nn.Dropout(),
            nn.Linear(256 * 6 * 6, 4096),
            nn.ReLU(inplace=True),
            nn.Dropout(),
            nn.Linear(4096, 4096),
            nn.ReLU(inplace=True),
            nn.Linear(4096, num_classes),
        )

    def forward(self, x):
        x = self.features(x)
        x = self.avgpool(x)
        x = x.view(x.size(0), 256 * 6 * 6)
        x = self.classifier(x)
        return x
```

```

# Check if CUDA is available
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Create an instance of the model
model = AlexNet(num_classes=10).to(device)

# Print the model architecture
print(model)

```

```

AlexNet(
    (features): Sequential(
        (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
        (1): ReLU(inplace=True)
        (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
        (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
        (4): ReLU(inplace=True)
        (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
        (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (7): ReLU(inplace=True)
        (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (9): ReLU(inplace=True)
        (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
        (11): ReLU(inplace=True)
        (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
    (classifier): Sequential(
        (0): Dropout(p=0.5, inplace=False)
        (1): Linear(in_features=9216, out_features=4096, bias=True)
        (2): ReLU(inplace=True)
        (3): Dropout(p=0.5, inplace=False)
        (4): Linear(in_features=4096, out_features=4096, bias=True)
        (5): ReLU(inplace=True)
        (6): Linear(in_features=4096, out_features=10, bias=True)
    )
)

```

2. Define Loss Function and Optimizer

```
criterion = nn.CrossEntropyLoss()  
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)
```

3. Load and Prepare CIFAR-10 Dataset

```
# Transformations for the CIFAR-10 dataset
transform = transforms.Compose([
    transforms.Resize(224), # Resize images to fit AlexNet's input size
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

# Load CIFAR-10 training and test sets
train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)

# Creating a validation set from the training set
train_size = int(0.8 * len(train_dataset))
val_size = len(train_dataset) - train_size
train_dataset, val_dataset = random_split(train_dataset, [train_size, val_size])

train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True, num_workers=2)
val_loader = DataLoader(val_dataset, batch_size=64, shuffle=False, num_workers=2)
test_loader = DataLoader(test_dataset, batch_size=64, shuffle=False, num_workers=2)
```

Training Function

```
def train_model(model, train_loader, criterion, optimizer, device):
    model.train() # Set the model to training mode

    running_loss = 0.0
    for inputs, labels in train_loader:
        inputs, labels = inputs.to(device), labels.to(device)

        optimizer.zero_grad()

        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    return running_loss / len(train_loader)
```

Validation Function

```
def validate_model(model, val_loader, device):
    model.eval()
    total = 0
    correct = 0
    with torch.no_grad():
        for inputs, labels in val_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    print(f'Validation Accuracy: {accuracy}%')
```

Testing Function

```
def test_model(model, test_loader, device):
    model.eval() # Set the model to evaluation mode
    correct = 0
    total = 0

    with torch.no_grad():
        for inputs, labels in test_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs.data, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    print(f'Accuracy on the test set: {accuracy}%')
```

4. Train the Model

```
num_epochs = 20
for epoch in range(num_epochs):
    train_loss = train_model(model, train_loader, criterion, optimizer, device)
    print(f'Epoch {epoch + 1}/{num_epochs}, Loss: {train_loss}')

# Validation step
validate_model(model, val_loader, device)
```

Epoch 1/20, Loss:
1.4952634948730468
Validation Accuracy: 48.83%
Epoch 2/20, Loss:
1.4009022371292115
Validation Accuracy: 51.77%
Epoch 3/20, Loss:
1.3326138536453247
Validation Accuracy: 53.71%
Epoch 4/20, Loss:
1.2592145084381103
Validation Accuracy: 56.17%
Epoch 5/20, Loss:
1.1939248827934266
Validation Accuracy: 60.88%
Epoch 6/20, Loss:
1.1271399013519288
Validation Accuracy: 62.23%
...
Epoch 18/20, Loss:
0.5888950924396514
Validation Accuracy: 77.26%
Epoch 19/20, Loss:
0.5628529898881912
Validation Accuracy: 78.27%
Epoch 20/20, Loss:
0.5252025803804398
Validation Accuracy: 78.55%

5. Test the Model

```
test_model(model, test_loader, device)
```

Accuracy on the test set: 78.52%

Using a Pre-trained Model (as is)

- We could have used the pre-trained AlexNet.

```
import torchvision.models as models
```

```
# Load the pre-trained AlexNet model
alexnet_pretrained = models.alexnet(pretrained=True)
alexnet_pretrained = alexnet_pretrained.to(device)
```

```
# If you want to use the model for inference (i.e., you don't want to train it further),
# you should set it to evaluation mode
alexnet_pretrained.eval()
```

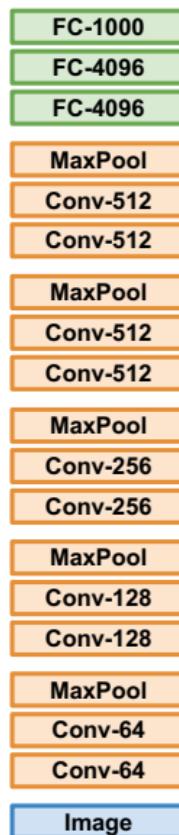
```
# Then test the model
test_model(alexnet_pretrained, test_loader, device)
```

Finetuning a Pre-trained Model

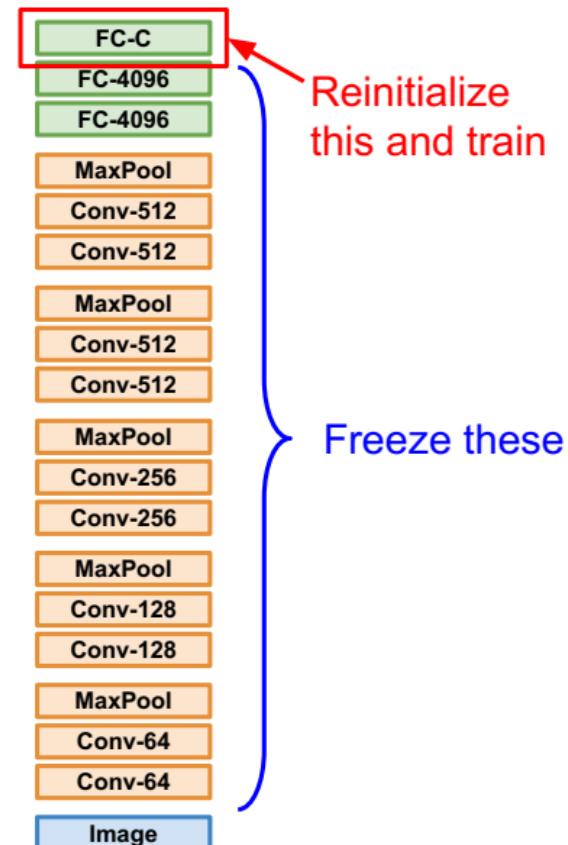
- Finetuning a pre-trained model is one of the most commonly used scenarios. Also called Transfer Learning.
- Extremely effective if:
 - You have a small dataset
 - Your dataset is not entirely different from the dataset the model is trained on

Transfer Learning with CNNs

1. Train on Imagenet



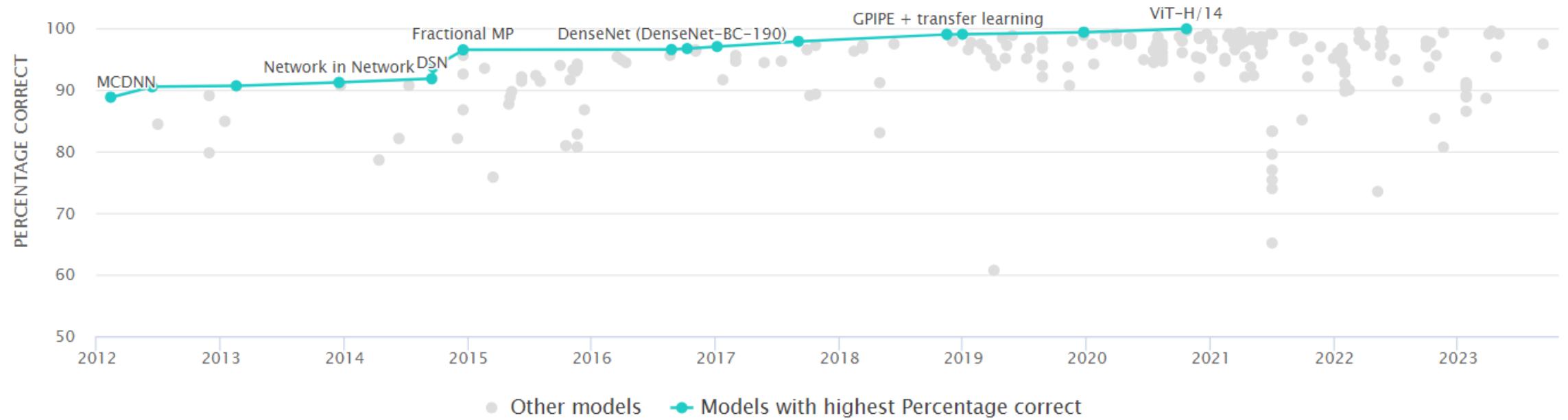
2. Small Dataset (C classes)



Assuming CIFAR-10 with 10 classes

```
alexnet_pretrained.classifier[6] = nn.Linear(alexnet_pretrained.classifier[6].in_features, 10)
```

State-of-the-Art on CIFAR-10 Image Classification



<https://paperswithcode.com/sota/image-classification-on-cifar-10>

Outline

- Introduction to Image Classification
- Neural Networks
- Convolutional Neural Networks
- Datasets for Image Classification
- Practical Example with CIFAR Dataset
- **Further Resources**

Online Demos for DL

demos for CNN by Andrej Karpathy

Places: Scene classification with neural nets

drawNet: Visualization of ConvNet activations

Deep Learning for Computer Vision: A Stanford course

Popular Datasets in Visual Recognition

<u>ImageNet</u> : Large-scale object dataset	<u>Microsoft Coco</u> : Large-scale image recognition, segmentation, and captioning dataset
<u>Cityscapes</u> : Autonomous driving dataset	<u>PASCAL VOC</u> : Object recognition dataset
<u>KITTI</u> : Autonomous driving dataset	<u>NYUv2</u> : Indoor RGB-D dataset
<u>VQA</u> : Visual question answering dataset	<u>MovieDescription</u> : a dataset for automatic description of movie clips
<u>Flickr30K</u> : Image captioning dataset	<u>MPI Sintel Dataset</u> : optical flow dataset

A huge list of image processing datasets is available at
<https://paperswithcode.com/datasets>

State-of-the-Art

- Links to the best work (along with its open-source implementations) in each area of image processing and computer vision is available at
<https://paperswithcode.com/sota>

Resources and Credits

- Convolutional Neural Network, Mohamed Loey, Behnac University, Egypt
- CS231N, Convolutional Neural Networks for Visual Recognition, Stanford University