



Computer Vision IT416

Dinesh Naik.

Dept of Information Technology

Special Thanks to: Eminent Scholars of Computer Vision and Other Resource base whose slides are directly referred.

Course Description

Introduction

I. low-level vision

- Computer and Image Processing Introduction
- History of Photography
- Applications and Usage
- Concept of Dimensions
- Image Formation on Camera
- Camera Mechanism
- Concept of Pixel
- Perspective Transformation
- Concept of Bits per Pixel
- Types of Images
- Color Codes Conversion
- Gray scale to RGB Conversion
- Concept of Sampling
- Pixel Resolution
- Concept of Zooming
- Zooming methods
- Spatial Resolution
- Pixels Dots and Lines per inch
- Gray Level Resolution
- Concept of Quantization
- Concept of Dithering
- Histograms Introduction

- Brightness and Contrast
- Image Transformations
- Histogram Sliding
- Histogram Stretching
- Histogram Equalization
- Gray Level Transformations
- Concept of convolution
- Concept of Masks
- Concept of Blurring
- Concept of Edge Detection
- Prewitt Operator
- Sobel operator
- Robinson Compass Mask
- Krisch Compass Mask
- Laplacian Operator
- Fourier series and Transform
- Convolution theorem
- Linear filters
- Edges and contours
- Binary image analysis
- Background subtraction
- Texture
- Motion and optical flow

II. Mid-level vision

- Segmentation and clustering algorithms
- Hough transform
- Fitting lines and curves
- Robust fitting, RANSAC
- Deformable contours
- Interactive segmentation

III. Multiple views

- Local invariant feature detection and description
- Image transformations and alignment
- Planar homography
- Epipolar geometry and stereo
- Object instance recognition

IV. Recognition: high-level vision

- Basics of Object detection and recognition.
- Supervised classification algorithms
- Dimensionality reduction
- Deep learning, Convolutional neural networks

Reference Books

The course textbook is:

[Computer Vision: Algorithms and Applications, by Rick Szeliski.](#)

- Computer Vision: A Modern Approach, David A. Forsyth and Jean Ponce
- Computer Vision, Linda G. Shapiro and George C. Stockman
- Introductory Techniques for 3-D Computer Vision, Emanuele Trucco and Alessandro Verri.
- Multiple View Geometry in Computer Vision, Richard Hartley and Andrew Zisserman.
- Pattern classification, Richard O. Duda, Peter E. Hart, and David G. Stork
- Pattern Recognition and Machine Learning. Christopher M. Bishop
- [Visual Object Recognition](#). K. Grauman and B. Leibe

Journals:

- IEEE-T-PAMI (Transactions on Pattern Analysis and Machine Intelligence)
- IEEE-T-IP (Transactions on Image processing)
- PR (Pattern Recognition)
- PRL (Pattern Recognition Letters)
- CVGIP (Computer Vision, Graphics & Image Processing)
- IJCV (International Journal of Computer Vision)

Evaluation Scheme

- **Mid Sem—20%**
- **End Sem—35%**
- **Assignment—45%**

Assignment Guidelines

Projects should be done GroupWise.

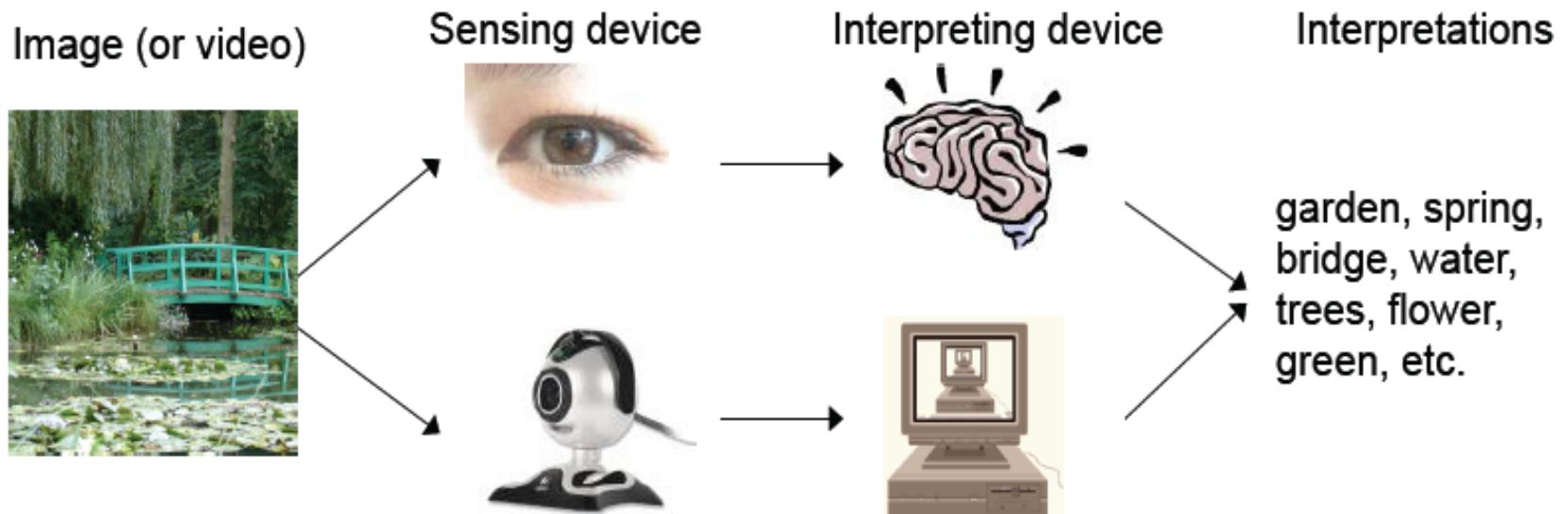
All group members should have individual unique contributions to the project (which must be stated clearly in your final report), and yet, be aware of what the other group members did. Note that this will be tested during the viva.

Your project work will be a research paper implementation[given by CI], or your own idea. In any case, if you wish to have some feedback about the triviality or difficulty of your project, please speak to me .

You will required to submit a report and appear for a viva during which you will demo your project, and answer questions about it. The final report should clearly but briefly describe the problem statement, a description of the main algorithm(s) you implemented, a description of the datasets on which they were tested, a detailed description of the results followed by a conclusion including an analysis of the good and bad aspects of your implementation or the algorithm.

You may use MATLAB/C/C++/Java/Python + any packages (OpenCV,ITK, etc) for your project. But merely invoking calls to someone's else's software is not substance enough. You should have your own non-trivial coding component. If software for the research paper you implement is already available, you should use it only for comparison sake - you will be expected to implement the paper on your own. Please discuss with me if you need any clarifications for your specific case.

What is (computer) vision?



The goal of computer vision

- To bridge the gap between pixels and “meaning”



La Gare Montparnasse, 1895

What we see

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

What a computer sees

Why study computer vision?

- Vision is useful: Images and video are everywhere!



GoogleTM
Image Search PicasaTM

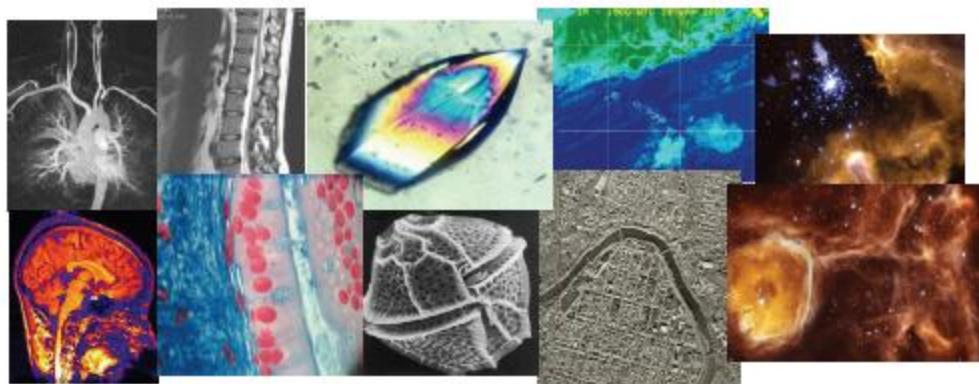
flickrTM GANNA
webshots^{Beta}

picsearchTM

YouTubeTM
Broadcast YourselfTM



Surveillance and security



Medical and scientific images

To do Research on Computer vision areas.

- Inherently **interdisciplinary** subject: numerous application areas - remote sensing, photography, visual psychology, archaeology, surveillance, etc.
- Fast becoming a popular field of study in India: scope for R&D work in numerous research labs (In India: Samsung, GE, Phillips, Siemens, Microsoft, HP, TI, Google; DRDO, ICRISAT, ISRO, etc.)



What is vision?

- What does it mean to see ?

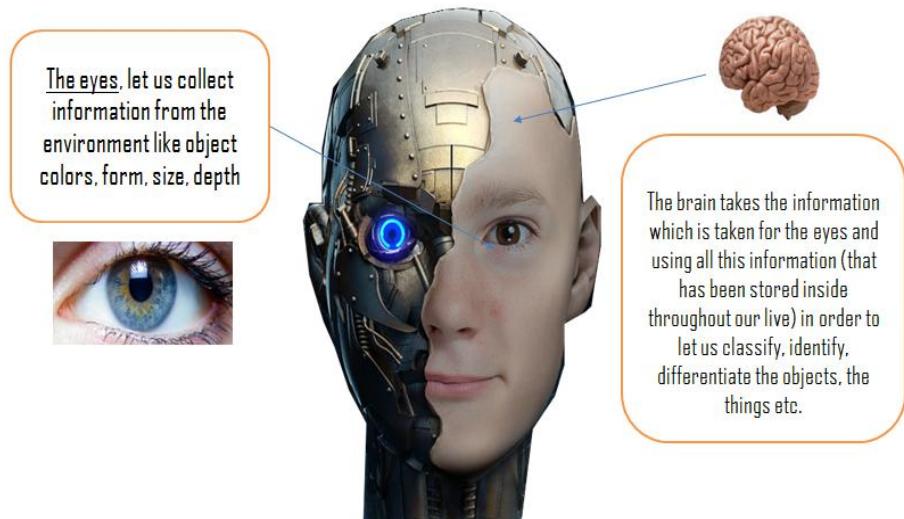


"The plain man's answer (and Aristotle's too) would be, to know what is where by looking. In other words, vision is the process of discovering from images what is present in the world, and where it is " David Marr, Vision 1982

What is vision?

- Recognize objects
 - people we know
 - things we own
- Locate objects in space
 - to pick them up
- Track objects in motion
 - catching a baseball
 - avoiding collisions with cars on the road
- Recognize actions
 - walking, running, pushing

Human vision System



What is Computer Vision?

- Make computers understand images and videos.



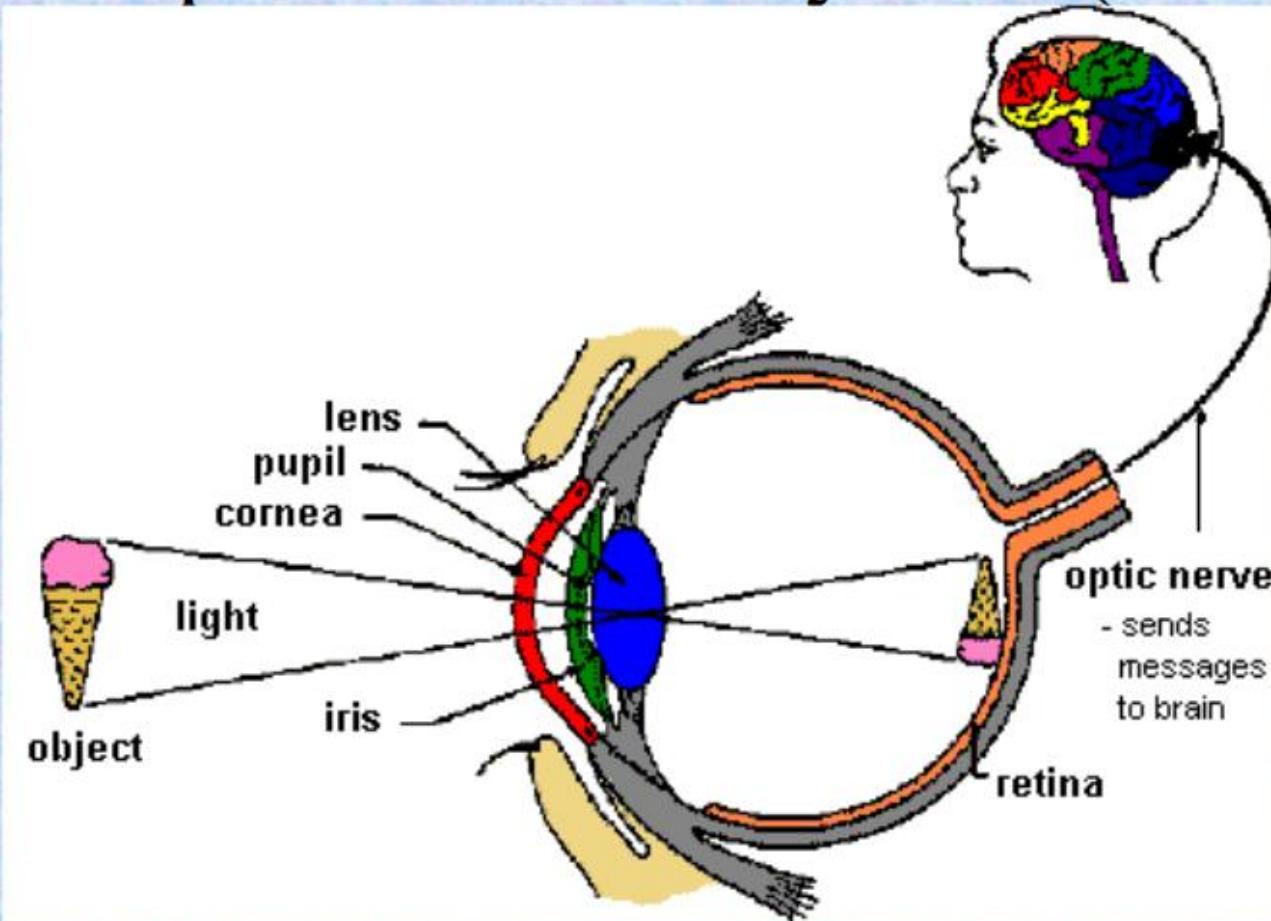
- What kind of scene?
- Where are the cars?
- How far is the building?

Every picture tells a story



Goal of computer vision
is to write computer
programs that can
interpret images

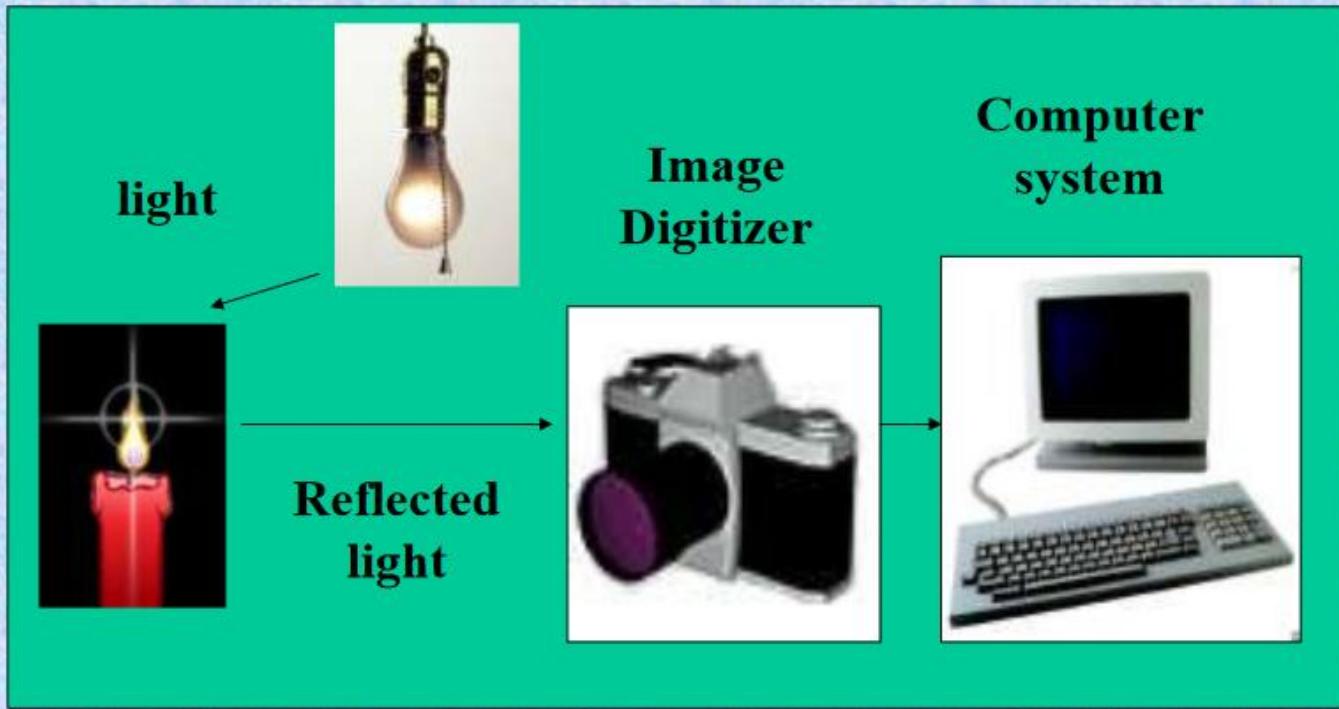
Human Vision System (HVS) Vs. Computer Vision System (CVS)



The Optics of the eye

7

A computer Vision System (CVS)

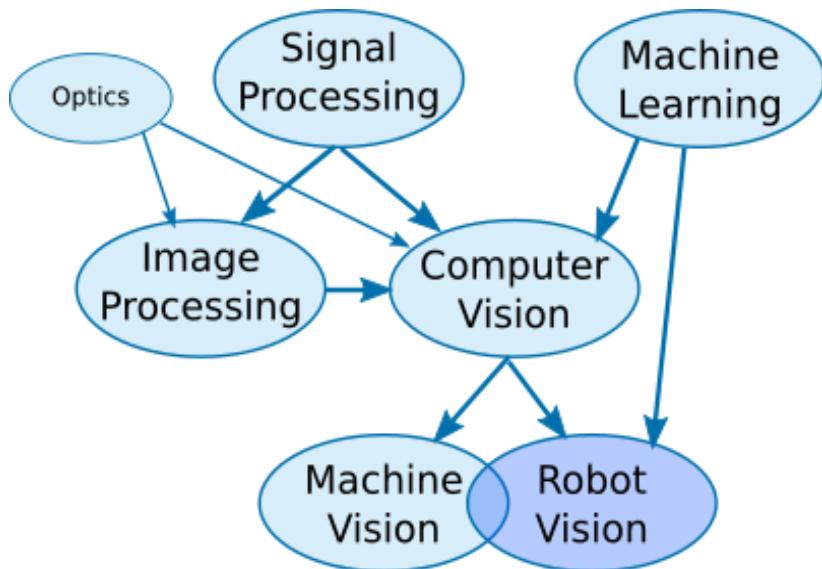


What's the difference between

- Robot Vision,
- Computer Vision,
- Image Processing,
- Machine Vision
- And Pattern Recognition.

What is Robot Vision?

- **Robot Vision** involves using a combination of camera hardware and computer algorithms to allow robots to process visual data from the world.
- Robot Vision is closely related to **Machine Vision**
- They are both closely related to Computer Vision.



Signal Processing involves processing electronic signals to either clean them up (e.g. removing noise), extract information, prepare them to output to a display or prepare them for further processing

Machine Vision

- Machine Vision refers to the industrial use of vision for automatic inspection, process control and robot guidance.
- Machine Vision is an engineering domain.
- In some ways, you could think of it as a child of Computer Vision.

Technique	Input	Output
Signal Processing	Electrical signals	Electrical signals
Image Processing	Images	Images
Computer Vision	Images	Information/features
Pattern Recognition/Machine Learning	Information/features	Information
Machine Vision	Images	Information
Robot Vision	Images	Physical Action

Computer Vision and Image Processing: What's the difference?

- Difference is **blurry**
- “Image processing” typically involves processing/analysis of (2D) images without referring to underlying 3D structure
- Computer vision – typically involves inference of **underlying 3D structure** from 2D images
- Many computer vision techniques also aim to infer properties of the scene directly – without 3D reconstruction.
- Computer vision – direct opposite of computer graphics

- In a way, Computer Vision can be considered the inversion of Computer Graphics.
- A computer graphics systems receives as its input the formal description of a visual scene, and its output is a visualization of that scene.
- A computer vision system receives as its input a visual scene, and its output is a formal description of that scene with regard to the system's task.
- Unfortunately, while a computer graphics task only allows one solution, computer vision tasks are often ambiguous, and it is unclear what the correct output should be.

Image Processing vs Computer Vision

- Computer Vision and Image Processing are like cousins, but they have quite different aims
- Computer Vision, on the other hand, is more about extracting information from images to make sense of them.
- we see that both of these domains are heavily influenced by the domain of Physics, specifically Optics

Image Processing techniques are primarily used to improve the quality of an image, convert it into another format (like a histogram) or otherwise change it for further processing

- If we include Pattern Recognition into the family tree, or more broadly Machine Learning. This branch of the family is focused on recognizing patterns in data, which is quite important for many of the more advanced functions required of Robot vision.

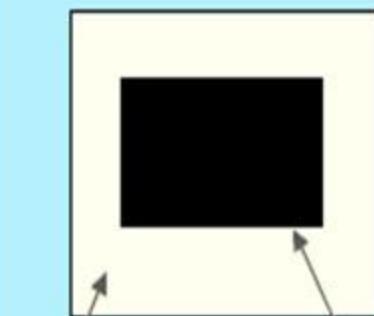
Digital Image processing is in many cases concerned with taking one array of pixels as input and producing another array of pixels as output which in some way represents an improvement to the original array.

Purpose:

1. **Improvement of Pictorial Information**
 - improve the contrast of the image,
 - remove noise,
 - remove blurring caused by movement of the camera during image acquisition,
 - it may correct for geometrical distortions caused by the lens.
2. **Automatic Machine perception** (termed Computer Vision, Pattern Recognition or Visual Perception) for intelligent interpretation of scenes or pictures.

Image acquisition using a CCD camera

ORIGINAL SCENE



Light Areas
(high intensity)

Dark Areas
(low intensity)

Image capture - 'sampling' using digital camera or scanner

Electrical response proportional to light exposure

CCD Response

✓	✓	✓	✓	✓	✓
✓	✗	✗	✗	✗	✓
✓	✗	✗	✗	✗	✓
✓	✓	✓	✗	✓	✓

No response in unexposed areas

Analogue to digital conversion followed by conversion to image pixel values for display

255	255	255	255	255	255
255	0	0	0	0	255
255	0	0	0	0	255
255	255	255	255	255	255

Pixel representations of white and black

Digital Image

A ***digital image*** is a two-dimensional (3-D image is called range data) array of intensity values, $f(x, y)$, which represents 2-D intensity function discretized both in spatial coordinates (**spatial sampling**) and brightness (**quantization**) values.

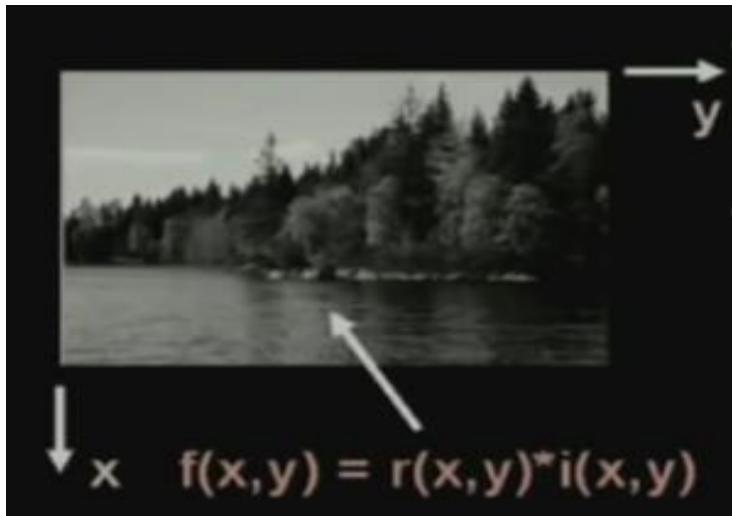
The elements of such an array are called **pixels** (picture elements).

The storage requirement for an image depends on the **spatial resolution** and number of bits necessary for pixel quantization.

The processing of an image depends on the application domain and the methodology used to solve a problem. There exists four broad categories of tasks in digital image processing:

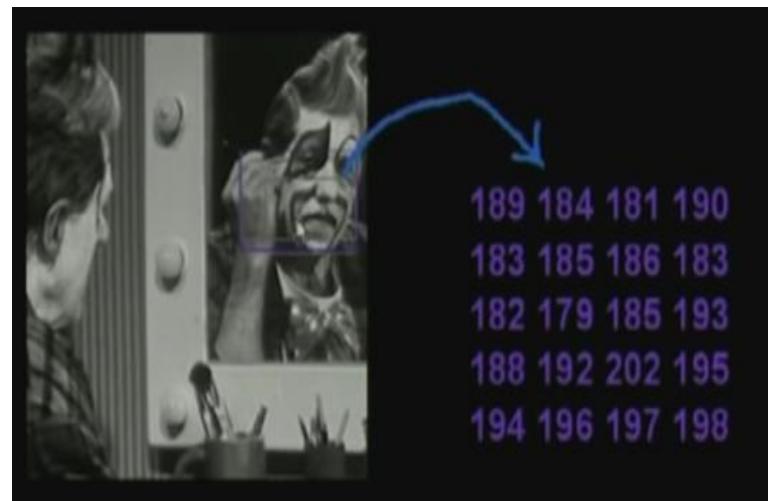
- (i) **Compression,**
- (ii) **Segmentation,**
- (iii) **Recognition and**
- (iv) **motion.**

Image Representation

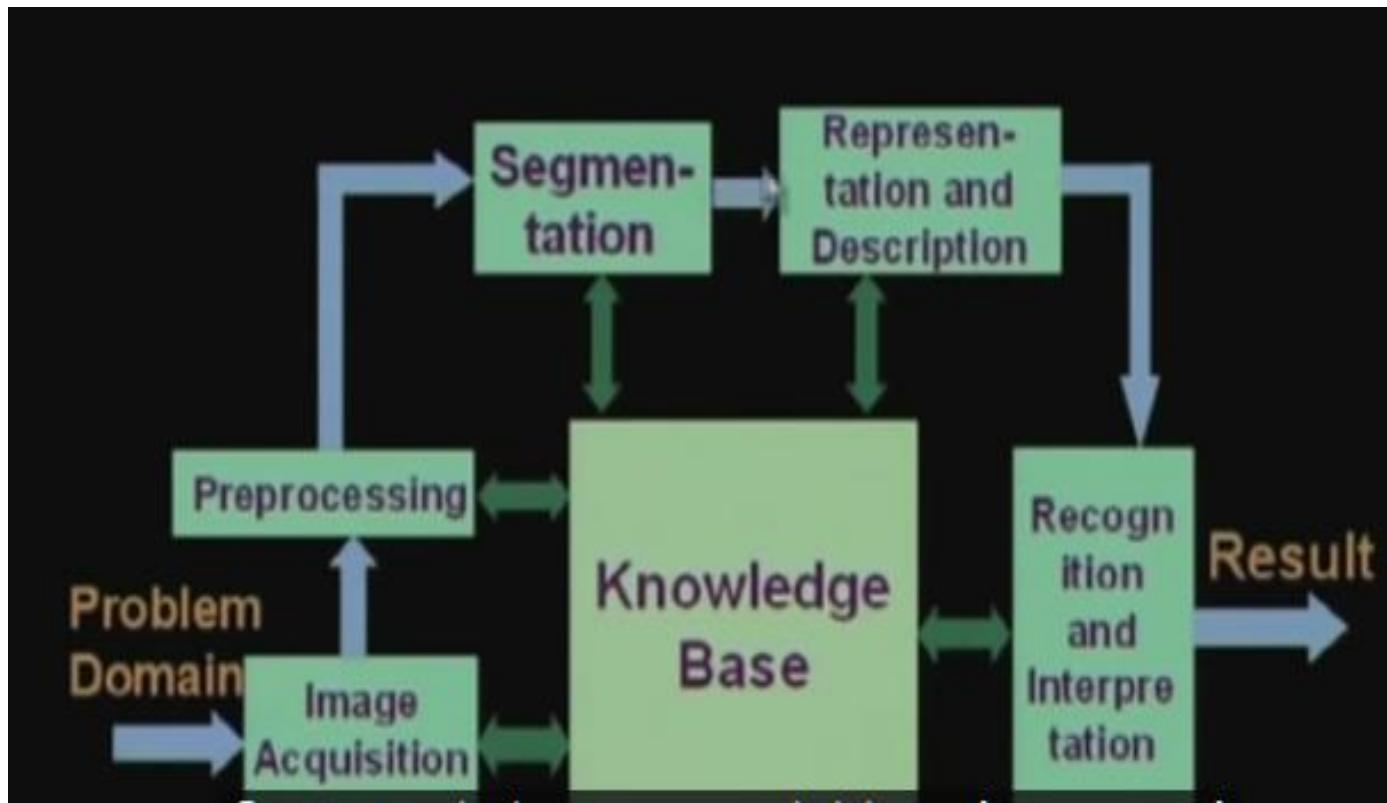


- An image is a 2-D light intensity function $f(x,y)$
- A digital image $f(x,y)$ is discretized both in spatial coordinates and brightness
- It can be considered as a matrix whose row, column indices specify a point in the image and the element value identifies gray level value at that point

-
- Spatial discretization by grids
 - Intensity discretization by quantization



Steps in Digital Image processing



What is digitization

An image to be represented in the form of a finite 2-D matrix

$$\begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1,N-1) \\ f(2,0) & f(2,1) & f(2,2) & \dots & f(2,N-1) \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M,1) & f(M,2) & \dots & f(M-1,N-1) \end{bmatrix}$$

Image representation by 2-D finite matrix –

Sampling

Each Matrix element represented by one of the finite set of discrete values-

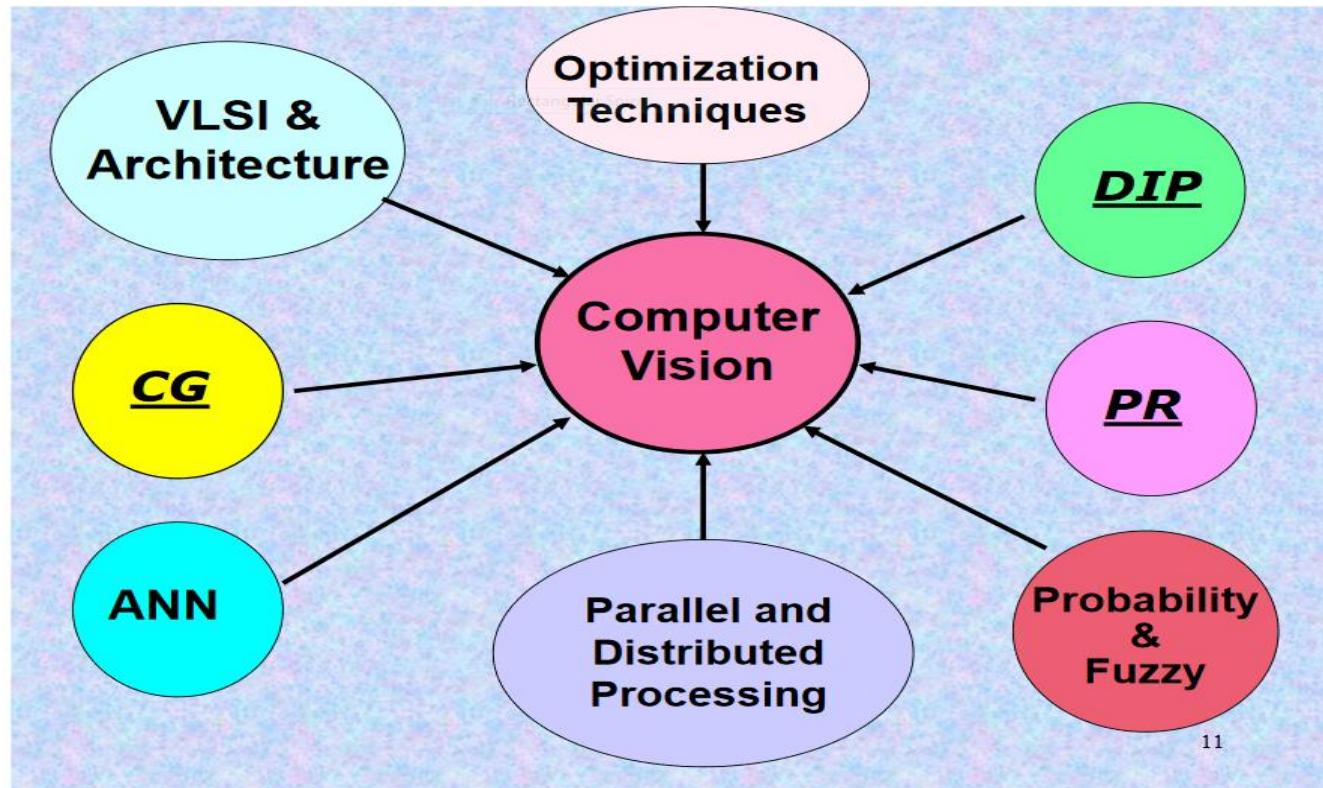
Quantization

There are generally three main categories of tasks involved in a complete computer vision system. They are:

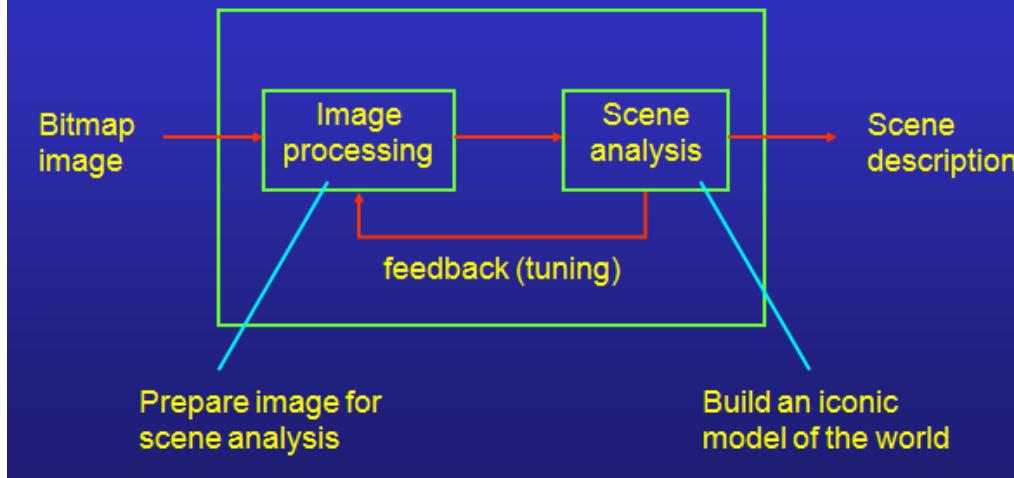
- ***Low level processing:*** Involves image processing tasks in which the quality of the image is improved for the benefit of human observers and higher level routines to perform better.
- ***Intermediate level processing:*** Involves the processes of feature extraction and pattern detection tasks. The algorithms used here are chosen and tuned in a manner as may be required to assist the final tasks of high level vision.
- ***High level vision:*** Involves autonomous interpretation of scenes for pattern classification, recognition and identification of objects in the scenes as well as any other information required for human understanding.

A top down approach, rather than a bottom-up approach is used in the design of these systems in many applications. The methods used to solve a problem in digital image processing depends on the application domain and nature of data being analyzed.

Computer Vision is an area of work, which is a combination of concepts, techniques and ideas from Digital Image Processing, Pattern Recognition, Artificial Intelligence and Computer Graphics



A simple two-stage model of computer vision:



- A simple two-stage model of computer vision:

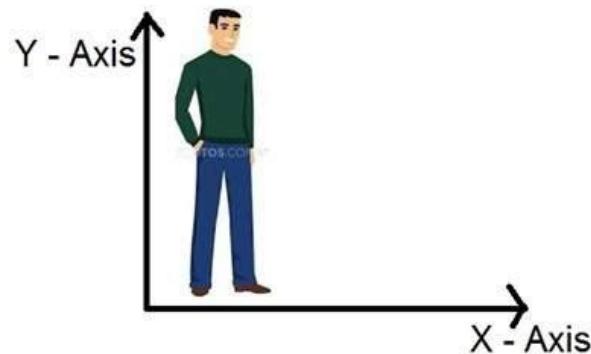
The **image processing** stage **prepares** the input image for the subsequent scene analysis.

Usually, image processing results in one or more **new images** that contain specific information on relevant features of the input image.

The information in the output images is **arranged in the same way** as in the input image. For example, in the upper left corner in the output images we find information about the upper left corner in the input image.

The **scene analysis** stage **interprets** the results from the image processing stage. Its output completely depends on the problem that the computer vision system is supposed to solve. For example, it could be the **number of bacteria** in a microscopic image, or the **identity of a person** whose retinal scan was input to the system.

Concept of Dimensions



History of Computer Vision



Marvin Minsky, MIT
Turing award, 1969

"In 1966, Minsky hired a first-year undergraduate student and assigned him a problem to solve over the summer:

connect a camera to a computer and get the machine to describe what it sees."

Crevier 1993, pg. 88

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

PROJECT MAC

Artificial Intelligence Group
Vision Memo. No. 100.

July 7, 1966

THE SUMMER VISION PROJECT

Seymour Papert

Half a century later
we're still working



The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

1960's: interpretation of synthetic worlds



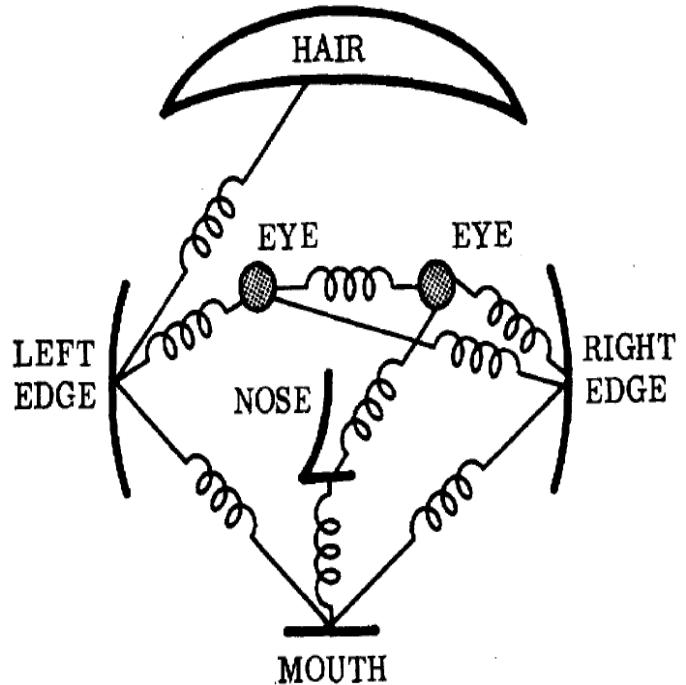
Extracting 3D information about solid objects from 2D photographs of line drawings.

Larry Roberts
“Father of Computer Vision”

Larry Roberts PhD Thesis, MIT, 1963,
Machine Perception of Three-Dimensional Solids

Slide credit: Steve Seitz

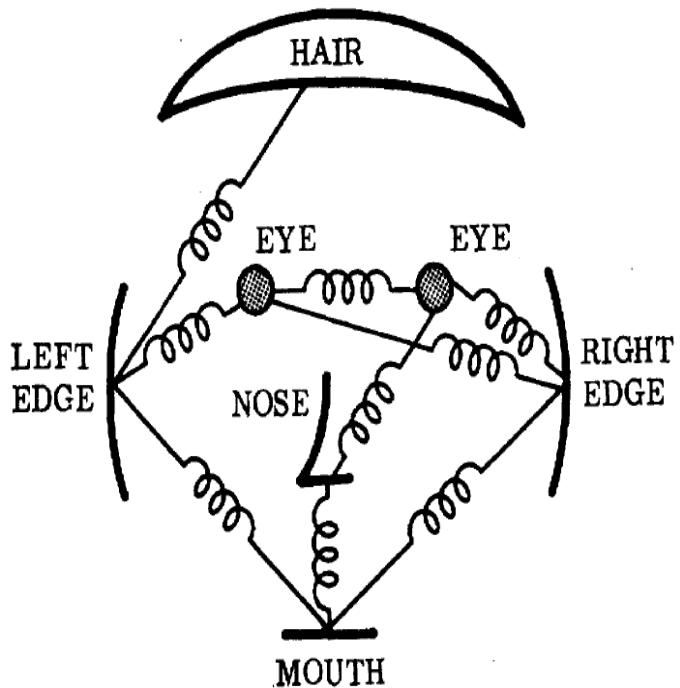
1970's: some progress on interpreting selected images



The representation and matching of pictorial structures

Fischler and Elschlager, 1973

1970's: some progress on interpreting selected images



The representation and matching of pictorial structures

Fischler and Elschlager, 1973

1 .
2 .
3 .
4 .
5 .
6 .
7 .
8 .
9 .
10 .
11 .
12 .
13 .
14 .
15 .
16 .
17 .
18 .
19 .
20 .
21 .
22 .
23 .
24 .
25 .
26 .
27 .
28 .
29 .
30 .
31 .
32 .
33 .
34 .
35 .

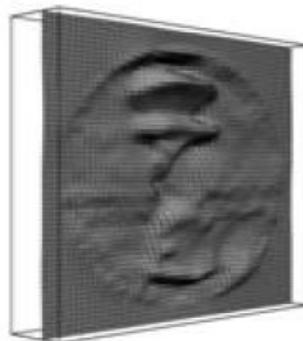
HAIR WAS LOC

HAIR WAS LOCATED AT (13, 23)
L/EDGE WAS LOCATED AT (25, 13)
R/EDGE WAS LOCATED AT (25, 28)
L/EYE WAS LOCATED AT (22, 16)
R/EYE WAS LOCATED AT (22, 23)
NOSE WAS LOCATED AT (27, 20)
MOUTH WAS LOCATED AT (29, 19)

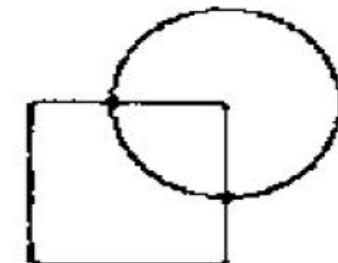
1980's: ANNs come and go; shift toward geometry and increased mathematical rigor



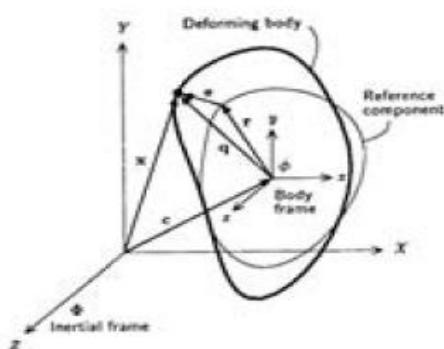
(a)



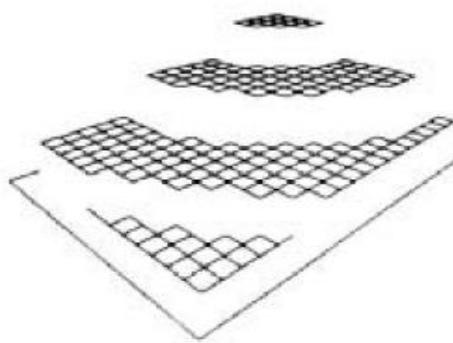
(b)



(c)



(d)



(e)



(f)

Image credit: Rick Szeliski

1990's: face recognition



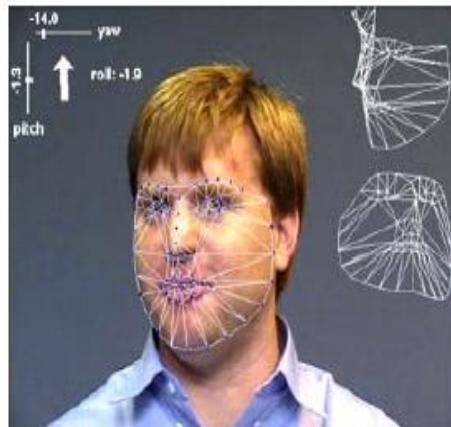
(a)



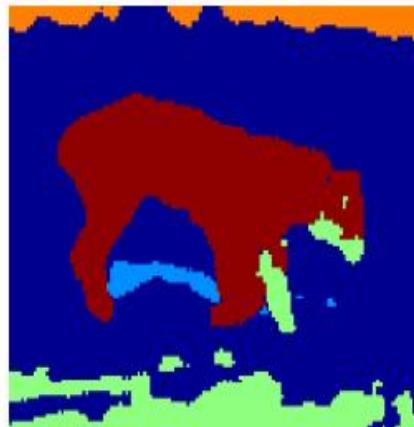
(b)



(c)



(d)



(e)



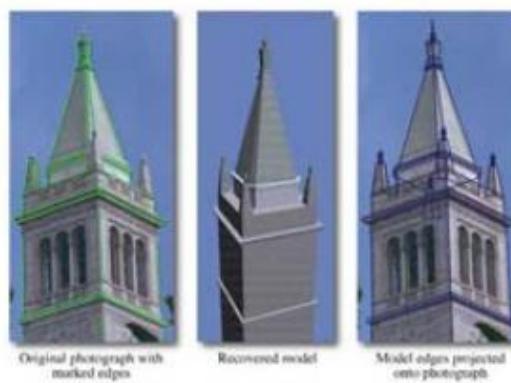
(f)

Image credit: Rick Szeliski

2000's: broader recognition; large annotated datasets available; video processing starts



(a)



Original photograph with
marked edges

Recovered model

Model edges projected
onto photograph



(c)



(d)



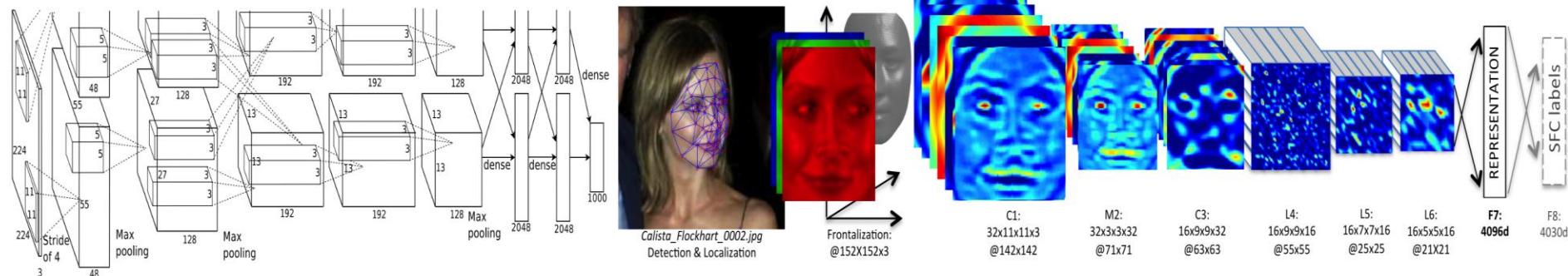
(e)



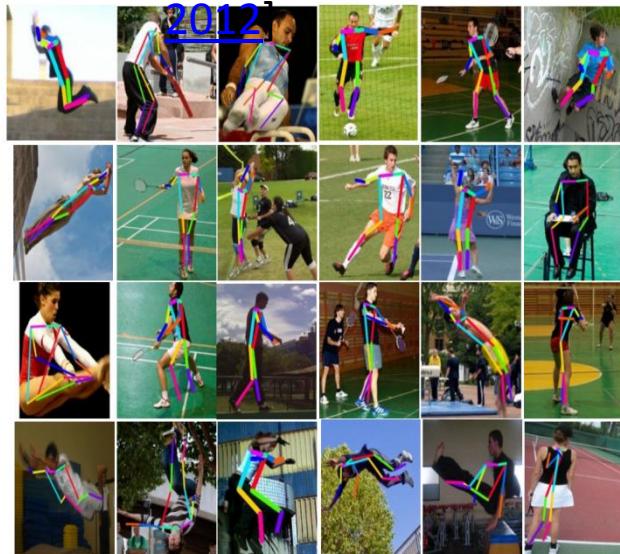
(f)

Image credit: Rick Szeliski

2010's: resurgence of deep learning



[\[AlexNet NIPS\]](#)

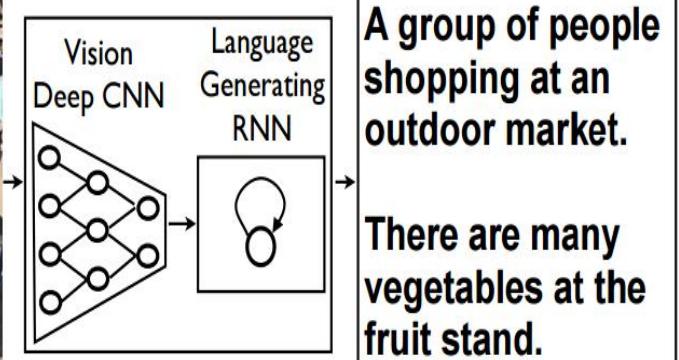


[\[DeepPose CVPR 2014\]](#)

[\[DeepFace CVPR 2014\]](#)



[\[Show, Attend and Tell ICML 2015\]](#)



2020's: autonomous vehicles



2030's: robot uprising?



Different fields of applications include:

- Character Recognition,
- Document processing,
- Commercial (signature & seal verification) application,
- Biometry and Forensic (*authentication: recognition and verification of persons using face, palm & fingerprint*),
- Pose and gesture identification,
- Automatic inspection of industrial products,
- Industrial process monitoring,
- Biomedical Engg. (Diagnosis and surgery),
- Military surveillance and target identification,
- Navigation and mobility (for robots and unmanned vehicles - land, air and underwater),
- Remote sensing (using satellite imagery),
- GIS
- Safety and security (night vision),
- Traffic monitoring,
- Sports (training and incident analysis)
- VLDB (organization and retrieval)
- Entertainment and virtual reality.

TARGETED INDUSTRIAL APPLICATIONS

Intelligent Traffic Control

Vehicle Segmentation

Anti-forging Stamps

Visual Tracking Systems

Card Counting Systems

Illegal content (adult) Filter

Drive Quality Test

Scratch Detection

Camera Flame Detection

Smart Traffic Monitoring

CCTV Fog Penetration

Vehicle Categorization

Key Image Search/Index

Vehicle Wheel alignment

Security Monitoring

Number Plate Identification

Robust Shadow Detection

Referrals for Line calls

Different categories of work being done in CV, to solve problems:

**2-D image analysis –
segmentation, target detection,
matching, CBIR;**

**3-D multi-camera calibration;
Correspondence and stereo;
Reconstruction of
3-D Objects and surfaces;**

**Pattern Recognition
for Objects, scenes;**

**Video and motion analysis;
Video analytics; CBVR;
Compression;**

**Feature extraction:
Canny, GHT, Snakes,
DWT, Corners,
SIFT, GLOH, LESH;**

**Multi-sensor data,
Decision and feature fusion;**

**Image and Video-based
Rendering;**

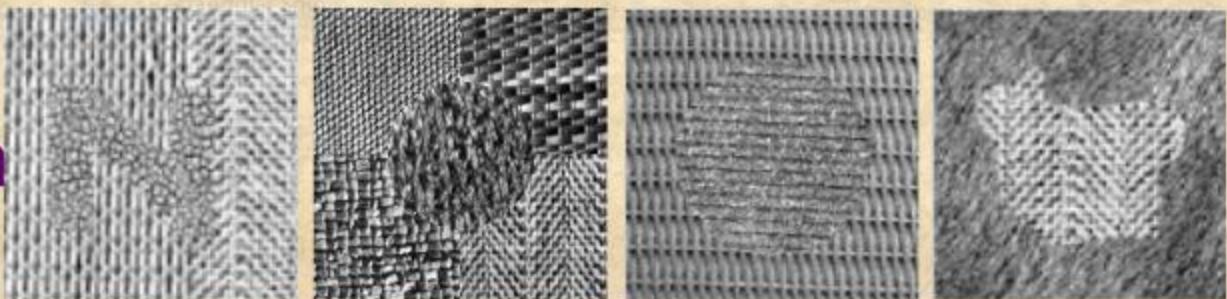
**Steganography and
Watermarking;**

The various sub-categories of technology in these related fields are:

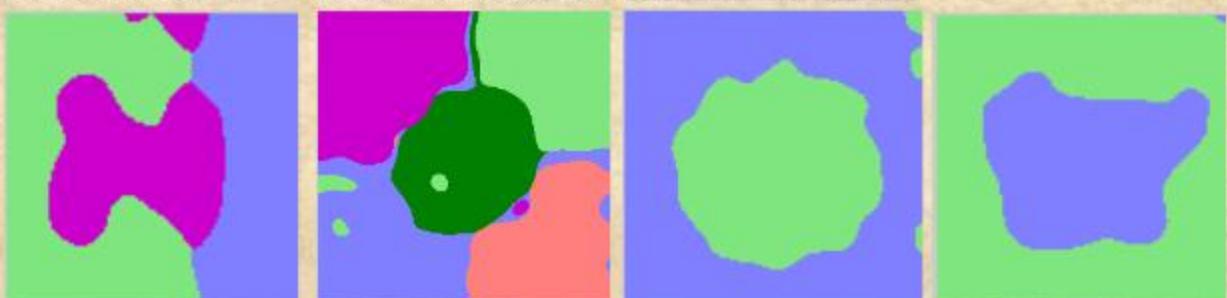
- *image enhancement*,
- *image reconstruction*
- *image restoration and filtering*,
- *range data processing*,
- *representation and description*,
- *stereo image processing*
- *feature extraction*,
- *computational geometry*,
- *image segmentation*,
- *image morphology*,
- *image matching*,
- *artificial neural networks*,
- *color image processing*,
- *Neuro-fuzzy techniques*,
- *image synthesis*,
- *computational geometry*,
- *image representation*,
- *parallel architectures & algorithms*.

Results of Segmentation

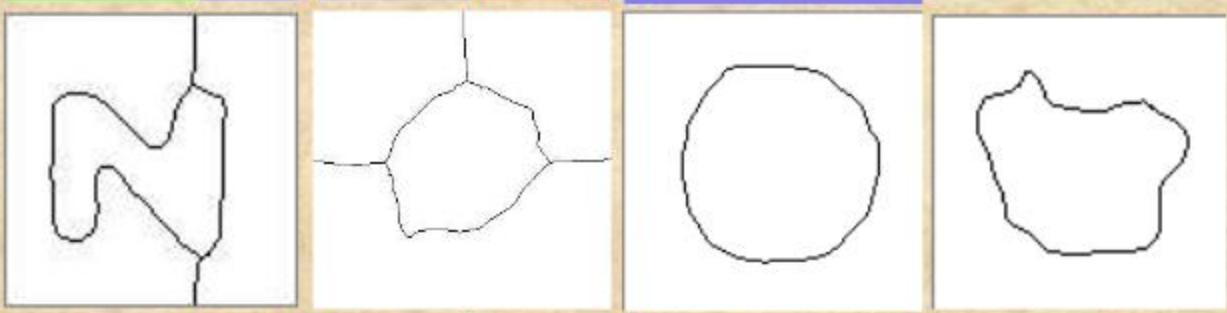
Input Image



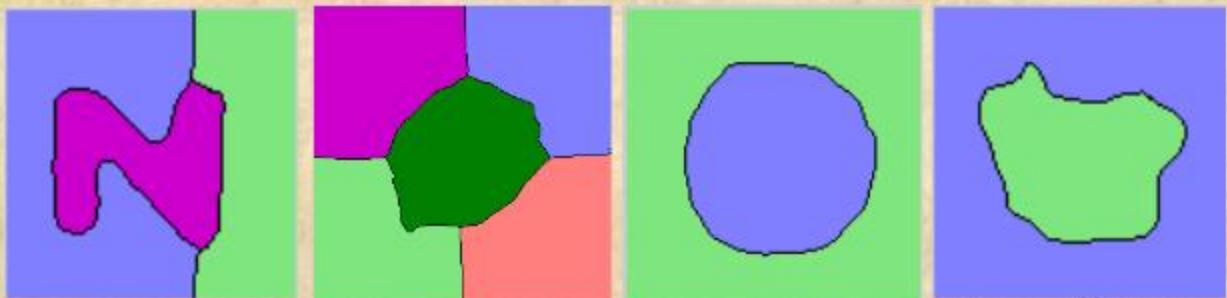
Segmented map
before integration



Edge map before
integration

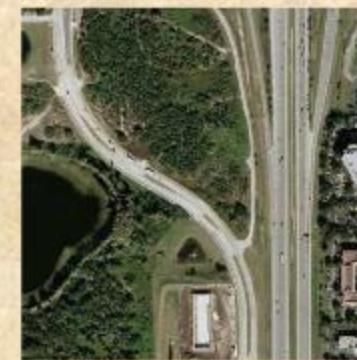


Segmented map
and Edge map
after integration

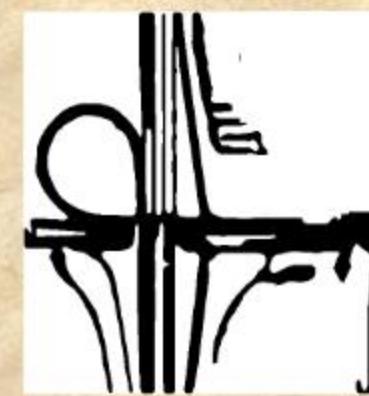
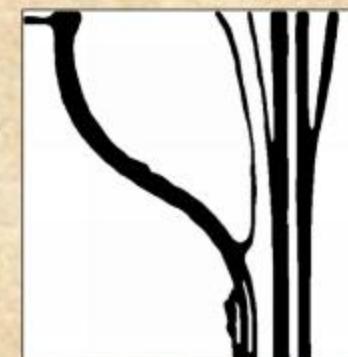


Road extraction from Satellite Images

SAT
Images



Results



Hand-
drawn

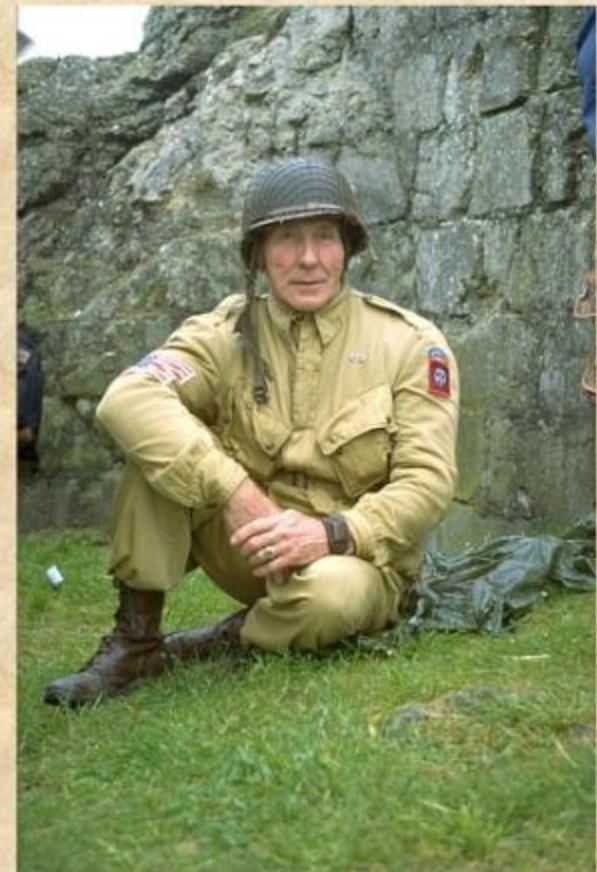


SNAKECUT – Extraction of a Foreground Object with holes

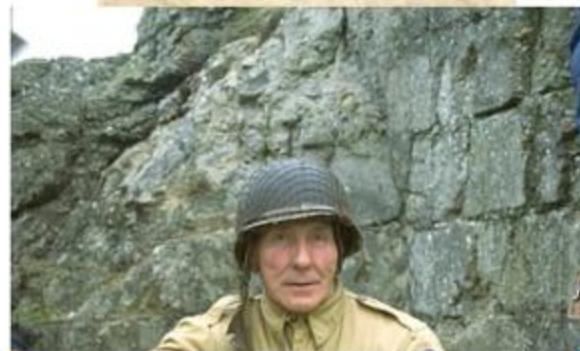
Our proposed approach for segmentation of an object with a hole, using a combination of (i) Active Contour and (ii) GrabCut

■ Here, objective is to crop the soldier from the input image

■ Cropped image should not contain any part of the background



**Snake
Output**



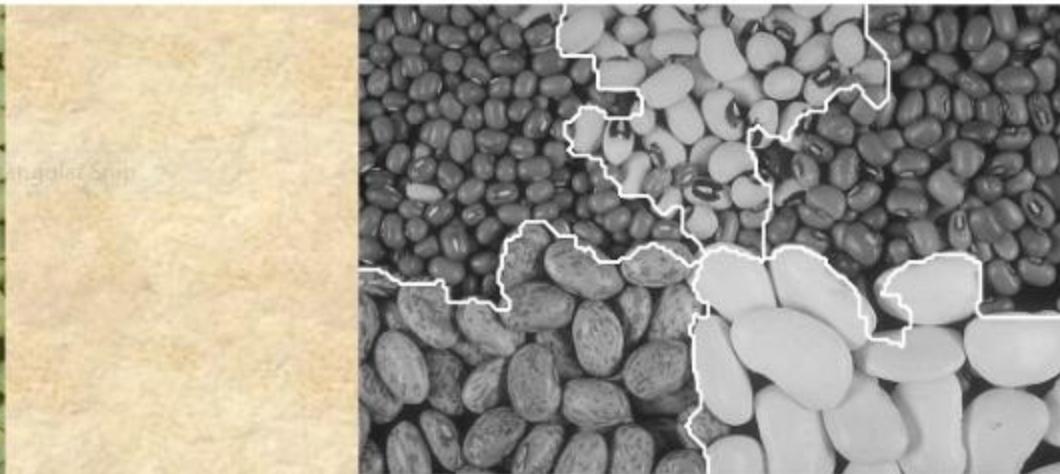
GrabCut Output



**SnakeCut
Output**

Object Extraction From an Image





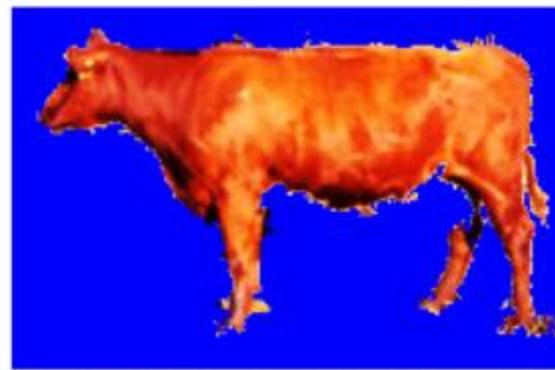


Image

Rectangular Snip

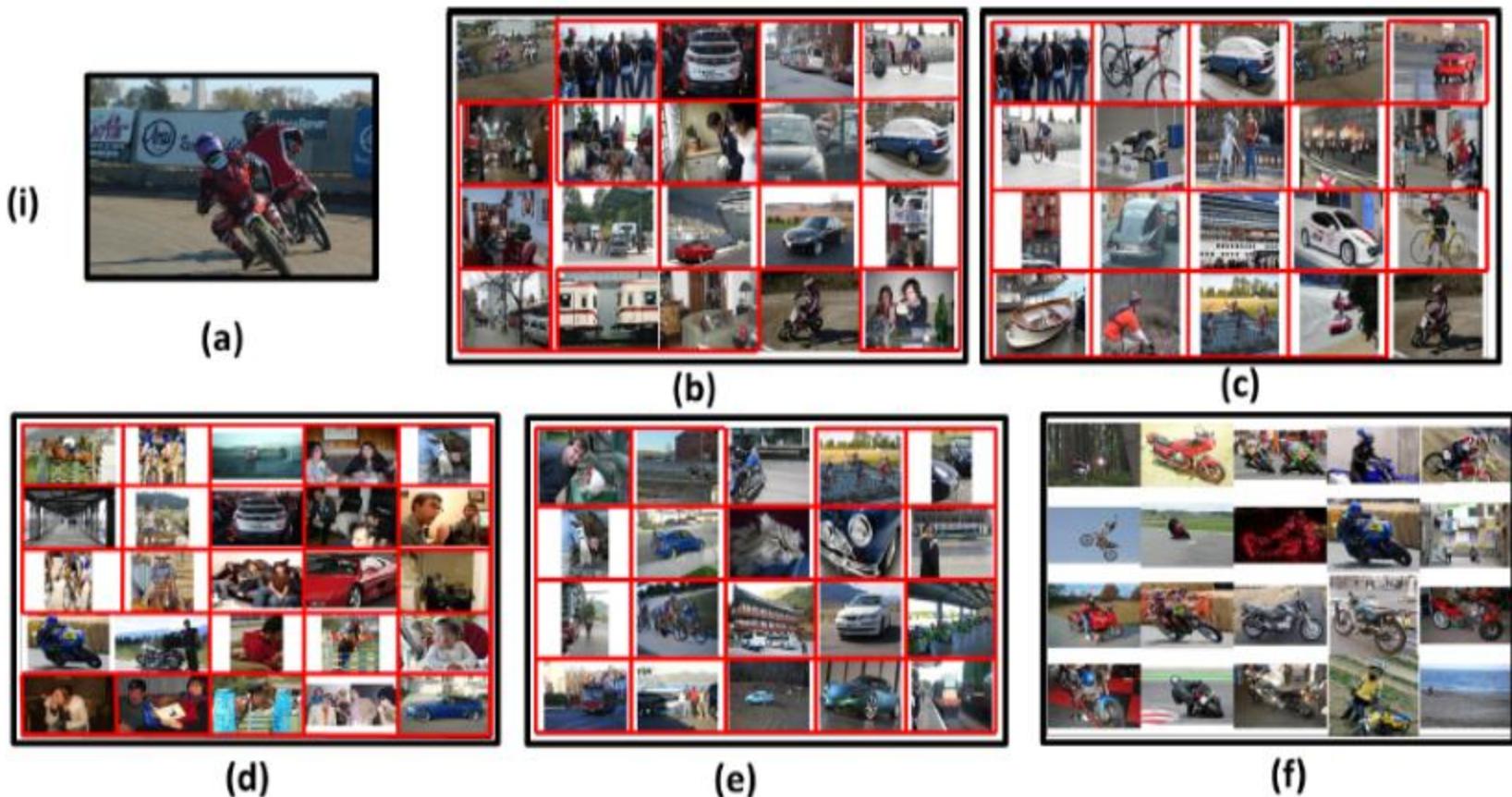


Segmentation



**Object Detection or segmentation –
involves object Detection and recognition modules**

Smart CBIR - Retrieval Results



Results of top 20 image retrievals (arranged in row-major order) shown for visual comparative study, using: (a) query image from the PASCAL datasets; (b) MTH (2010); (c) MSD (2011), (d) SLAR (2012); (e) CDH (2013); and (f) our proposed RADAR framework. Erroneous results are highlighted using a red template

Visual data on the Internet

- Flickr
 - 10+ billion photographs
 - 60 million images uploaded a month
- Facebook
 - 250 billion+
 - 300 million a day
- Instagram
 - 55 million a day
- YouTube
 - 100 hours uploaded every minute

 **90%** of net traffic will be visual!

Mostly about cats



Vision is Really Hard

- Vision is an amazing feature of natural intelligence
 - Visual cortex occupies about 50% of Macaque brain
 - More human brain devoted to vision than anything else



Why is Computer Vision Hard?



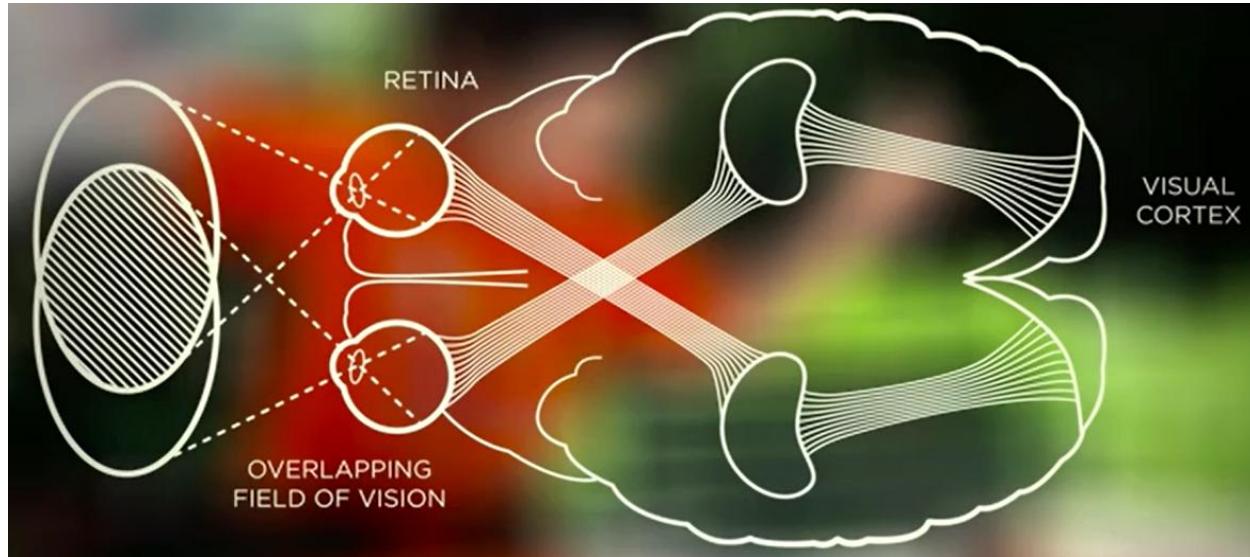
What did you see?

- Where this picture was taken?
- How many people are there?
- What are they doing?
- What object the person on the left standing on?
- Why this is a funny picture?

FEI FEI LI'S CONCEPT OF IMAGENET

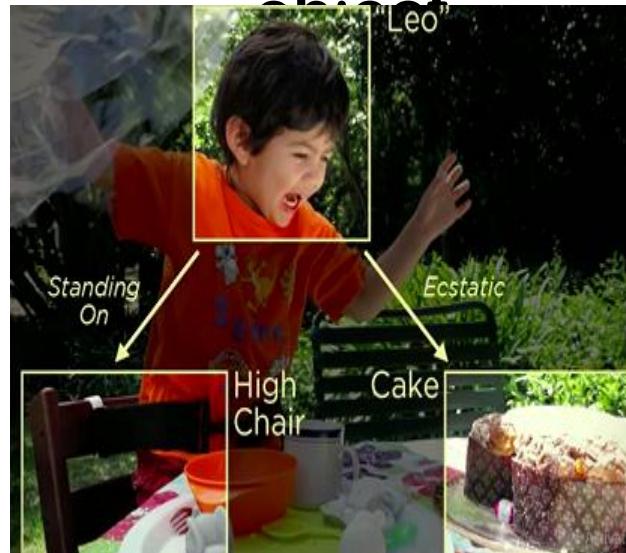
"If We Want Machines to Think, We Need to Teach Them to See"

Vision begins with the eyes, but truly takes place in the brain.

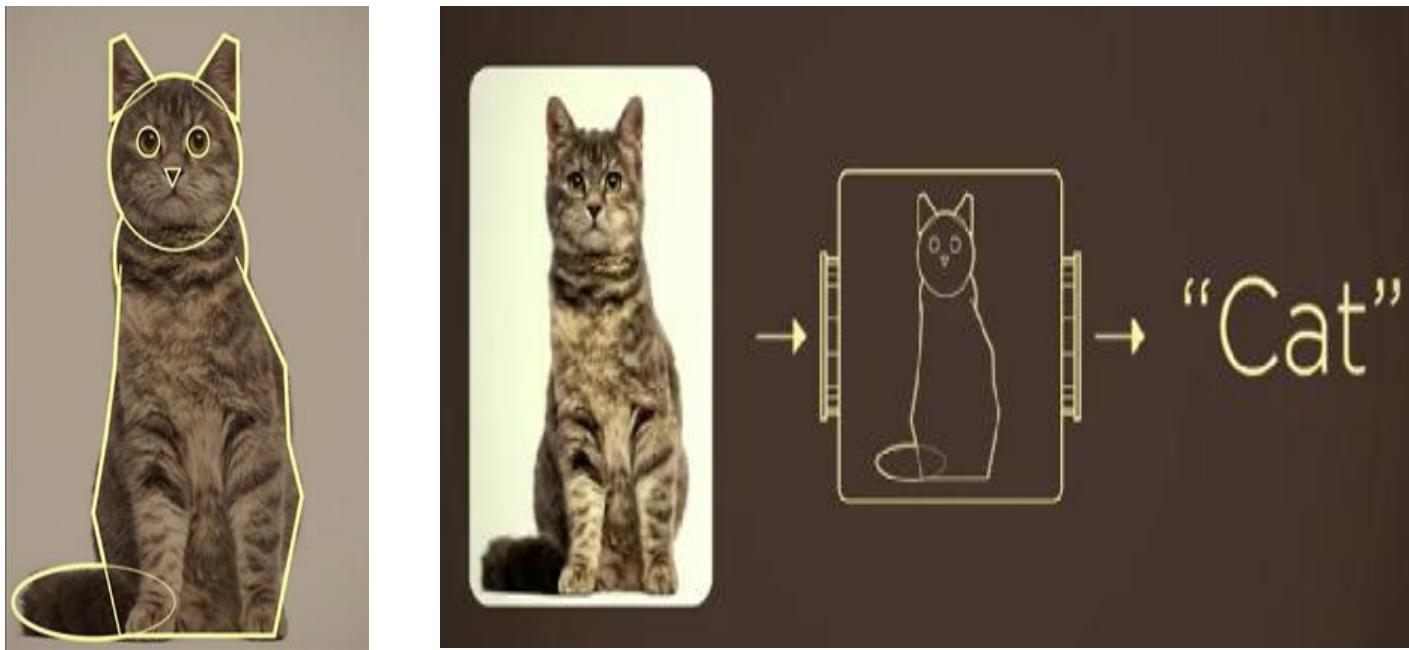


AIM OF IMAGENET

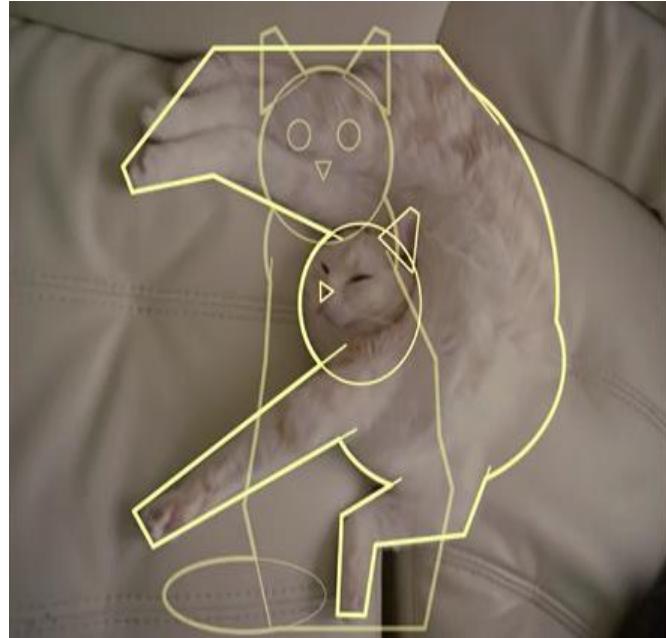
- To train the machine to recognize the input object



- To train the objects a mathematical model is used identifying the face as a circular object with circular eyes and body structure related to cat.



- But if this happens... then model will fail....



- This gave rise to a large classified data set called IMAGENET



11,231,732 images, 15589 synsets indexed

[Explore](#) New! [Download](#) New! [Challenge](#) [People](#) [Publication](#) [About](#)

Not logged in. [Login](#) | [Signup](#)

ImageNet is an image database organized according to the [WordNet](#) hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

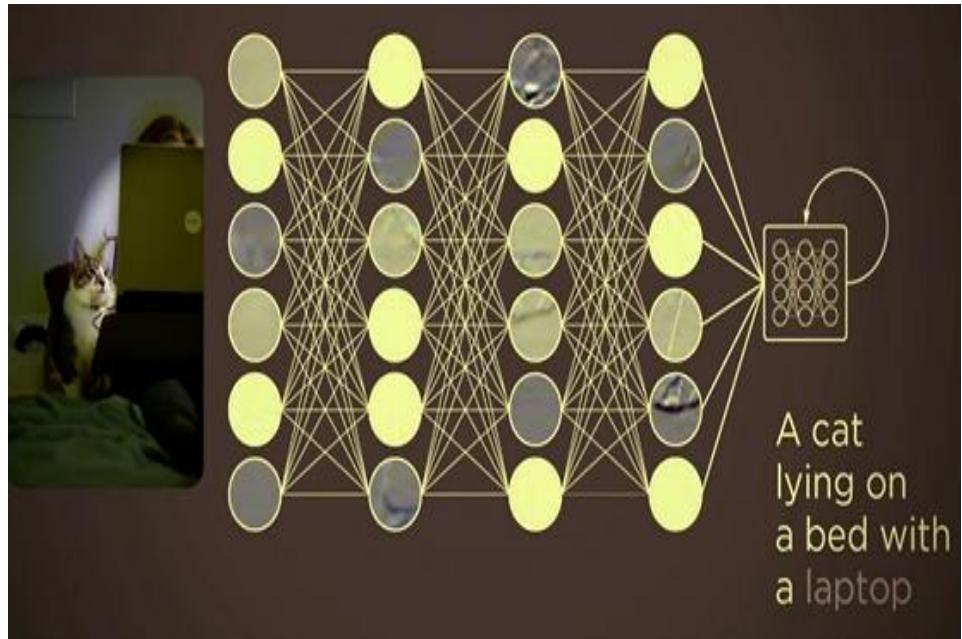
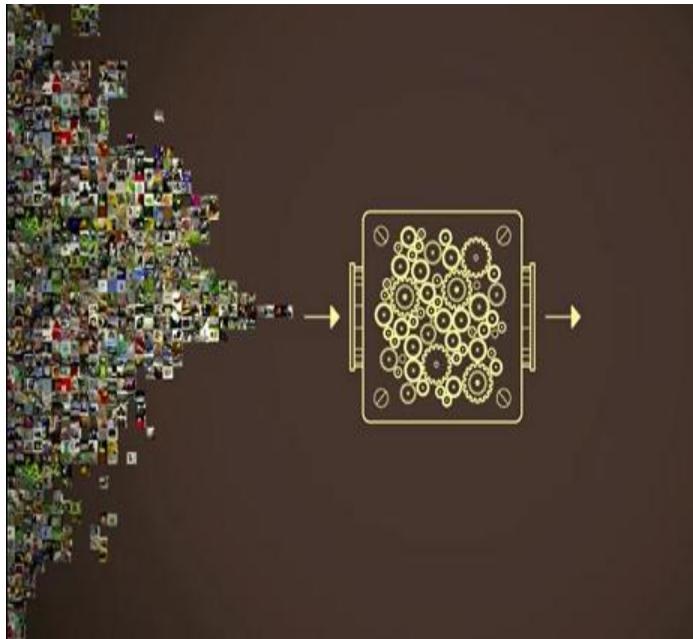
[Click here](#) to learn more about ImageNet, [Click here](#) to join the ImageNet mailing list.

What do these images have in common? *Find out!*

ImageNet 2010 Spring Release is up! [Click here](#) to check out what's new!

IMAGENET + MACHINE + CNN ALGO = CORRECT RESULT



Some more results of IMAGENET so obtained are:



- But it still fails a no. of times... so there are chances of improvement yet....

Animal



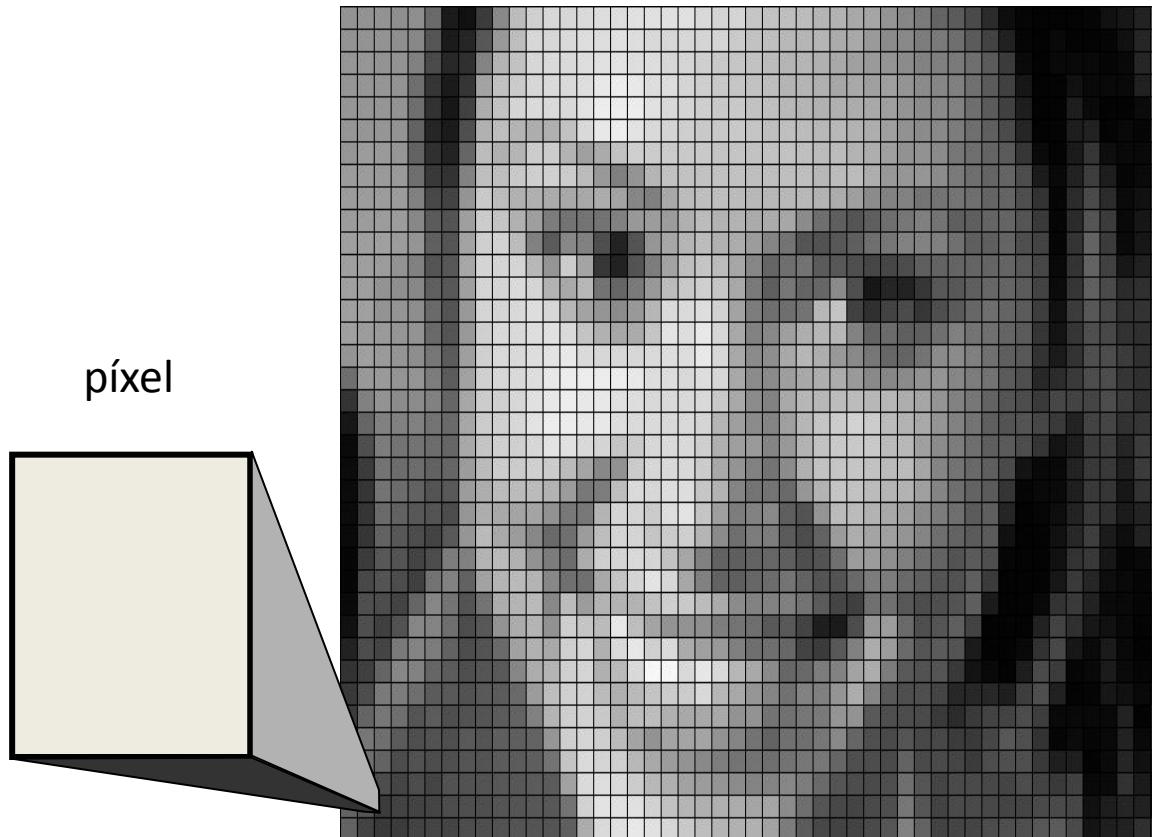
Digital Representation of Images

- A digital image is a matrix or two-dimensional array of numbers.
- Each cell in the matrix represents a pixel. Sample: Image of 20x15.

90	67	68	75	78	98	185	180	153	139	132	106	70	80	81	69	69	67	35	34
92	87	73	78	82	132	180	152	134	120	102	106	95	75	72	63	75	42	19	29
63	102	89	76	98	163	166	164	175	159	120	103	132	96	68	42	49	46	17	22
45	83	109	80	130	158	166	174	158	134	105	71	82	121	80	51	12	50	31	17
39	69	92	115	154	122	144	173	155	105	98	86	82	106	83	76	17	29	41	19
34	80	73	132	144	110	142	181	173	122	100	88	141	142	111	87	33	18	46	36
37	93	88	136	171	164	137	171	190	149	110	137	168	161	132	96	56	23	48	49
66	117	106	147	188	202	198	187	187	159	124	151	167	158	138	105	80	55	59	54
127	136	107	144	188	197	188	184	192	172	124	151	138	108	116	114	84	46	67	54
143	134	99	143	188	172	129	127	179	167	106	118	111	54	70	95	90	46	69	52
141	137	96	146	167	123	91	90	151	156	121	93	78	82	97	91	87	45	66	39
139	137	80	131	162	145	131	129	154	161	158	149	134	122	115	99	84	35	52	30
137	133	56	104	165	167	174	181	175	169	165	162	158	142	124	103	67	19	31	23
135	132	65	86	173	186	200	198	181	171	162	153	145	135	121	104	53	14	15	33
132	132	88	50	149	182	189	191	186	178	166	157	148	131	106	78	28	10	15	44

Digital Representation of Images

- A more common way to display an image ...



The Three Stages of Computer Vision

- low-level

image —————→ image

- mid-level

image —————→ features

- high-level

features —————→ analysis

Low-Level

sharpening



blurring

Low-Level



original image

Canny



edge image

Mid-Level



edge image

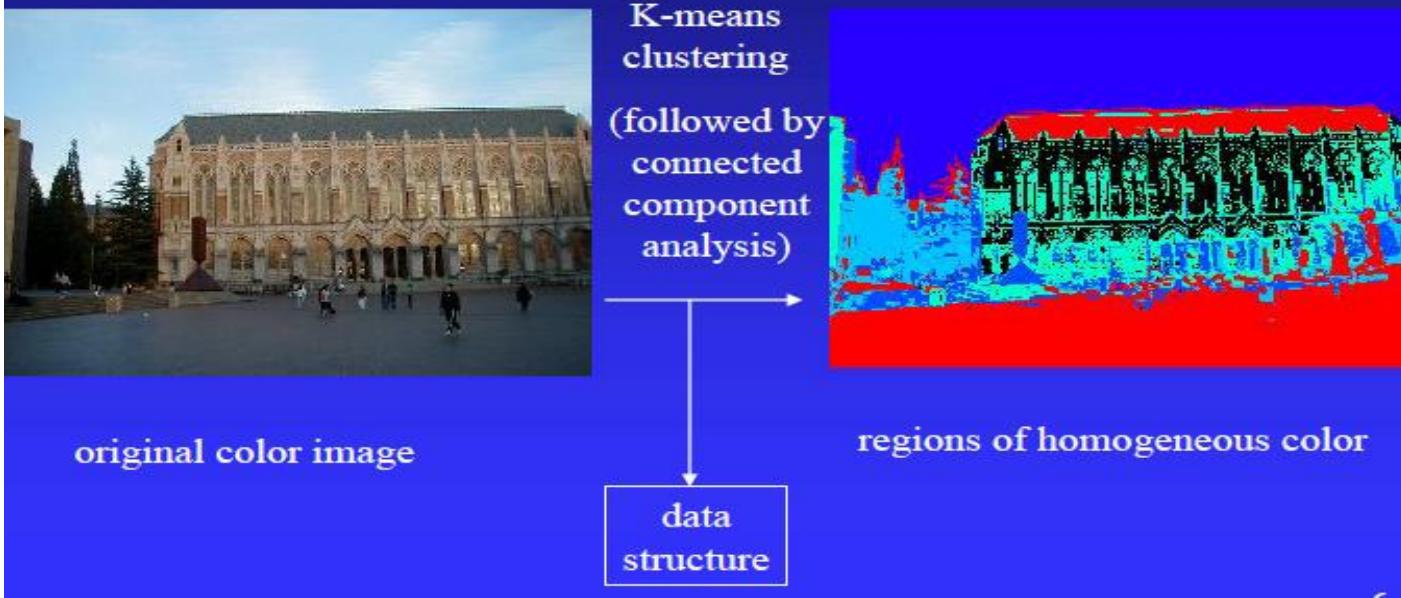
ORT

↓
data
structure



circular arcs and line segments

Mid-level



Low- to High-Level



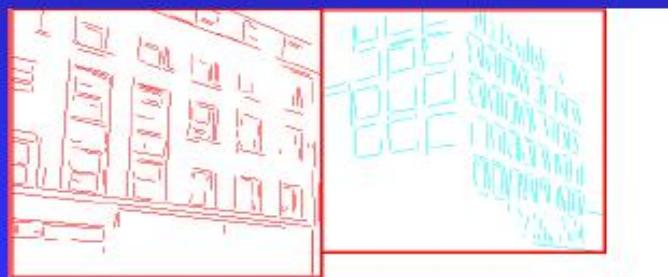
low-level

edge image

mid-level

consistent
line clusters

high-level



Building Recognition

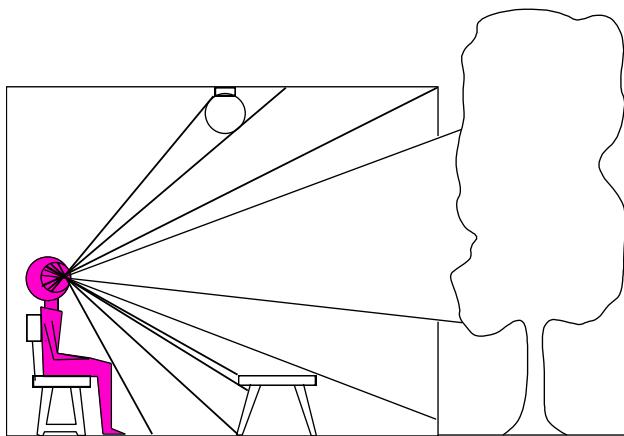


Image

- 2-D array of numbers (intensity values, gray levels)
 - Gray levels 0 (black) to 255 (white)
 - Color image is 3 2-D arrays of numbers
 - Red
 - Green
 - Blue
 - Resolution (number of rows and columns)
 - 128X128
 - 256X256
 - 512X512

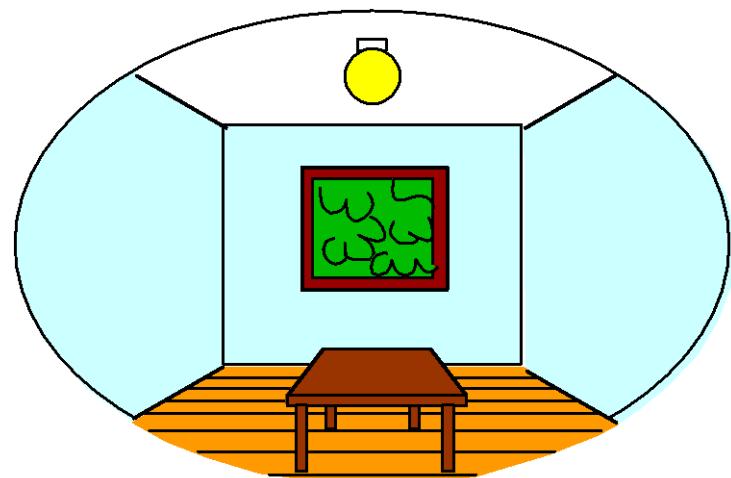
Image formation

3D world

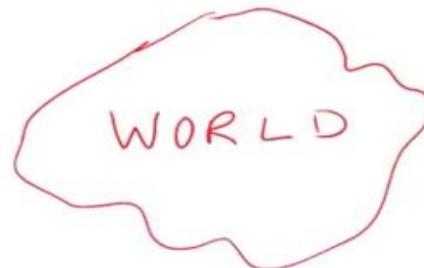


Point of observation

2D image



A camera creates an image ...



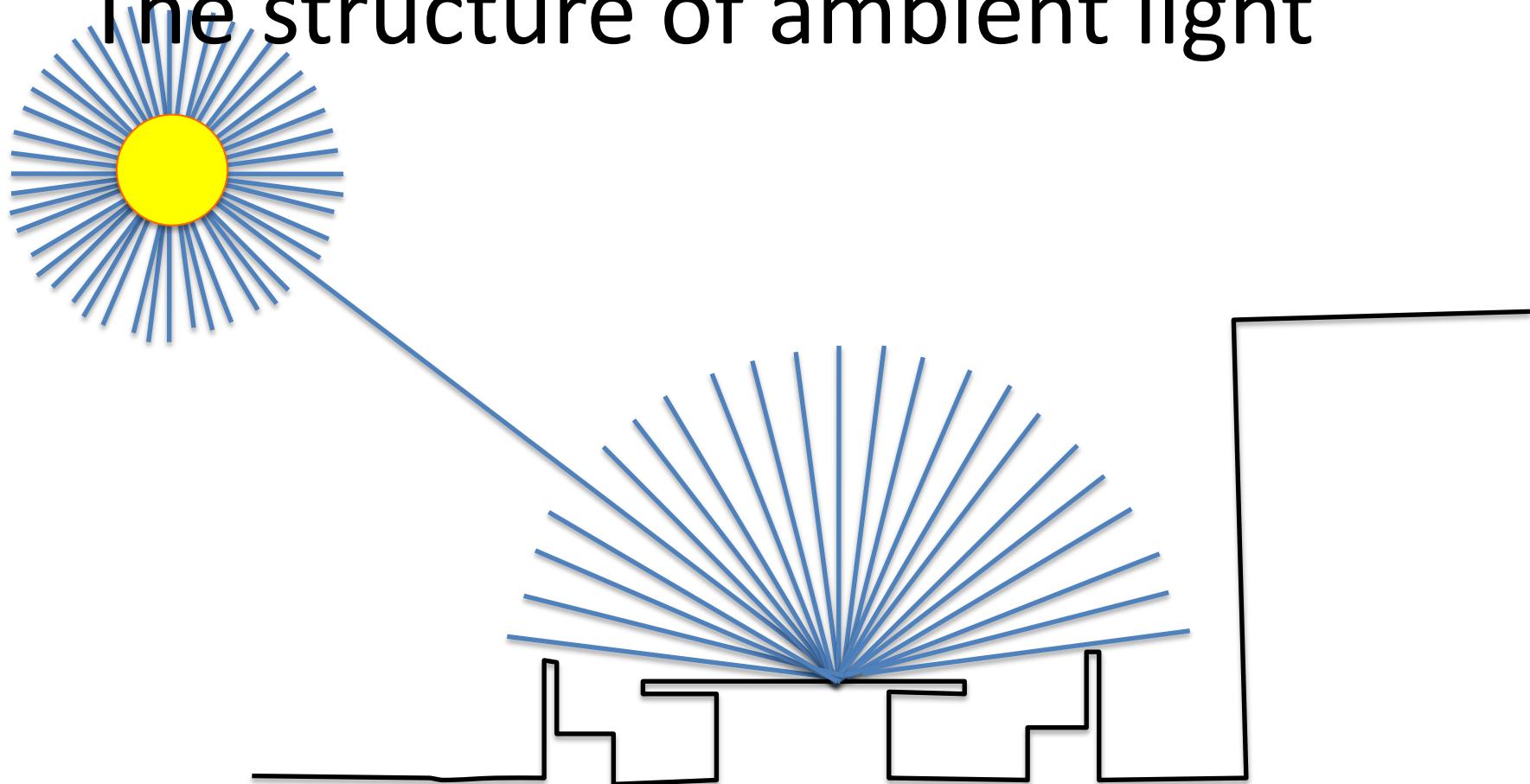
The image $I(x,y)$ measures how much light is captured at pixel (x,y)

Cameras, lenses, and calibration

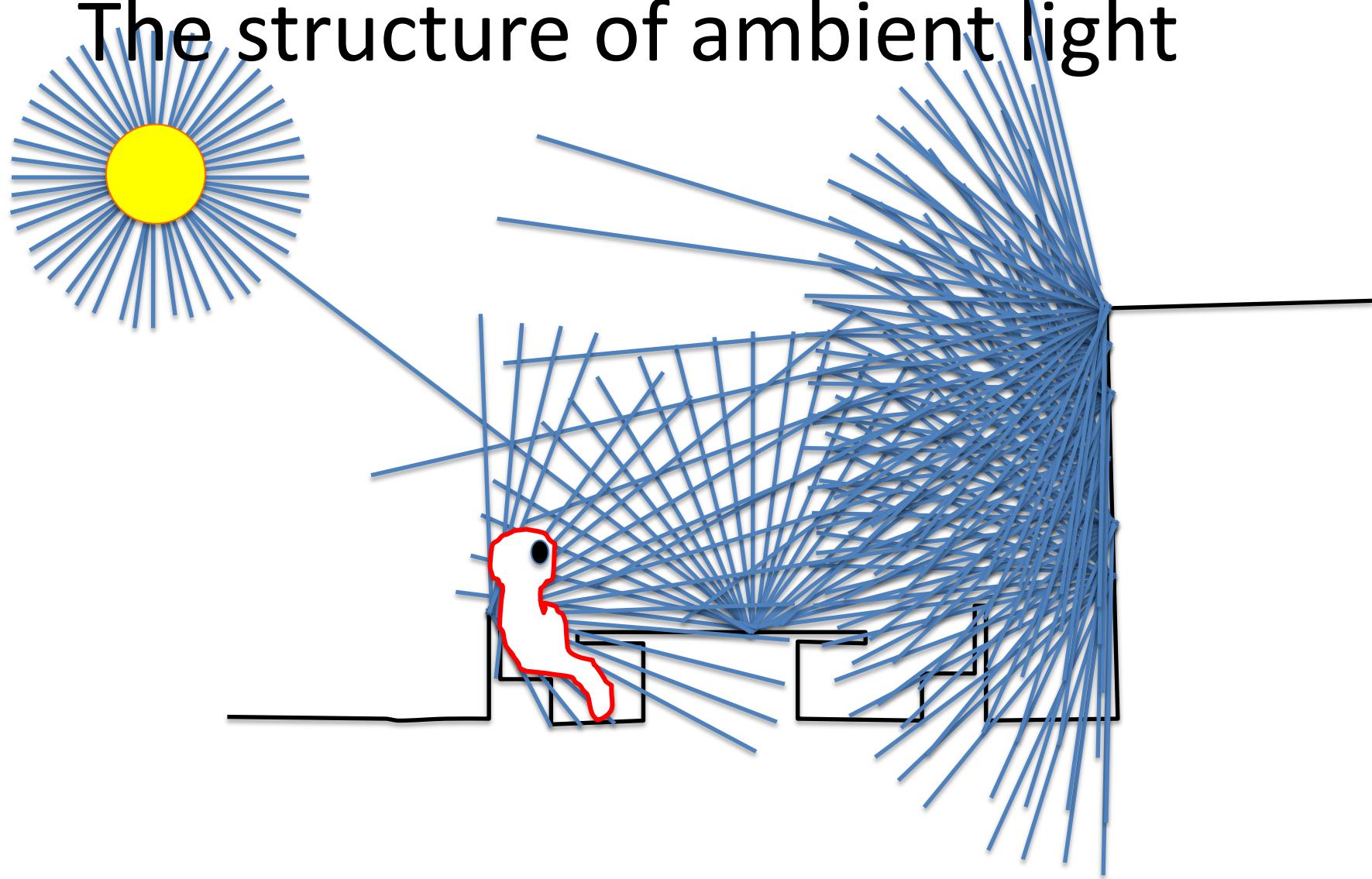
- Camera models
- Projection equations

Images are projections of the 3-D world onto a 2-D plane...

The structure of ambient light

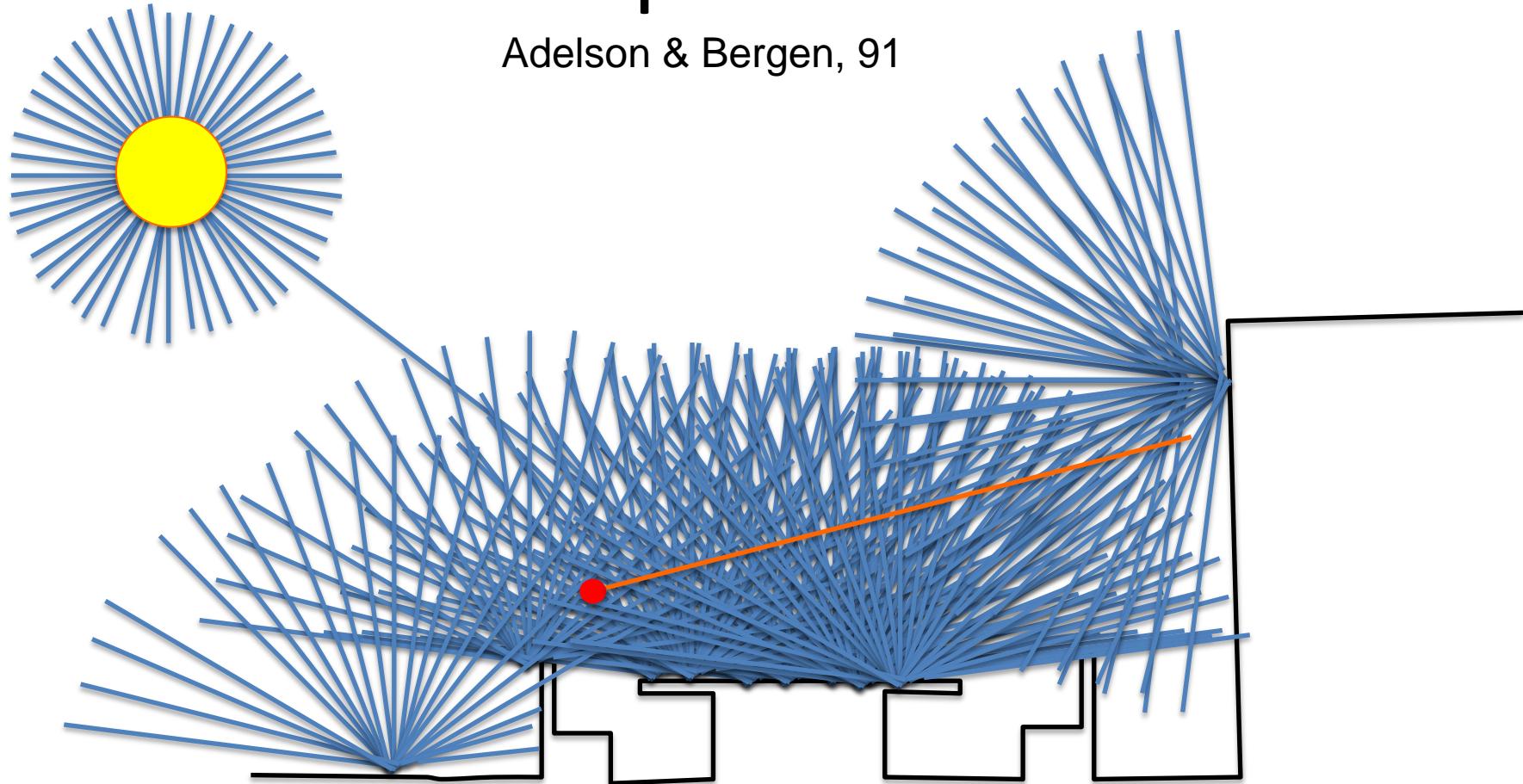


The structure of ambient light



The Plenoptic Function

Adelson & Bergen, 91

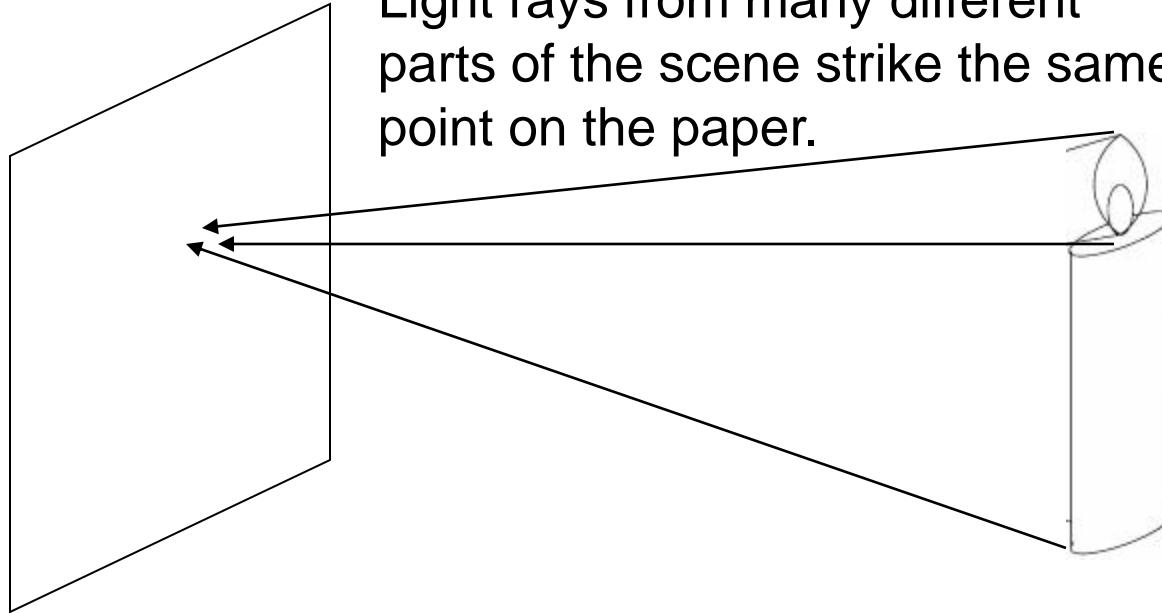


The intensity P can be parameterized as:

$$P(\theta, \phi, \lambda, t, X, Y, Z)$$

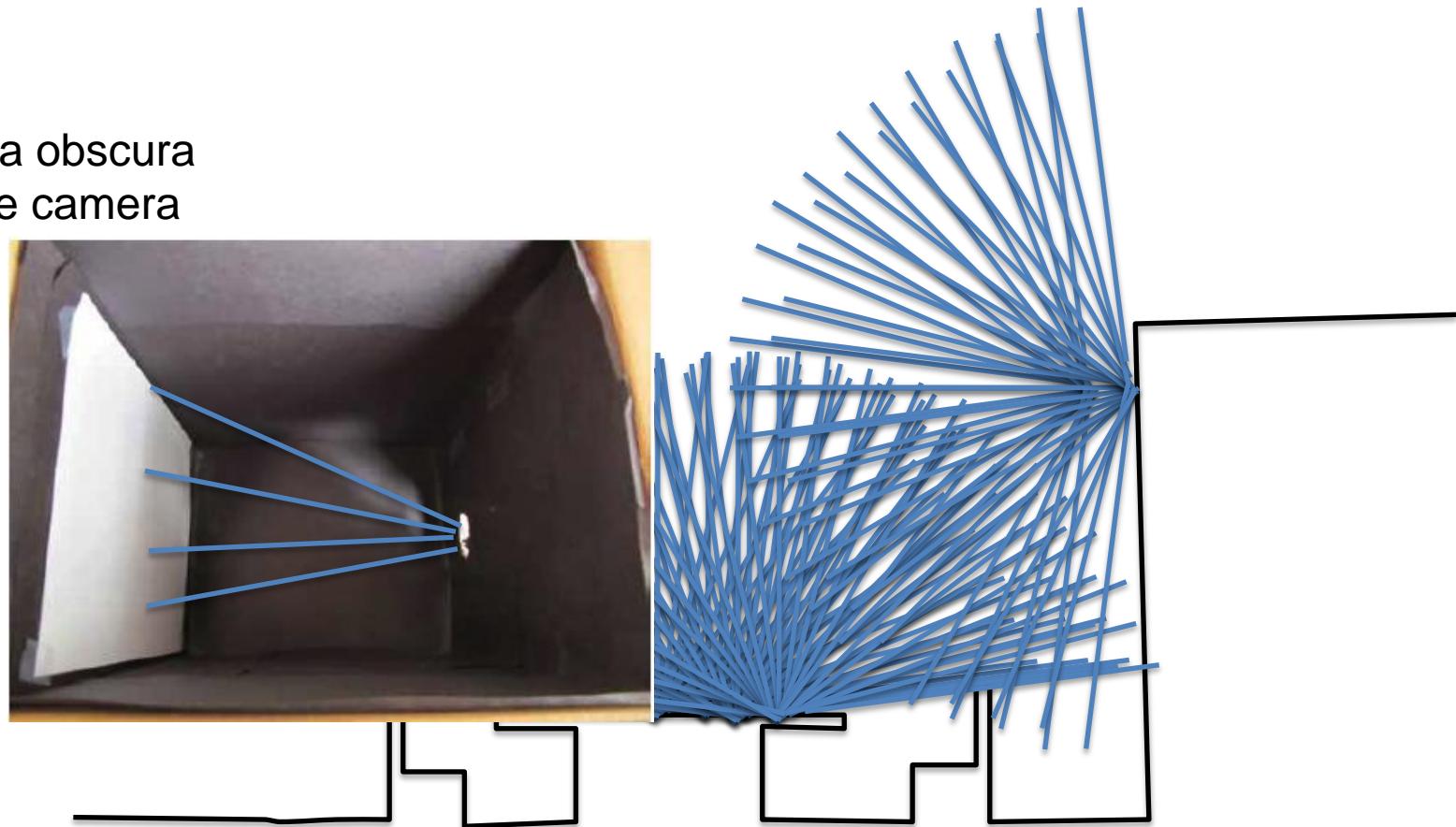
“The complete set of all convergence points constitutes the permanent possibilities of vision.” Gibson

Light rays from many different parts of the scene strike the same point on the paper.

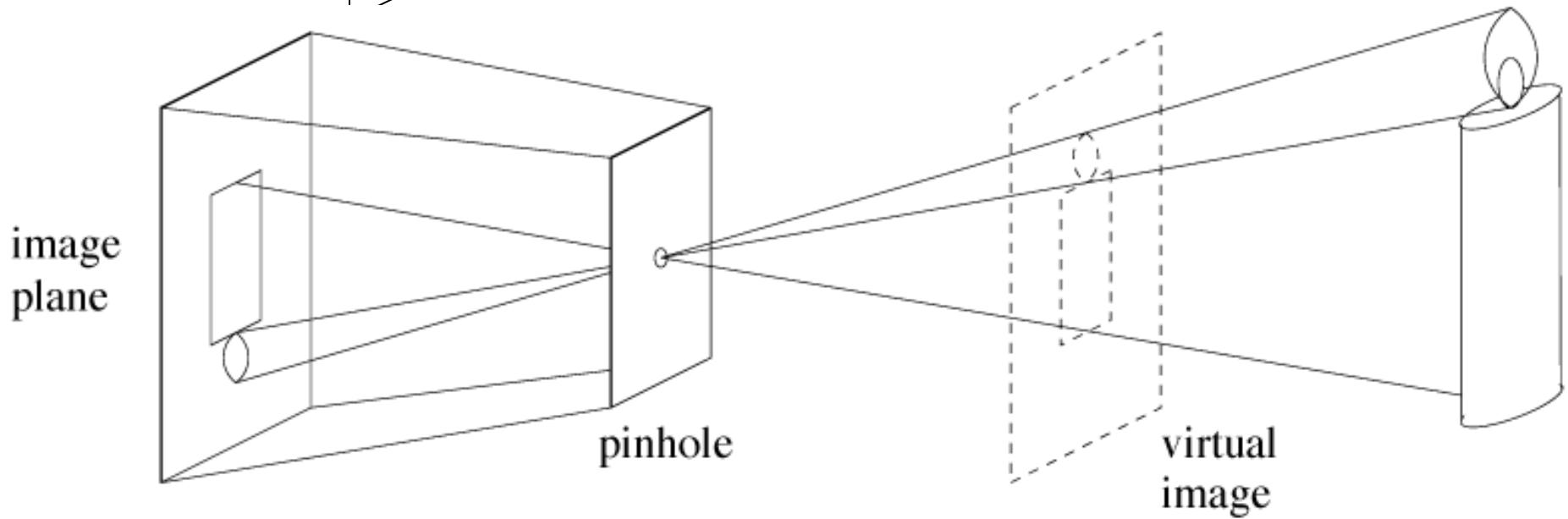
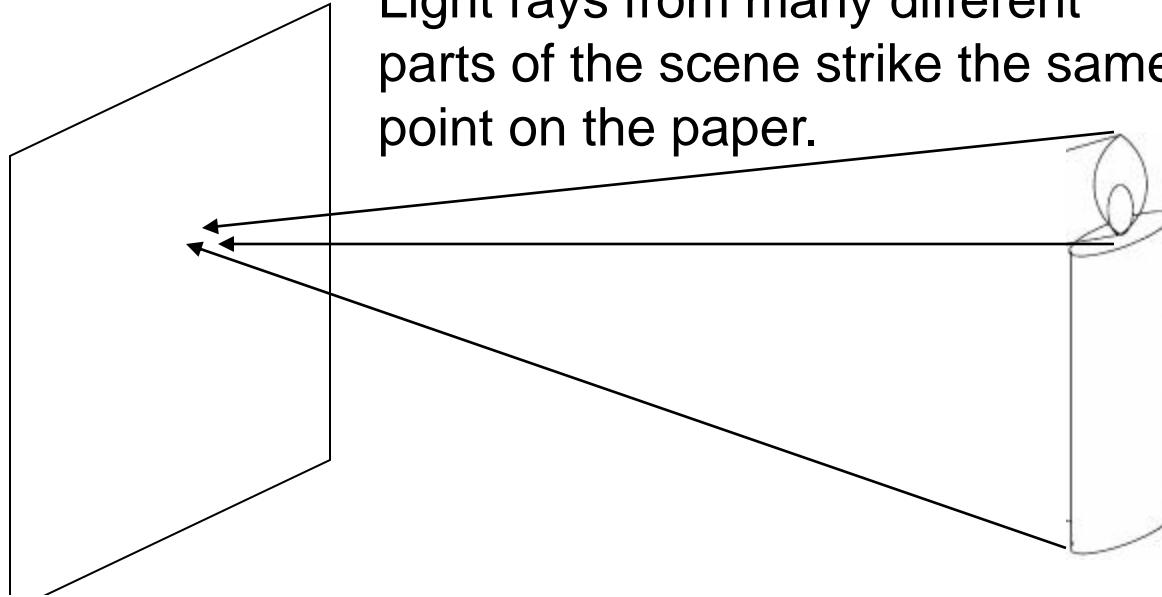


Measuring the Plenoptic function

The camera obscura
The pinhole camera



Light rays from many different parts of the scene strike the same point on the paper.



The pinhole camera only allows rays from one point in the scene to strike each point of the paper.

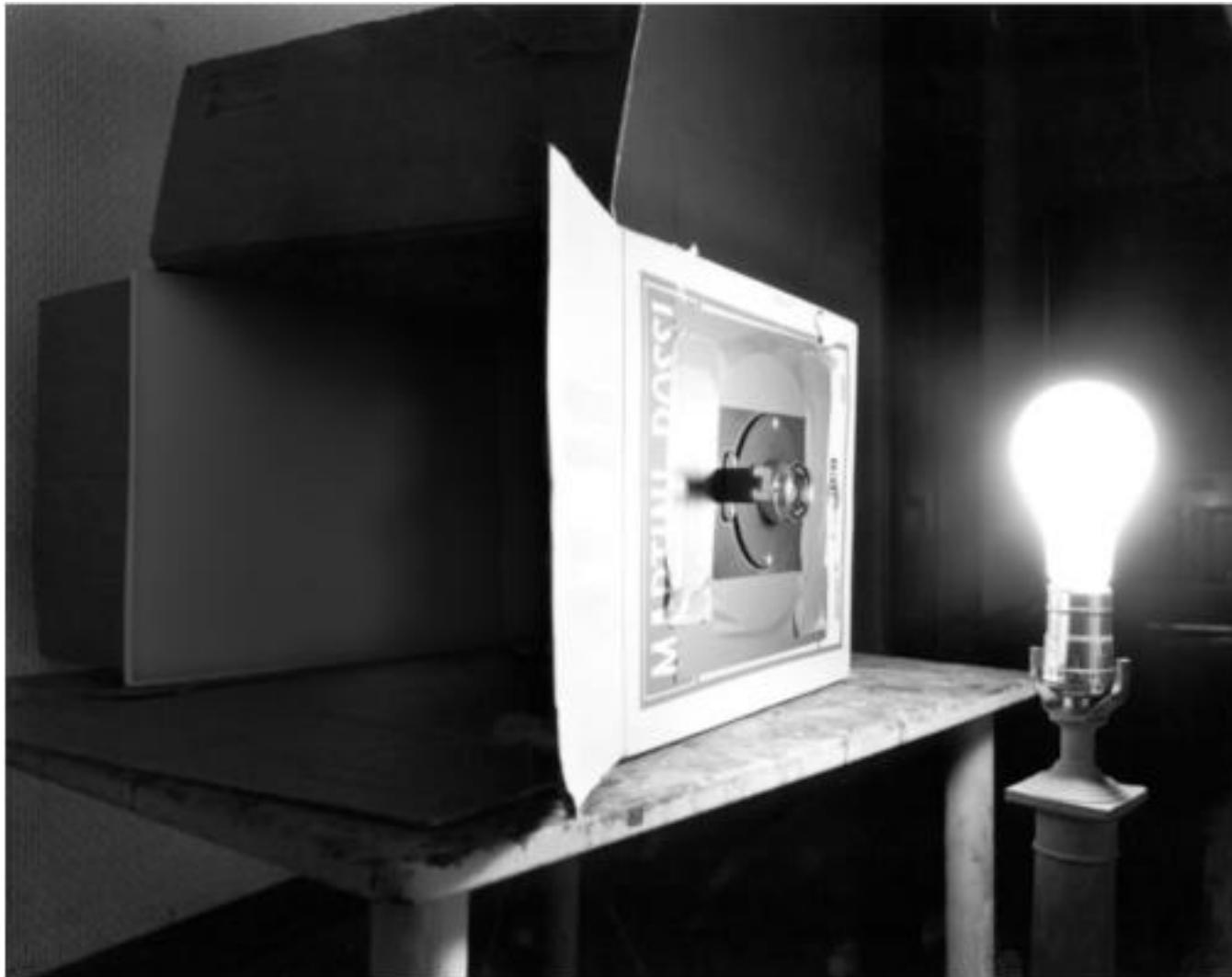
Forsyth & Ponce

Pinhole camera



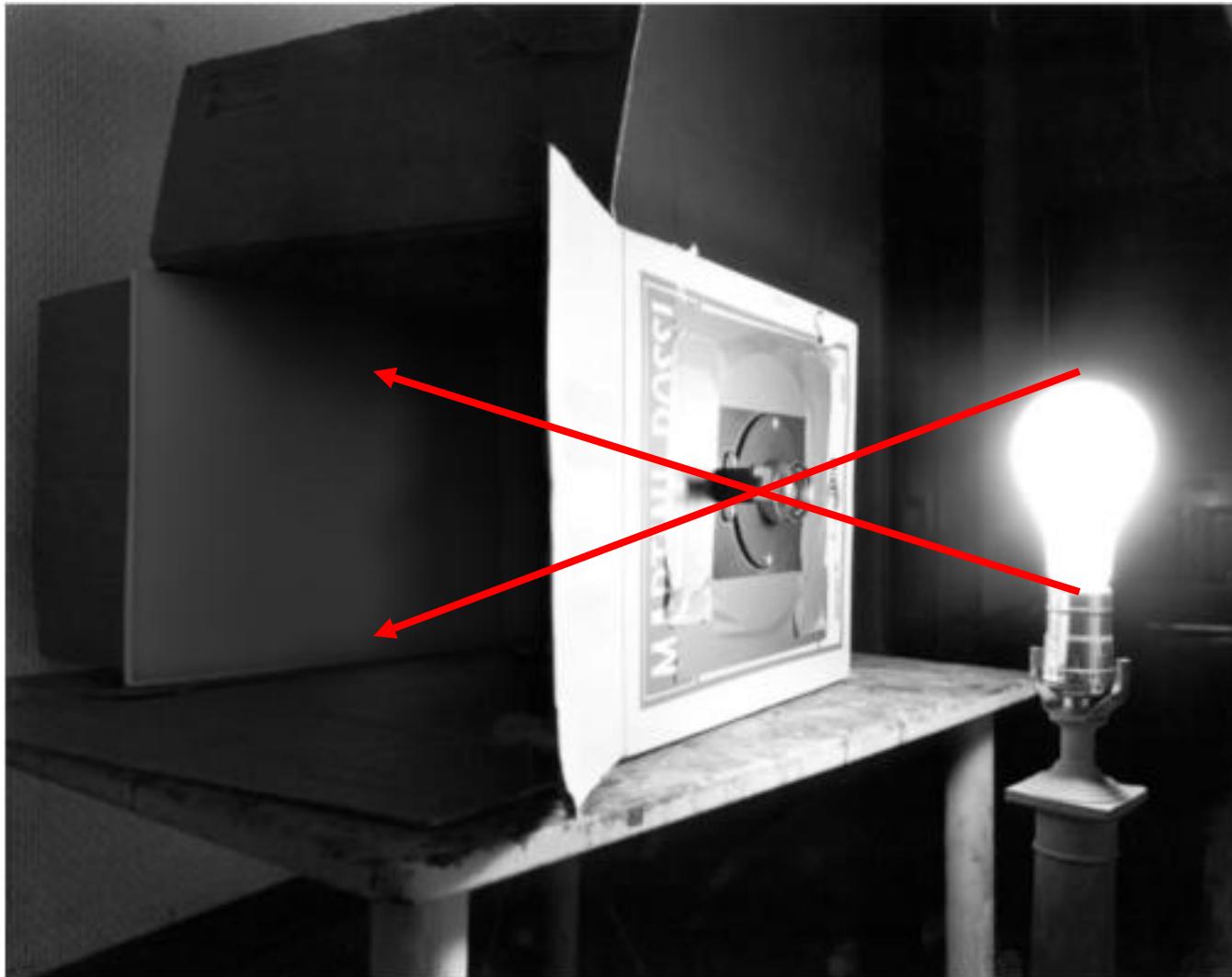
Photograph by Abelardo Morell, 1991

Pinhole camera



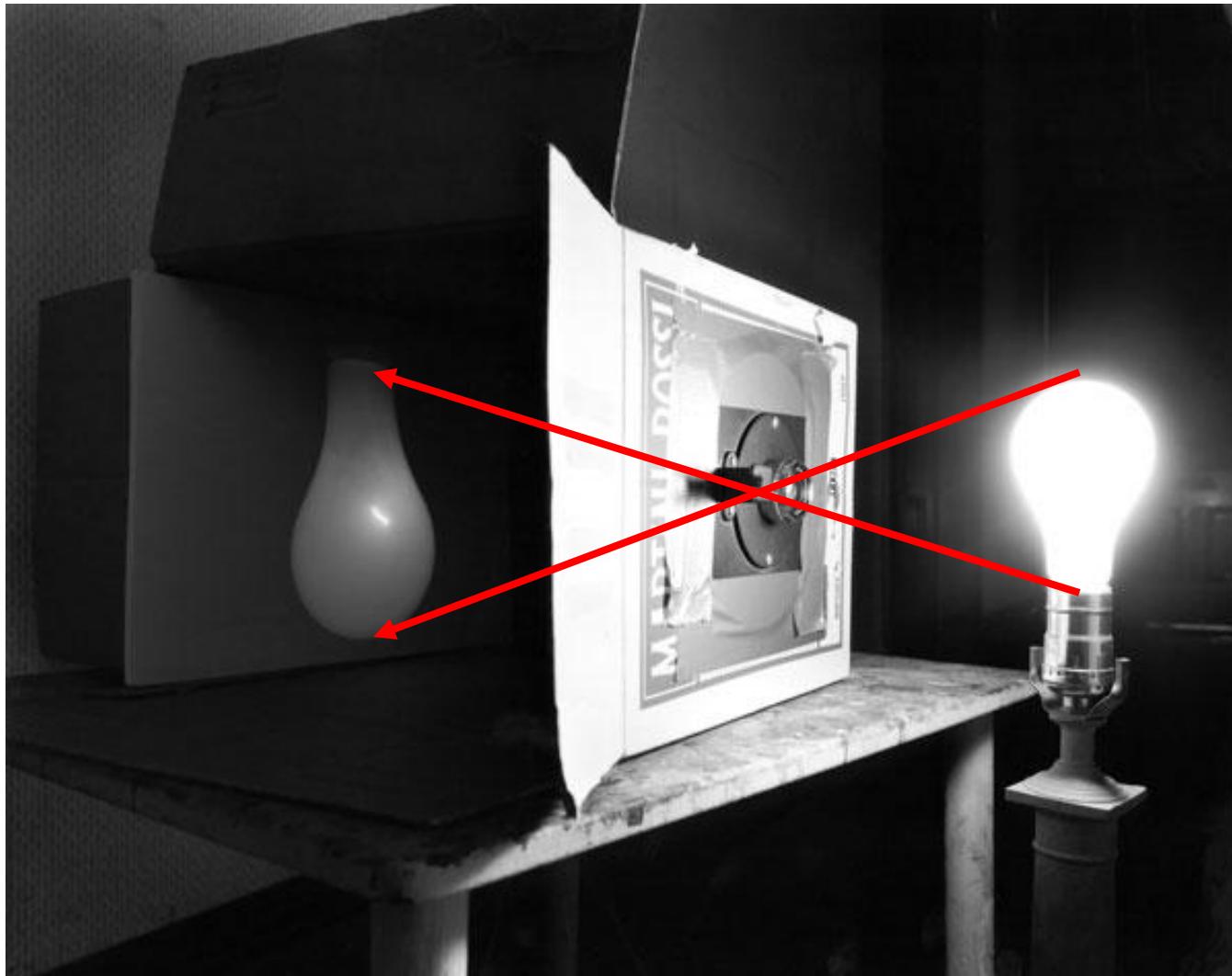
Photograph by Abelardo Morell, 1991

Pinhole camera



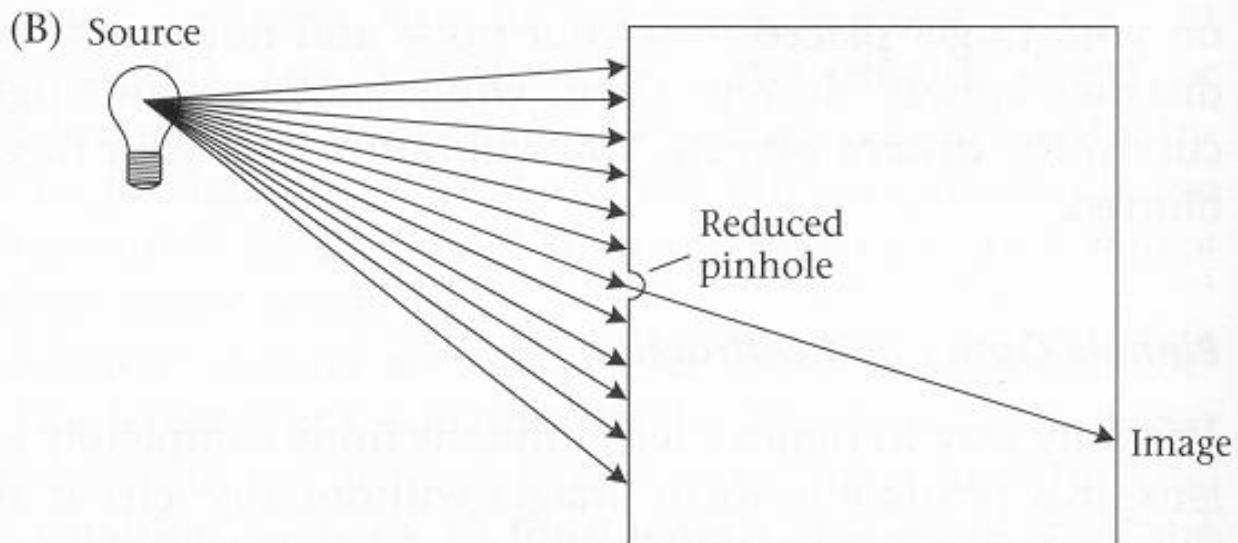
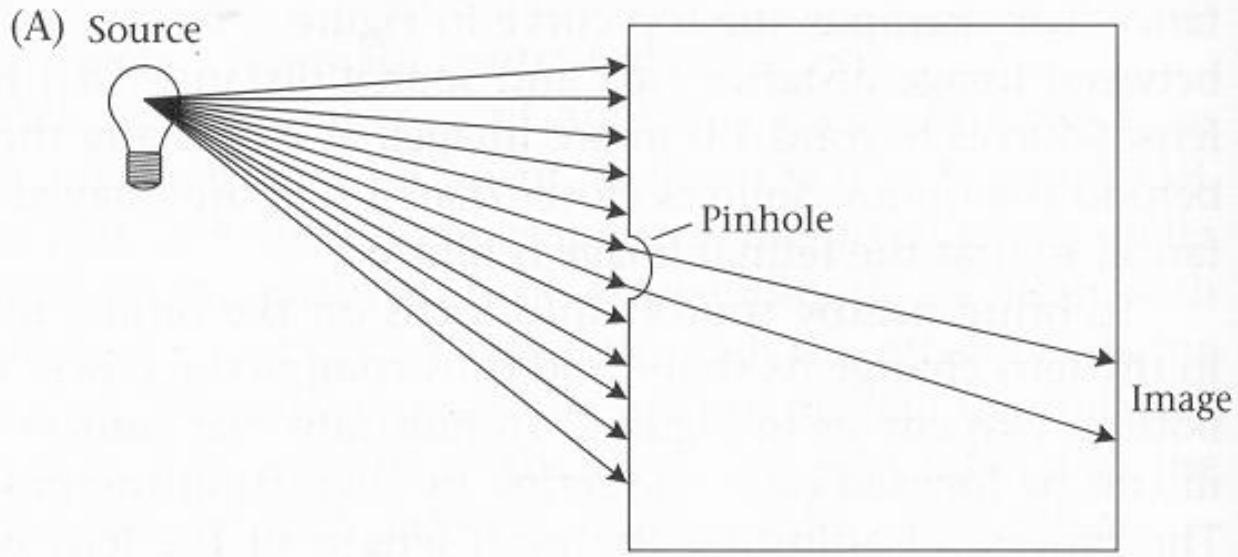
Photograph by Abelardo Morell, 1991

Pinhole camera

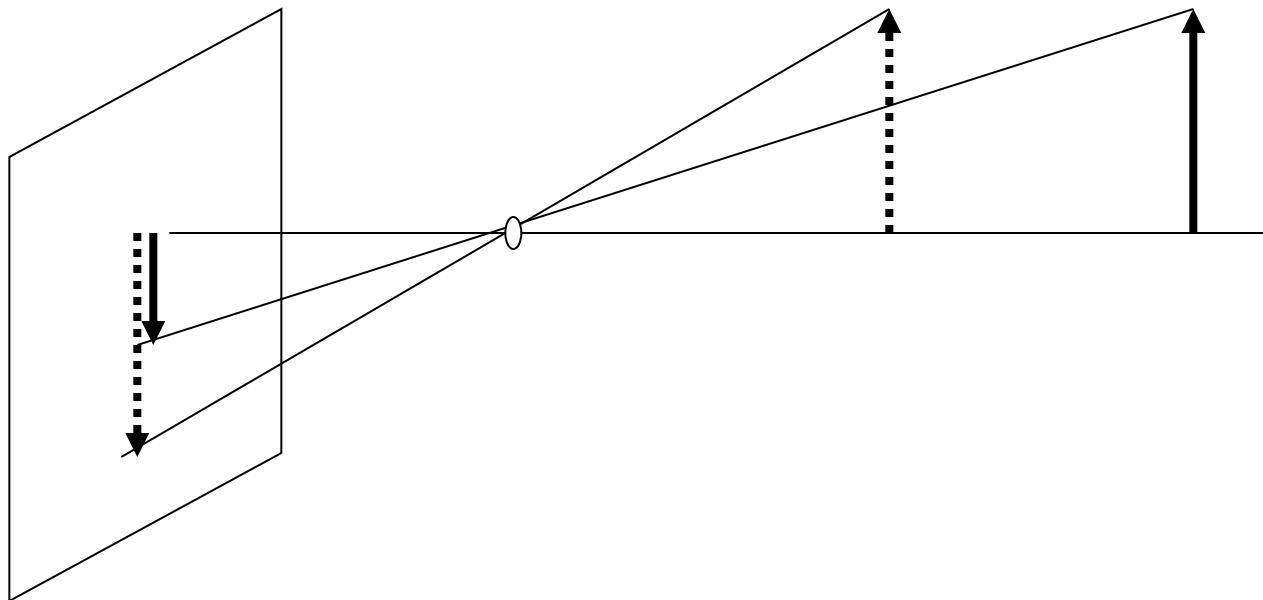


Photograph by Abelardo Morell, 1991

Effect of pinhole size

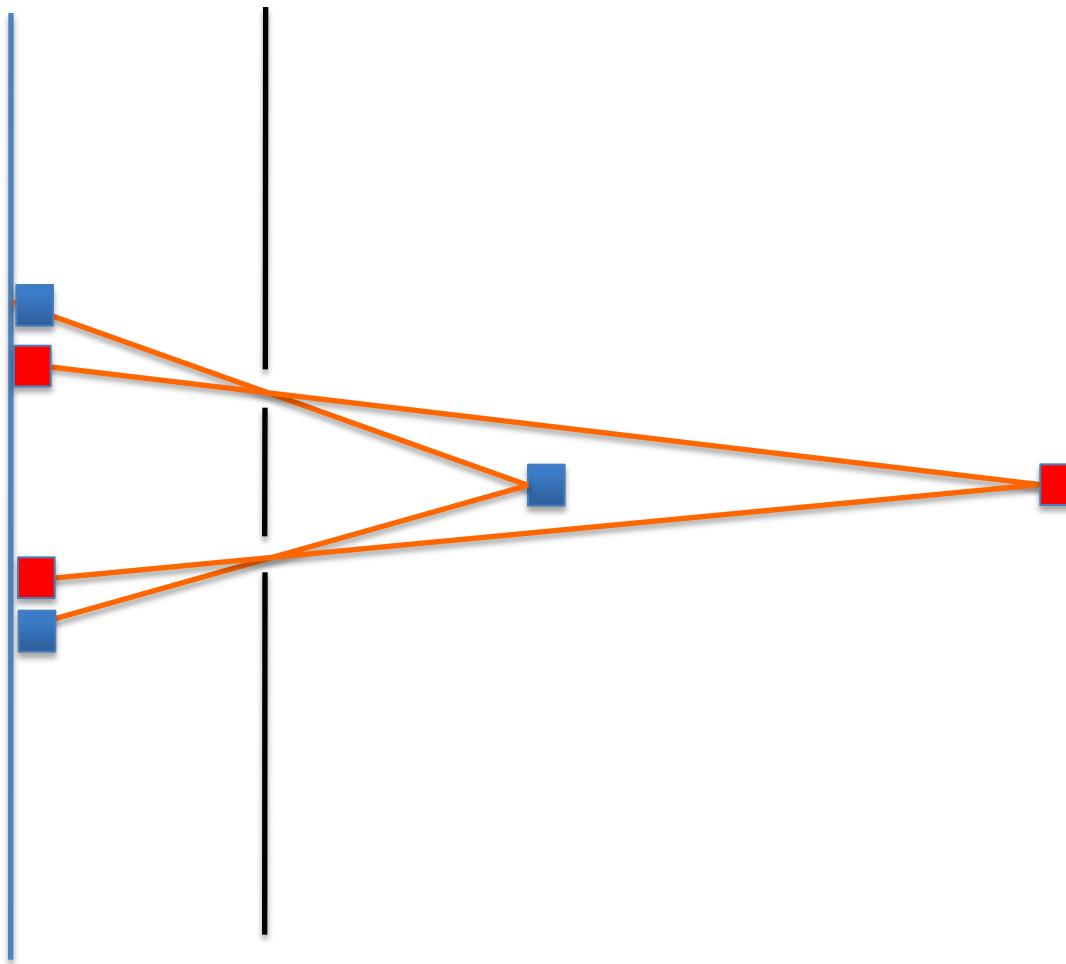


Measuring distance

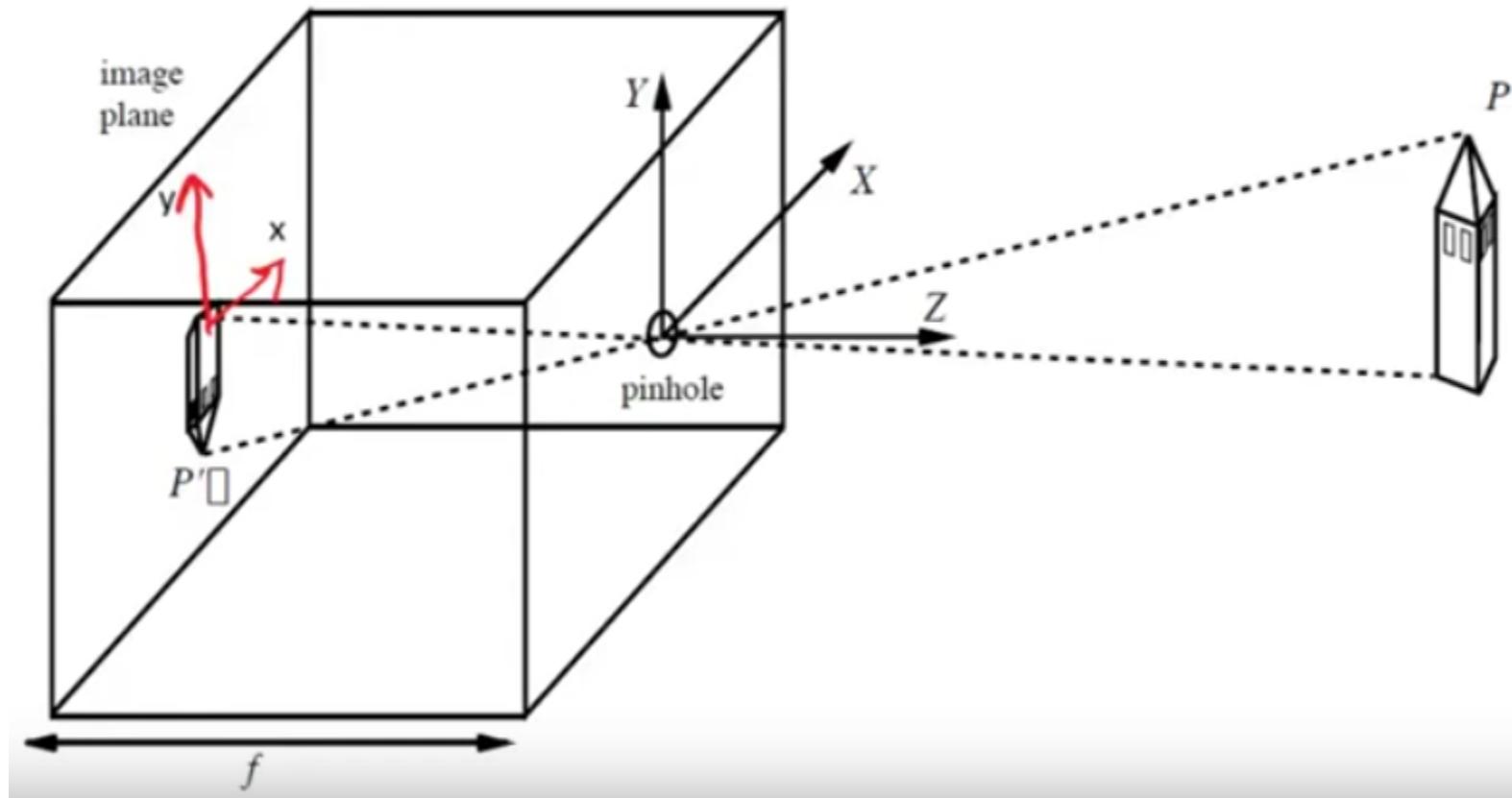


- Object size decreases with distance to the pinhole
- Therefore, given a single projection, if we know the size of the object we can know how far it is.
- But for objects of unknown size, the 3D information seems to be lost.

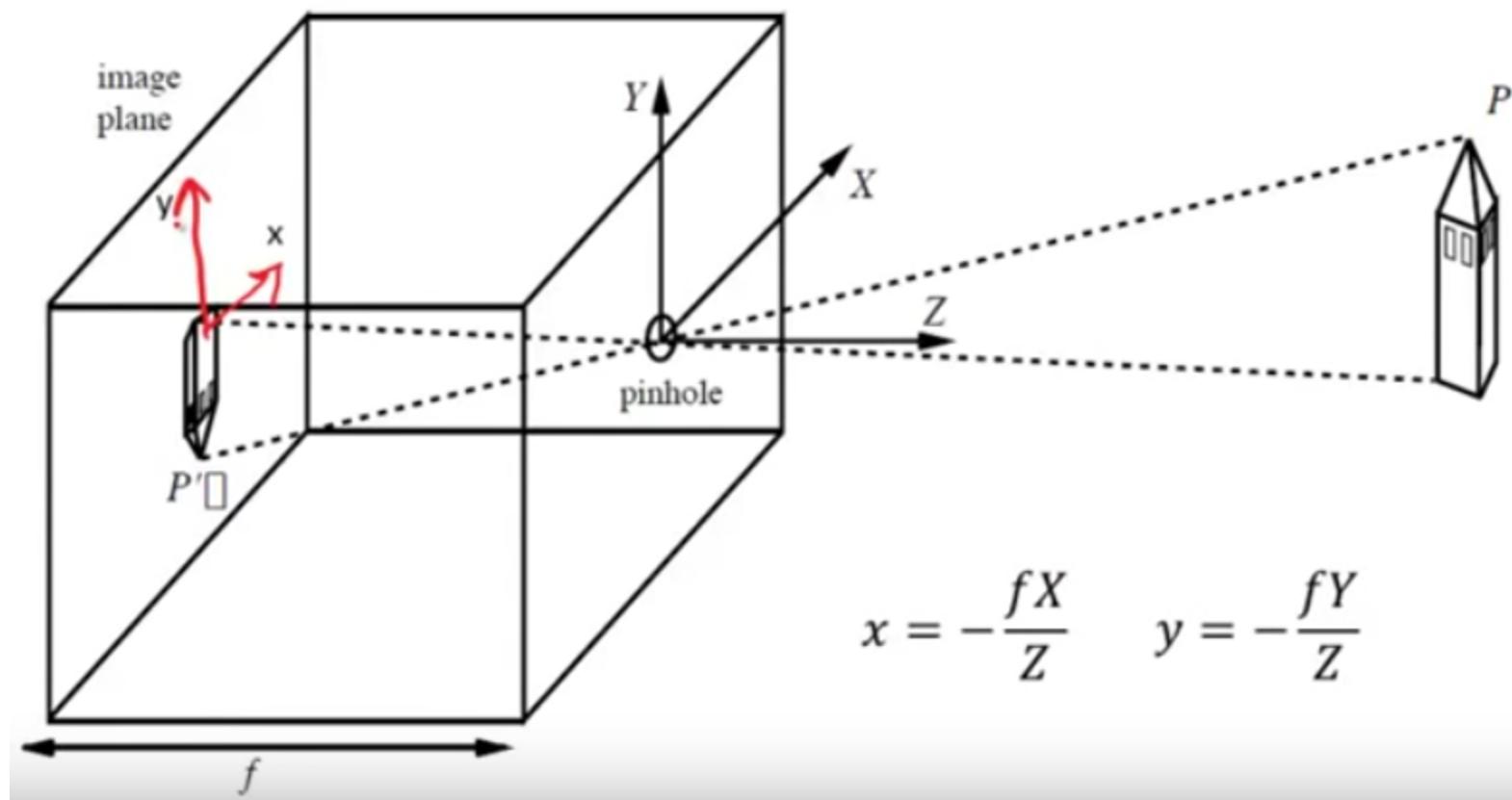
Two pinholes



The Pinhole Camera

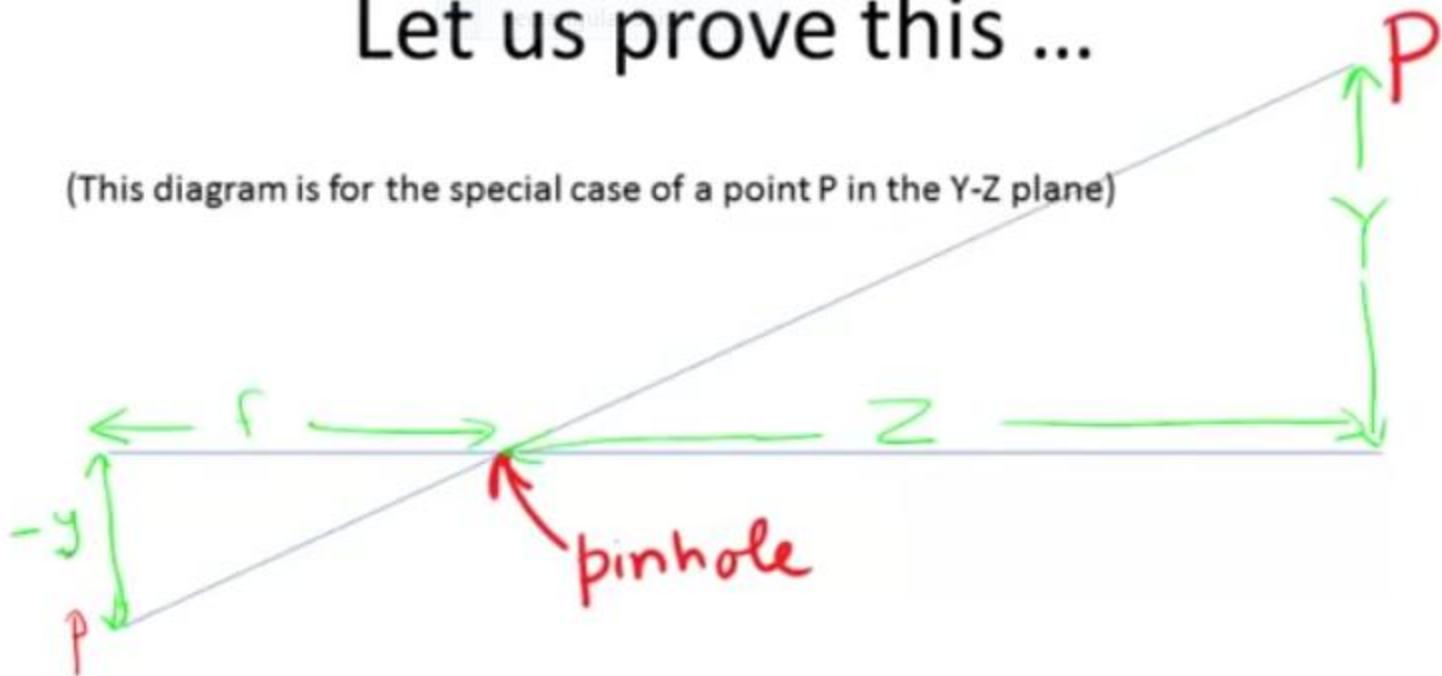


The Pinhole Camera



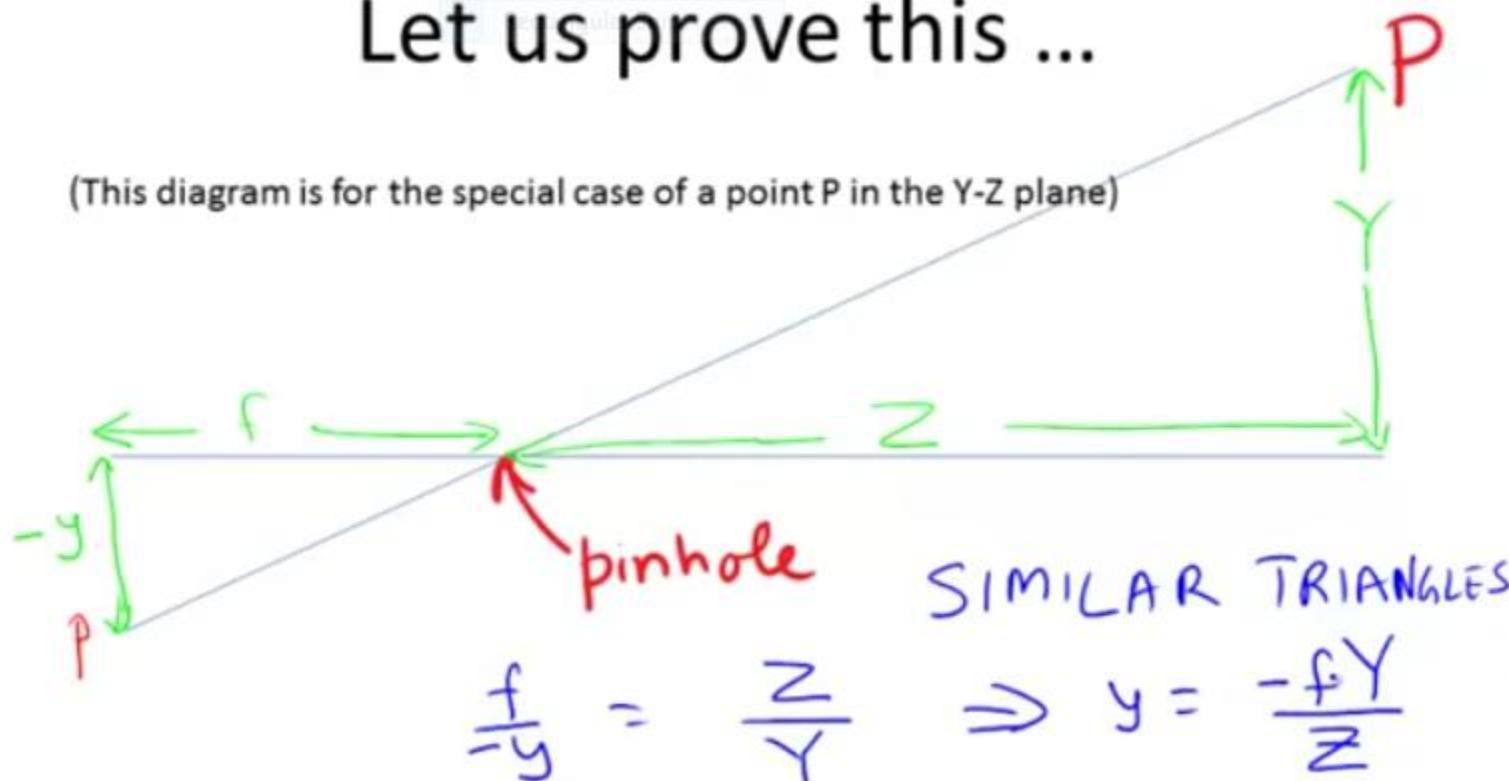
Let us prove this ...

(This diagram is for the special case of a point P in the Y-Z plane)



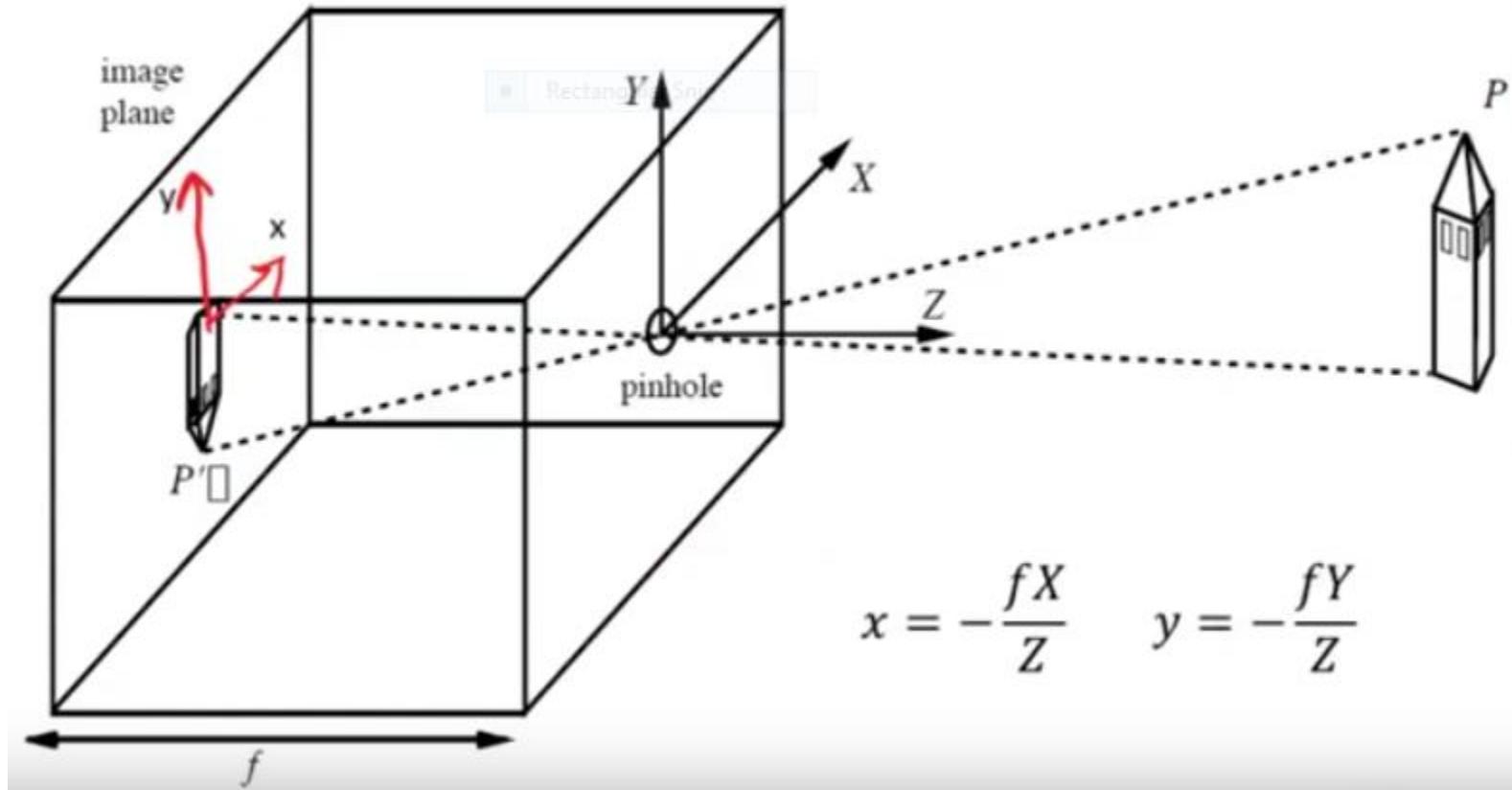
Let us prove this ...

(This diagram is for the special case of a point P in the Y-Z plane)



SIMILAR TRIANGLES

$$\frac{f}{-y} = \frac{z}{y} \Rightarrow y = -\frac{fy}{z}$$



Basic Relationships Between Pixels

- Neighborhood
- Adjacency
- Connectivity
- Paths
- Regions and boundaries

Neighbors of a Pixel

- Any pixel $p(x, y)$ has two vertical and two horizontal neighbors, given by
 $(x+1, y), (x-1, y), (x, y+1), (x, y-1)$
- This set of pixels are called the 4-neighbors of P, and is denoted by $N_4(P)$.
- Each of them are at a unit distance from P.

Neighbors of a Pixel

$$f_{\text{xx}} = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & f(0,3) & f(0,4) & \cdots \\ f(1,0) & f(1,1) & f(1,2) & f(1,3) & f(1,4) & \cdots \\ f(2,0) & f(2,1) & f(2,2) & f(2,3) & f(2,4) & \cdots \\ f(3,0) & f(3,1) & f(3,2) & f(3,3) & f(3,4) & \cdots \\ | & | & | & | & | & \cdots \\ | & | & | & | & | & \cdots \end{bmatrix}$$

□ A Pixel p at coordinates (x, y) has 4 horizontal and vertical neighbors.

□ Their coordinates are given by:

$$(x+1, y) \quad (x-1, y) \quad (x, y+1) \quad \& \quad (x, y-1) \\ f(2,1) \quad f(0,1) \quad f(1,2) \quad f(1,0)$$

□ This set of pixels is called the **4-neighbors** of p denoted by $N_4(p)$.

□ Each pixel is unit distance from (x, y) .

Neighbors of a Pixel (Contd..)

- The four diagonal neighbors of $p(x,y)$ are given by,
 $(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$
- This set is denoted by $N_D(P)$.
- Each of them are at Euclidean distance of 1.414 from P.

Neighbors of a Pixel

$$f_{\text{xx}} = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & f(0,3) & f(0,4) & \dots \\ f(1,0) & f(1,1) & f(1,2) & f(1,3) & f(1,4) & \dots \\ f(2,0) & f(2,1) & f(2,2) & f(2,3) & f(2,4) & \dots \\ f(3,0) & f(3,1) & f(3,2) & f(3,3) & f(3,4) & \dots \\ | & \downarrow & \downarrow & \downarrow & \downarrow & \dots \\ | & \downarrow & \downarrow & \downarrow & \downarrow & \dots \end{bmatrix}$$

- A Pixel p at coordinates (x, y) has 4 diagonal neighbors.
- Their coordinates are given by:
$$(x+1, y+1) \quad (x+1, y-1) \quad (x-1, y+1) \quad \& \quad (x-1, y-1)$$
$$f(2,2) \quad f(2,0) \quad f(0,2) \quad f(0,0)$$
- This set of pixels is called the diagonal-neighbors of p denoted by $N_D(p)$.
- diagonal neighbors + 4-neighbors = 8-neighbors of p .
- They are denoted by $N_8(p)$.
So, $N_8(p) = N_4(p) + N_D(p)$

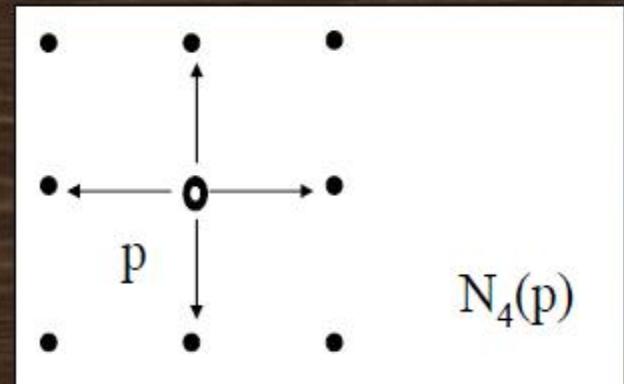
Neighbors of a Pixel (Contd..)

- The points $N_D(P)$ and $N_4(P)$ are together known as 8-neighbors of the point P, denoted by $N_8(P)$.
- Some of the points in the N_4 , N_D and N_8 may fall outside image when P lies on the border of image.

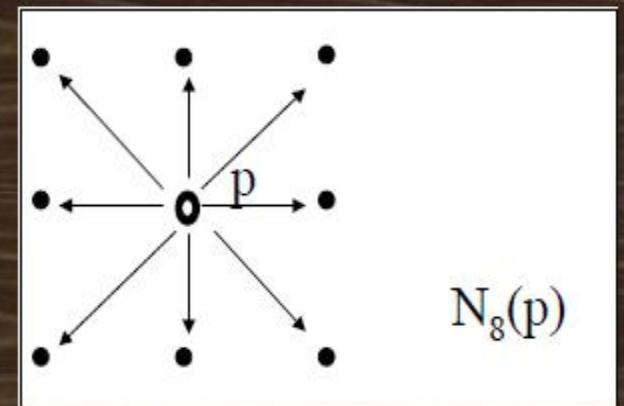
Neighbors of a Pixel (Contd..)

Neighbors of a pixel

- a. 4-neighbors of a pixel p
are its vertical and horizontal neighbors denoted by $N_4(p)$



- b. 8-neighbors of a pixel p
are its vertical horizontal and 4 diagonal neighbors denoted by $N_8(p)$



Neighbors of a Pixel (Contd..)

N_D	N_4	N_D
N_4	P	N_4
N_D	N_4	N_D

- N_4 - 4-neighbors
- N_D - diagonal neighbors
- N_8 - 8-neighbors ($N_4 \cup N_D$)

Adjacency

- Two pixels are connected if they are neighbors and their gray levels satisfy some specified criterion of similarity.
- For example, in a binary image two pixels are connected if they are 4-neighbors and have same value (0/1).

Adjacency, Connectivity

Adjacency: Two pixels are adjacent if they are neighbors and their intensity level 'V' satisfy some specific criteria of similarity.

e.g. $V = \{1\}$

$V = \{0, 2\}$

Binary image = {0, 1}

Gray scale image = {0, 1, 2, -----, 255}

In binary images, 2 pixels are adjacent if they are neighbors & have some intensity values either 0 or 1.

In gray scale, image contains more gray level values in range 0 to 255.

Adjacency (contd.)

- Let V be set of gray levels values used to define adjacency.
- **4-adjacency**: Two pixels p and q with values from V 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**: Two pixels p and q with values from V are m-adjacent if,
 - q is in $N_4(P)$.
 - q is in $N_D(p)$ and the set $[N_4(p) \cap N_4(q)]$ is empty (has no pixels whose values are from V).

Adjacency, Connectivity

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

e.g. $V = \{0, 1\}$

1	1	0
1	1	0
1	0	1

p in **RED** color

q can be any value in **GREEN** color.

Adjacency, Connectivity

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

e.g. $V = \{1, 2\}$

○	1	1
○	2	○
○	○	1

p in **RED** color

q can be any value in **GREEN** color

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

(i) q is in $N_4(p)$ OR

(ii) q is in $N_D(p)$ & the set $N_4(p) \cap N_4(q)$ have no pixels whose values are from 'V'.

e.g. $V = \{1\}$

O a	1 b	1 c
O d	1 e	O f
O g	O h	1 j

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

- (i) q is in $N_4(p)$

e.g. $V = \{1\}$

- (ii) b & c

O a	1 b	1 c
O d	1 e	O f
O g	O h	1 i

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

- (i) q is in $N_4(p)$

e.g. $V = \{1\}$

- (i) b & c

O	a	1	b	1	c
O	d	1	e	O	f
O	g	O	h	1	i

Soln: b & c are m-adjacent.

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

- (i) q is in $N_4(p)$

e.g. $V = \{1\}$

- (ii) b & e

O a	1 b	1 c
O d	1 e	O f
O g	O h	1 i

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

- (i) q is in $N_4(p)$

e.g. $V = \{1\}$

- (ii) b & e

O a	1 b	1 c
O d	1 e	O f
O g	O h	1 i

Soln: b & e are m-adjacent.

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

(i) q is in $N_4(p)$ OR

e.g. $V = \{1\}$

(iii) e & j

O a 1 b 1 c

O d 1 e O f

O g O h 1 j

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

- (i) q is in $N_D(p)$ & the set $N_4(p) \cap N_4(q)$ have no pixels whose values are from 'V'.

e.g. $V = \{1\}$

(iii) e & j

O	a	1	b	1	c
O	d	1	e	O	f
O	g	O	h	1	i

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

- (i) q is in $N_D(p)$ & the set $N_4(p) \cap N_4(q)$ have no pixels whose values are from 'V'.

e.g. $V = \{1\}$

(iii) e & i

O a	1 b	1 c
O d	1 e	O f
O g	O h	1 i

Soln: e & i are m-adjacent.

Adjacency, Connectivity

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

(i) q is in $N_4(p)$ OR

(ii) q is in $N_D(p)$ & the set $N_4(p) \cap N_4(q)$ have no pixels whose values are from 'V'.

e.g. $V = \{1\}$

(iv) e & c

O a 1 b 1 c

O d 1 e O f

O g O h 1 i

m-adjacency: Two pixels p and q with the values from set 'V' are m-adjacent if

(i) q is in $N_4(p)$ OR

(ii) q is in $N_D(p)$ & the set $N_4(p) \cap N_4(q)$ have no pixels whose values are from 'V'.

e.g. $V = \{1\}$

(iv) e & c

O a 1 b 1 c

O d 1 e O f

O g O h 1 i

Soln: e & c are NOT m-adjacent.

Adjacency, Connectivity

Connectivity: 2 pixels are said to be connected if there exists a path between them.

Let 'S' represent subset of pixels in an image.

Two pixels p & q are said to be connected in 'S' if there exists a path between them consisting entirely of pixels in 'S'.

For any pixel p in S, the set of pixels that are connected to it in S is called a **connected component of S.**

Paths & Path lengths

- A *path* from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates:
 $(x_0, y_0), (x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$,
where $(x_0, y_0)=(x, y)$ and $(x_n, y_n)=(s, t)$;
 (x_i, y_i) is adjacent to (x_{i-1}, y_{i-1}) $1 \leq i \leq n$
- Here n is the *length* of the path.
- We can define 4-, 8-, and m-paths based on type of adjacency used.

Paths

Example # 1: Consider the image segment shown in figure. Compute length of the **shortest-4**, **shortest-8 & shortest-m paths** between pixels p & q where,
 $V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	1	2	3

Paths

Example # 1:

Shortest-4 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1	2	3	

Paths

Example # 1:

Shortest-4 path:

$$V = \{1, 2\}.$$

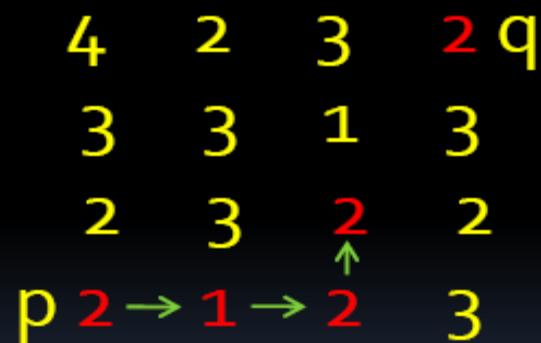
4	2	3	2	q
3	3	1	3	
2	3	2	2	
p 2 → 1 → 2 3				

Paths

Example # 1:

Shortest-4 path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-4 path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-4 path:

$$V = \{1, 2\}.$$



So, Path does not exist.

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	1	2	3

• Rectangular Snip

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1	2	3	

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-8 path:

$$V = \{1, 2\}.$$

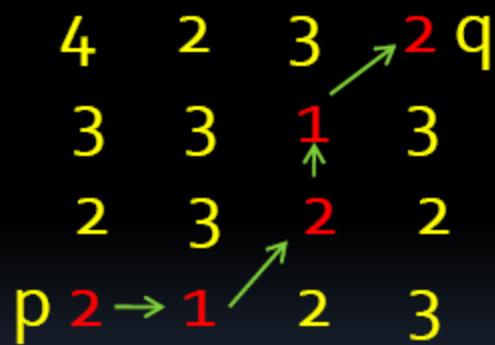


Paths

Example # 1:

Shortest-8 path:

$$V = \{1, 2\}.$$



Paths

Example # 1:

Shortest-8path:

$$V = \{1, 2\}.$$



So, shortest-8path = 4

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	1	2	3

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1	2	3	

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.

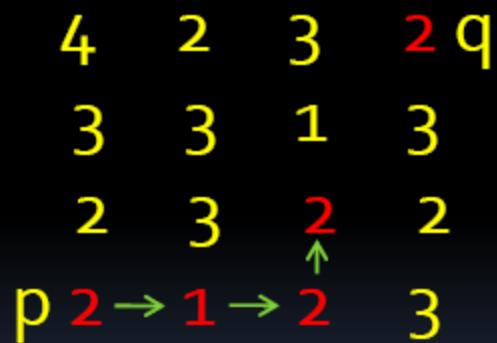
4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1 → 2		3	

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.

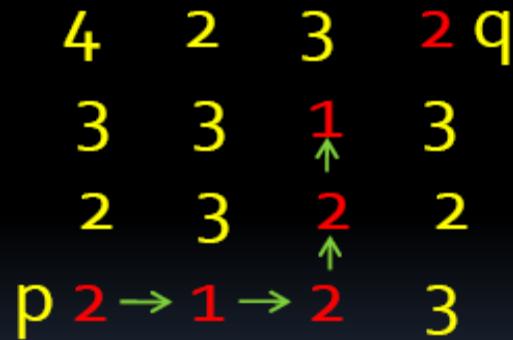


Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



So, shortest-m path = 5

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



So, shortest-m path = 5

Connected Components

- If p and q are pixels of an image subset S then p is *connected* to q in S if there is a path from p to q consisting entirely of pixels in S .
- For every pixel p in S , the set of pixels in S that are connected to p is called a *connected component* of S .
- If S has only one connected component then S is called *Connected Set*.

Regions and Boundaries

- A subset R of pixels in an image is called a *Region* of the image if R is a connected set.
- The *boundary* of the region R is the set of pixels in the region that have one or more neighbors that are not in R .

Distance measures

Given pixels p , q and z with coordinates (x, y) , (s, t) , (u, v) respectively, the distance function D has following properties:

- a. $D(p, q) \geq 0$ [$D(p, q) = 0$, iff $p = q$]
- b. $D(p, q) = D(q, p)$
- c. $D(p, z) \leq D(p, q) + D(q, z)$

The following are the different Distance measures:

- **Euclidean Distance :**

$$D_e(p, q) = \sqrt{(x-s)^2 + (y-t)^2}$$

- b. **City Block Distance:** →

$$D_1(p, q) = |x-s| + |y-t|$$

2	2	2	2	2
2	1	2		
2	1	0	1	2
2	1	2		
2	2			

- c. **Chess Board Distance:** →

$$D_\infty(p, q) = \max(|x-s|, |y-t|)$$

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Path

A path from $p(x,y)$ to $q(s,t)$ is a sequence of distinct pixels.

$$(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$$

Where

$$(x_0, y_0) = (x, y), (x_n, y_n) = (s, t)$$

(x_i, y_i) is adjacent to (x_{i-1}, y_{i-1})

for $1 \leq i \leq n$

$n \Rightarrow$ length of the path.

Connected component

Let

$$S \subseteq I \text{ and } p, q \in S$$

Then p is connected to q in S if there is a path
From p to q consisting entirely of pixels in S

For any $p \in S$, the set of pixels in S that are
Connected to p is call a connected component
of S .

=> Any two pixels of a connected component
are connected to each other
Distinct connected components are disjoint

Connected component labelling

Ability to assign different labels to various disjoint connected components of an Image.



Connected component labeling is a fundamental step in automated image analysis

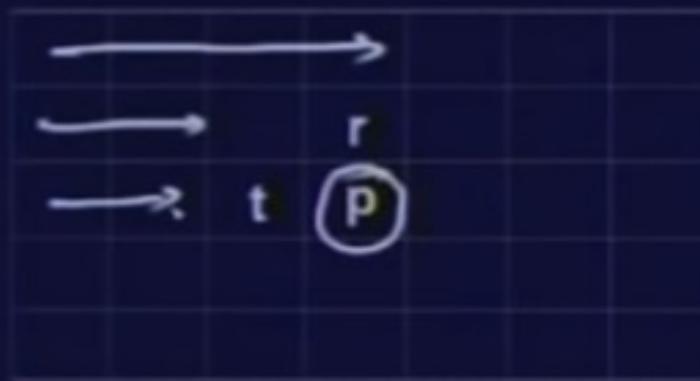
- Shape
- Area
- Boundary
- Shape/Area/Boundary based features**

Algorithm

Scan an image from left to right and from top to bottom.

Assume 4 - connectivity

P be a pixel at any step in the scanning process.



Before p, points r and t are scanned

STEPS

$I(p)$ => Pixel value at position p.

$L(p)$ => Label assigned to pixel location p.

If $I(p) = 0$, move to next scanning position.

If $I(p) = 1$ and $I(r) = I(t) = 0$

Then assign a new label to position p

If $I(p) = 1$ and only one of the two neighbor
is 1

Then assign its label to p.

If $I(p) = 1$ and both r and t are 1's, then

If $L(r) = L(t)$ than $L(p)=L(r)$

If $L(r) \neq L(t)$ then assign one of the
labels to p and make a note that the
two labels are equivalent

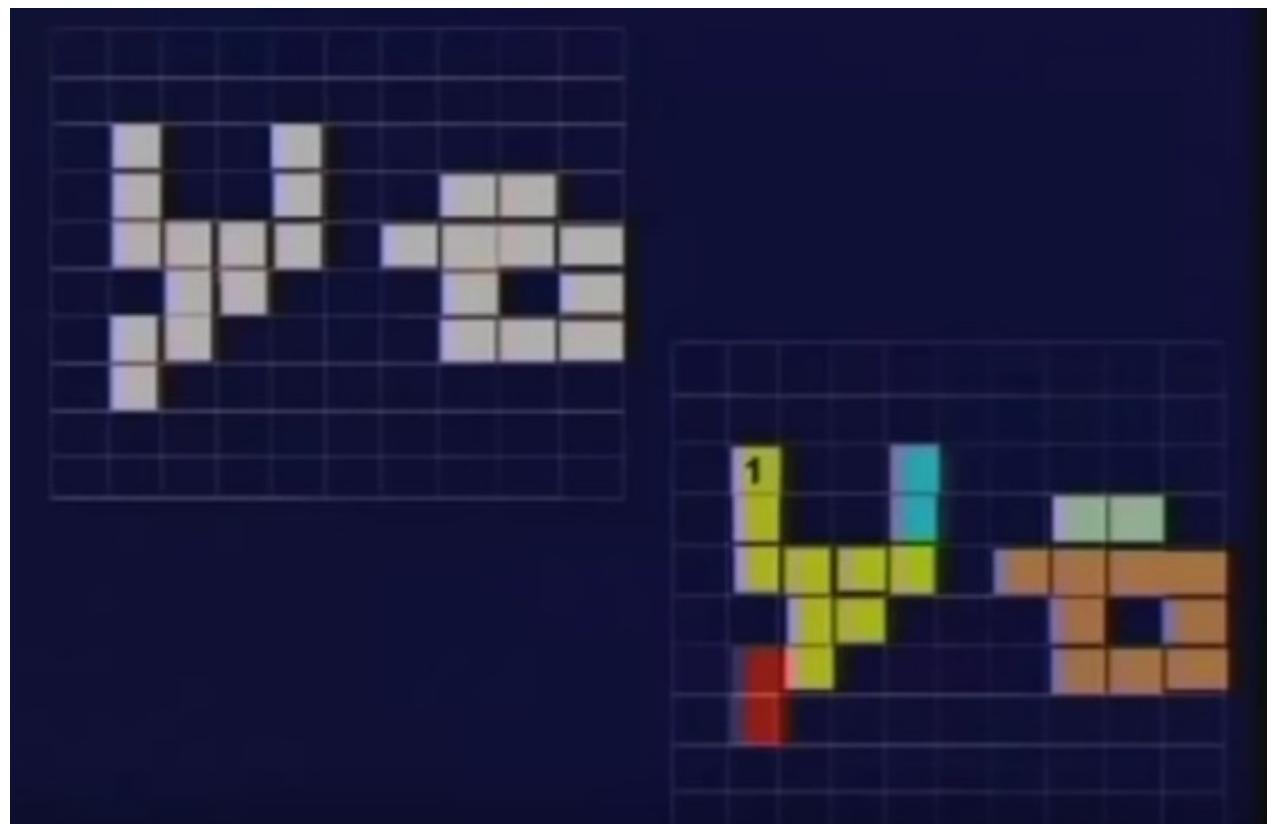
At end of the scan all pixels with value 1 are labeled.

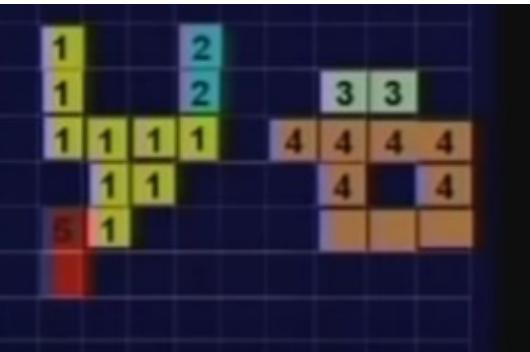
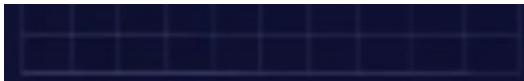
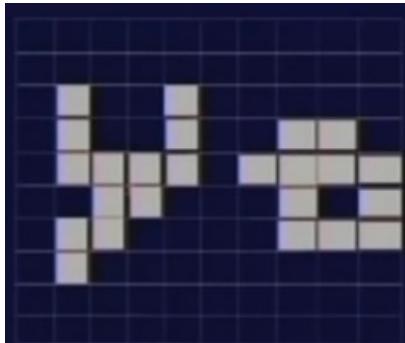
Some labels are equivalent.

During second pass process equivalent pairs to from equivalence classes.

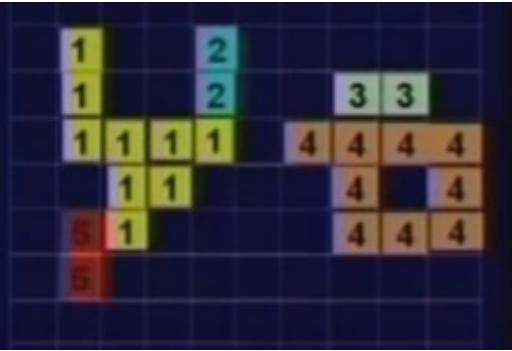
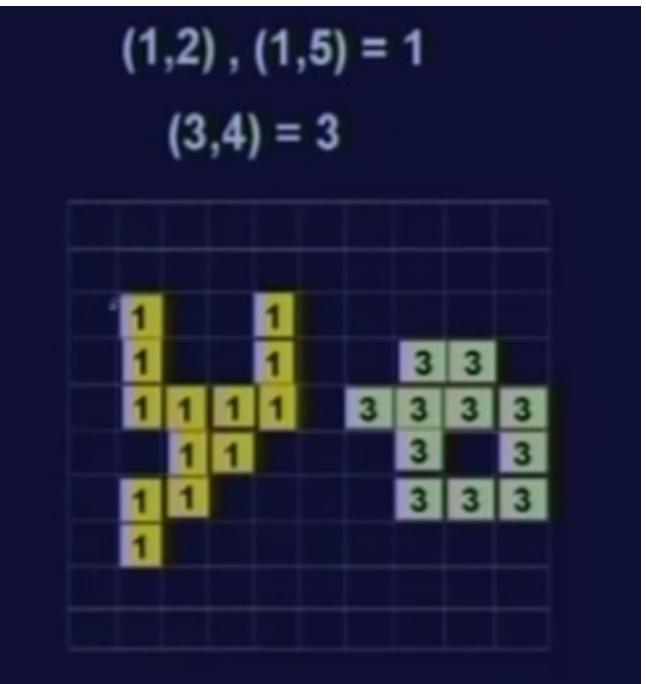
**Assign a different label to each class.
In the second pass through the image
replace each label by the label assigned
to its equivalence class.**

Algorithm demonstration





(1,2) , (3,4) , (1,5)



Equivalent pairs:

(1,2) , (3,4) , (1,5)

Paths & Path lengths

- A *path* from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates:
 $(x_0, y_0), (x_1, y_1), (x_2, y_2) \dots (x_n, y_n)$,
where $(x_0, y_0)=(x, y)$ and $(x_n, y_n)=(s, t)$;
 (x_i, y_i) is adjacent to (x_{i-1}, y_{i-1}) $1 \leq i \leq n$
- Here n is the *length* of the path.
- We can define 4-, 8-, and m-paths based on type of adjacency used.

Paths

Example # 1: Consider the image segment shown in figure. Compute length of the **shortest-4**, **shortest-8** & **shortest-m paths** between pixels p & q where,
 $V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	1	2	3

Paths

Example # 1:

Shortest-4 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1	2	3	

Paths

Example # 1:

Shortest-4 path:

$V = \{1, 2\}$.

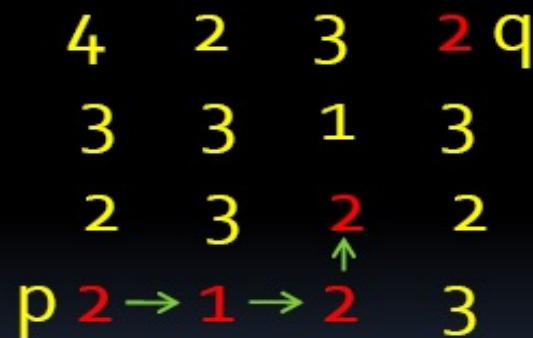
4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	→ 1 → 2	3	

Paths

Example # 1:

Shortest-4 path:

$$V = \{1, 2\}.$$

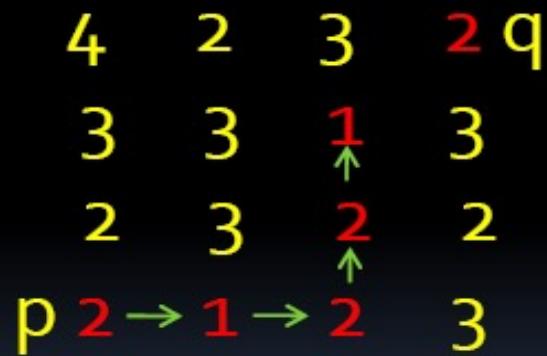


Paths

Example # 1:

Shortest-4 path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-4 path:

$$V = \{1, 2\}.$$



So, Path does not exist.

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	1	2	3

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1	2	3	

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.



• Rectangular Snip

Paths

Example # 1:

Shortest-8 path:

$$V = \{1, 2\}.$$

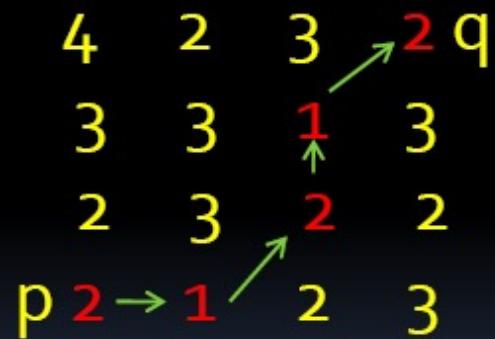


Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.

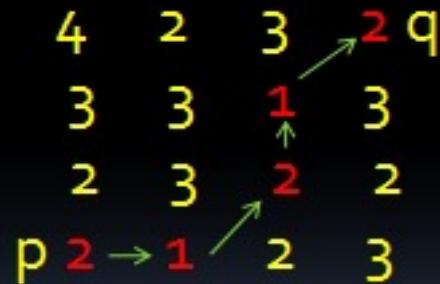


Paths

Example # 1:

Shortest-8path:

$$V = \{1, 2\}.$$



So, shortest-8path = 4

Paths

Example # 1:

Shortest-m path:

$$V = \{1, 2\}.$$

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2	1	2	3

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1	2	3	

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p	2 → 1 → 2		3	

Paths

Example # 1:

Shortest-m path:

$$V = \{1, 2\}.$$



Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



Paths

Example # 1:

Shortest-m path:

$$V = \{1, 2\}.$$



Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



So, shortest-m path = 5

Paths

Example # 1:

Shortest-m path:

$V = \{1, 2\}$.



So, shortest-m path = 5

Relationships between Pixels

- On completion the students will be able to
 - 1. Learn different distance measures
 - 2. Application of Distance measure
 - 3. Arithmetic/ Logical operations on images
 - 4. Neighborhood operations on images

Take three pixels

$$P \approx (x,y) \quad q \approx (s,t) \quad z \approx (u,v)$$

D is a **distance function or metric** if

$$D(p,q) \geq 0 ; \quad D(p,q) = 0 \quad \text{iff} \quad p = q$$

$$D(p,q) = D(q,p)$$

$$D(p,z) \leq D(p,q) + D(q,z)$$

Euclidian Distance

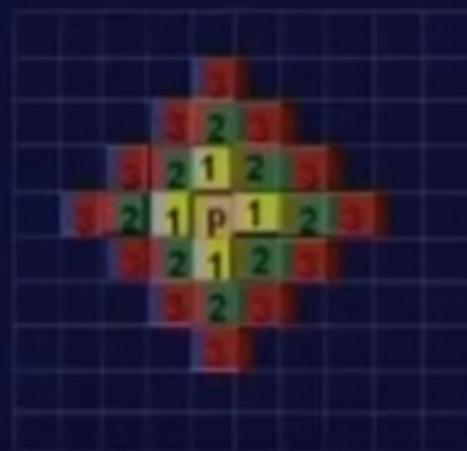
$$D_e(p,q) = [(x-s)^2 + (y-t)^2]^{\frac{1}{2}}$$

Set of points $S = \{ q \mid D(p,q) \leq r \}$ are the
points contained in a disk of radius r
centered at p .

D_4 distance or City-Block (Manhattan) Distance.

$$D_4(p,q) = |x-s| + |y-t|$$

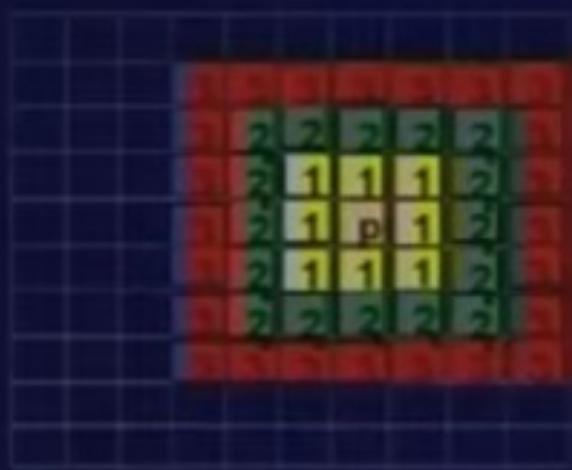
Points having city block distance from p less than or equal to r from diamond centered at p.



D_8 distance or chess board distance is defined as

$$D_8(p,q) = \max(|x-s|, |y-t|)$$

$S = \{ q \mid D_8(p,q) \leq r \}$ forms a square centered at p.



Points with $D_8 = 1$ are 8 neighbors of p

Arithmetic / Logical Operation

Following Arithmetic/Logical operations between two pixels p and q are used extensively

Arithmetic

$$p+q$$

$$p-q$$

$$p^*q$$

$$p\%q$$

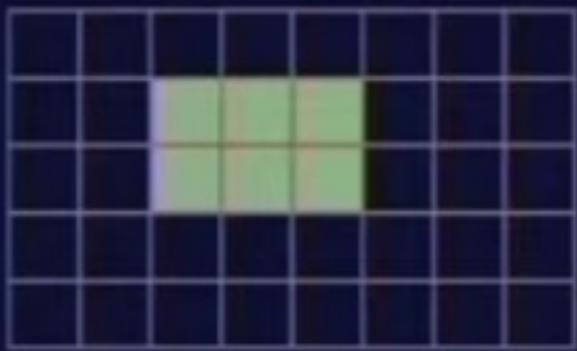
Logical

$$p.q$$

$$p+q$$

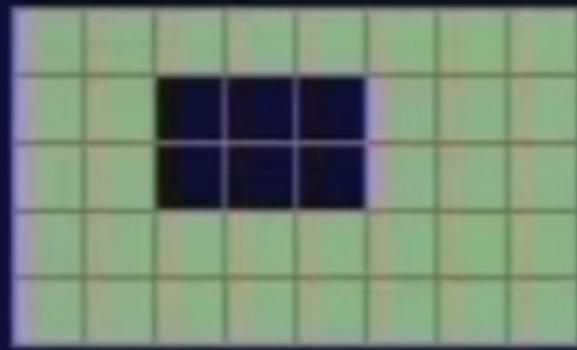
$$p'$$

Logical operations apply to binary images
Only => Usually pixel by



A

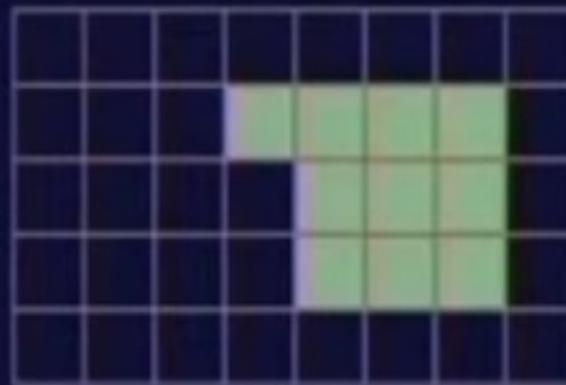
NOT (A)



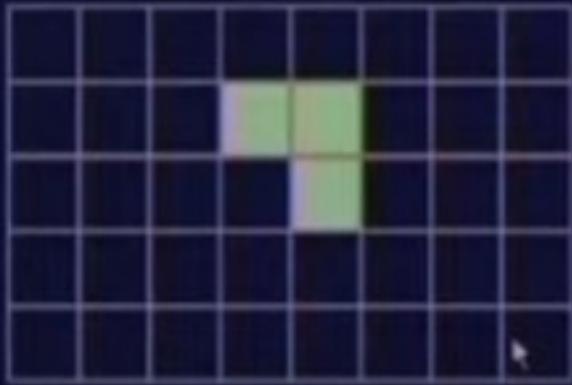
A



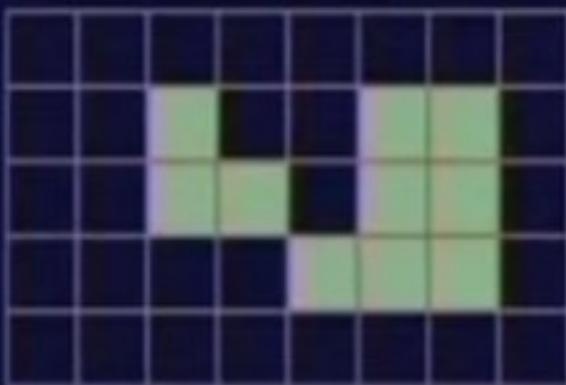
B



(A) AND (B)



(A) XOR (B)



Neighborhood Operations

The value assigned to a pixel is a function of its gray label and the gray labels of its neighbors.

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

$$Z = \frac{1}{9} (Z_1 + Z_2 + Z_3 + \dots + Z_9) = \text{Average}$$

Template

More general form

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

$$Z = w_1 z_1 + w_2 z_2 + \dots + w_9 z_9$$

$$= \sum_{i=1}^9 w_i z_i$$

Same as averaging if $w_i = 1/9$

Neighborhood Operations

Various important operations can be Implemented by proper selection of Coefficients W_i

- Noise filtering
- Thinning
- Edge detection
- etc...

Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

August 25, 2021

Concept of Zooming.

Aspect ratio

Another important concept with the pixel resolution is aspect ratio.

Aspect ratio is the ratio between width of an image and the height of an image. It is commonly explained as two numbers separated by a colon (8:9). This ratio differs in different images, and in different screens. The common aspect ratios are:

1.33:1, 1.37:1, 1.43:1, 1.50:1, 1.56:1, 1.66:1, 1.75:1, 1.78:1, 1.85:1, 2.00:1, e.t.c

Advantage

Aspect ratio maintains a balance between the appearance of an image on the screen, means it maintains a ratio between horizontal and vertical pixels. It does not let the image to get distorted when aspect ratio is increased.

For example

If you are given an image with aspect ratio of 6:2 of an image of pixel resolution of 480000 pixels given the image is an gray scale image.

And you are asked to calculate two things.

- Resolve pixel resolution to calculate the dimensions of image
- Calculate the size of the image

Solution:

Given:

Aspect ratio: $c:r = 6:2$

Pixel resolution: $c * r = 480000$

Bits per pixel: grayscale image = 8bpp

Find:

Number of rows = ?

Number of cols = ?

Solving first part:

$$\text{Equation 1.} \quad \underline{c:r = 6:2} \rightarrow c = 6 r / 2$$

$$\text{Equation 2.} \quad c = 480000/r$$

$$\text{Comparing both equations} \rightarrow \frac{6r}{2} = \frac{480000}{r}$$

$$r^2 = \sqrt{\frac{480000 * 2}{6}}$$

$$\text{That gives} \quad r = 400.$$

That gives $r = 400$.

Put r in equation 1, we get $\rightarrow c = 1200$.

So rows = 400 cols = 1200.

Solving 2nd part:

$$\text{Size} = \text{rows} * \text{cols} * \text{bpp}$$

$$\text{Size of image in bits} = 400 * 1200 * 8 = 3840000 \text{ bits}$$

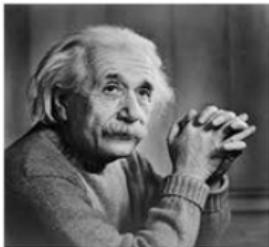
$$\text{Size of image in bytes} = 480000 \text{ bytes}$$

$$\text{Size of image in kilo bytes} = 48 \text{ kb (approx)}.$$

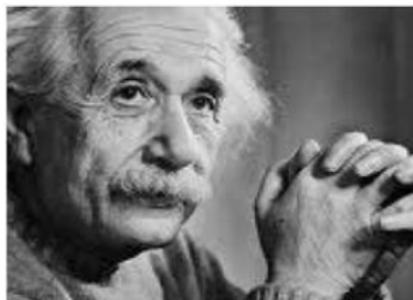
Zooming

Zooming simply means enlarging a picture in a sense that the details in the image became more visible and clear. Zooming an image has many wide applications ranging from zooming through a camera lens, to zoom an image on internet e.t.c.

For example



is zoomed into



You can zoom something at two different steps.

The first step includes zooming before taking an particular image. This is known as pre processing zoom. This zoom involves hardware and mechanical movement.

The second step is to zoom once an image has been captured. It is done through many different algorithms in which we manipulate pixels to zoom in the required portion.

We will discuss them in detail in the next tutorial.

Optical Zoom vs digital Zoom

These two types of zoom are supported by the cameras.

Optical Zoom:

The optical zoom is achieved using the movement of the lens of your camera. An optical zoom is actually a true zoom. The result of the optical zoom is far better than that of digital zoom. In optical zoom, an image is magnified by the lens in such a way that the objects in the image appear to be closer to the camera. In optical zoom the lens physically extends to zoom or magnify an object.

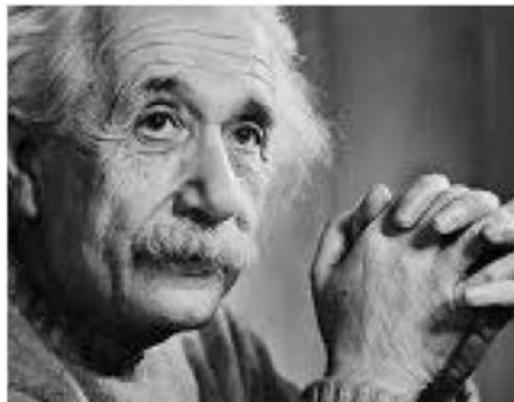
Digital Zoom:

Digital zoom is basically image processing within a camera. During a digital zoom, the center of the image is magnified and the edges of the picture get cropped out. Due to magnified center, it looks like that the object is closer to you.

During a digital zoom, the pixels got expand , due to which the quality of the image is compromised.

The same effect of digital zoom can be seen after the image is taken through your computer by using an image processing toolbox / software, such as Photoshop.

The following picture is the result of digital zoom done through one of the following methods given below in the zooming methods.



Now since we are leaning digital image processing, we will not focus, on how an image can be zoomed optically using lens or other stuff. Rather we will focus on the methods, that enable to zoom a digital image.

Zooming methods:

Although there are many methods that does this job, but we are going to discuss the most common of them here.

They are listed below.

- Pixel replication or (Nearest neighbor interpolation)
- Zero order hold method
- Zooming K times

All these three methods are formally introduced in the next tutorial.

Method 1: Pixel replication:

Introduction:

It is also known as Nearest neighbor interpolation. As its name suggest, in this method, we just replicate the neighboring pixels. As we have already discussed in the tutorial of Sampling, that zooming is nothing but increase amount of sample or pixels. This algorithm works on the same principle.

Working:

In this method we create new pixels from the already given pixels. Each pixel is replicated in this method n times row wise and column wise and you got a zoomed image. Its as simple as that.

For example:

If you have an image of 2 rows and 2 columns and you want to zoom it twice or 2 times using pixel replication, here how it can be done.

For a better understanding, the image has been taken in the form of matrix with the pixel values of the image.

1	2
3	4

The above image has two rows and two columns, we will first zoom it row wise.

Row wise zooming:

When we zoom it row wise, we will just simple copy the rows pixels to its adjacent new cell.

Here how it would be done.

1	1	2	2
3	3	4	4

As you can see in the above matrix, each pixel is replicated twice in the rows.

Column size zooming:

The next step is to replicate each of the pixel column wise, that we will simply copy the column pixel to its adjacent new column or simply below it.

Here how it would be done.

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

New image size:

As it can be seen from the above example, that an original image of 2 rows and 2 columns has been converted into 4 rows and 4 columns after zooming. That means the new image has a dimensions of
(Original image rows * zooming factor, Original Image cols * zooming factor)

Advantage and disadvantage:

One of the advantage of this zooming technique is, it is very simple. You just have to copy the pixels and nothing else.

The disadvantage of this technique is that image got zoomed but the output is very blurry. And as the zooming factor increased, the image got more and more blurred. That would eventually result in fully blurred image.

Method 2: Zero order hold

Introduction

Zero order hold method is another method of zooming. It is also known as zoom twice. Because it can only zoom twice. We will see in the below example that why it does that.

Working

In zero order hold method, we pick two adjacent elements from the rows respectively and then we add them and divide the result by two, and place their result in between those two elements. We first do this row wise and then we do this column wise.

For example

Lets take an image of the dimensions of 2 rows and 2 columns and zoom it twice using zero order hold.

1	2
3	4

First we will zoom it row wise and then column wise.

Row wise zooming

1	1	2
3	3	4

As we take the first two numbers : $(2 + 1) = 3$ and then we divide it by 2, we get 1.5 which is approximated to 1. The same method is applied in the row 2.

Column wise zooming

1	1	2
2	2	3
3	3	4

We take two adjacent column pixel values which are 1 and 3. We add them and got 4. 4 is then divided by 2 and we get 2 which is placed in between them. The same method is applied in all the columns.

New image size

As you can see that the dimensions of the new image are 3 x 3 where the original image dimensions are 2 x 2. So it means that the dimensions of the new image are based on the following formula

$$(2(\text{number of rows}) \text{ minus } 1) \times (2(\text{number of columns}) \text{ minus } 1)$$

Advantages and disadvantage.

One of the advantage of this zooming technique , that it does not create as blurry picture as compare to the nearest neighbor interpolation method. But it also has a disadvantage that it can only run on the power of 2. It can be demonstrated here.

Reason behind twice zooming:

Consider the above image of 2 rows and 2 columns. If we have to zoom it 6 times, using zero order hold method , we can not do it. As the formula shows us this.

It could only zoom in the power of 2 2,4,8,16,32 and so on.

Even if you try to zoom it, you can not. Because at first when you will zoom it two times, and the result would be same as shown in the column wise zooming with dimensions equal to 3x3. Then you will zoom it again and you will get dimensions equal to 5 x 5. Now if you will do it again, you will get dimensions equal to 9 x 9.

Whereas according to the formula of yours the answer should be 11x11. As $(6(2) \text{ minus } 1) \times (6(2) \text{ minus } 1)$ gives 11 x 11.

Method 3: K-Times zooming

Introduction:

K times is the third zooming method we are going to discuss. It is one of the most perfect zooming algorithm discussed so far. It caters the challenges of both twice zooming and pixel replication. K in this zooming algorithm stands for zooming factor.

Working:

It works like this way.

First of all, you have to take two adjacent pixels as you did in the zooming twice. Then you have to subtract the smaller from the greater one. We call this output (OP).

Divide the output(OP) with the zooming factor(K). Now you have to add the result to the smaller value and put the result in between those two values.

Add the value OP again to the value you just put and place it again next to the previous putted value. You have to do it till you place k-1 values in it.

Repeat the same step for all the rows and the columns , and you get a zoomed images.

For example:

Suppose you have an image of 2 rows and 3 columns, which is given below. And you have to zoom it thrice or three times.

15	30	15
30	15	30

K in this case is 3. $K = 3$.

The number of values that should be inserted is $k-1 = 3-1 = 2$.

Row wise zooming

Take the first two adjacent pixels. Which are 15 and 30.

Subtract 15 from 30. $30 - 15 = 15$.

Divide 15 by k. $15/k = 15/3 = 5$. We call it OP.(where op is just a name)

Add OP to lower number. $15 + OP = 15 + 5 = 20$.

Add OP to 20 again. $20 + OP = 20 + 5 = 25$.

We do that 2 times because we have to insert k-1 values.

Now repeat this step for the next two adjacent pixels. It is shown in the first table.

After inserting the values, you have to sort the inserted values in ascending order, so there remains a symmetry between them.

It is shown in the second table

Table 1.

15	20	25	30	20	25	15
30	20	25	15	20	25	30

Column wise zooming

The same procedure has to be performed column wise. The procedure include taking the two adjacent pixel values, and then subtracting the smaller from the bigger one. Then after that, you have to divide it by k. Store the result as OP. Add OP to smaller one, and then again add OP to the value that comes in first addition of OP. Insert the new values.

Here what you got after all that.

15	20	25	30	25	20	15
20	21	21	25	21	21	20
25	22	22	20	22	22	25
30	25	20	15	20	25	30

New image size

The best way to calculate the formula for the dimensions of a new image is to compare the dimensions of the original image and the final image. The dimensions of the original image were 2 X 3. And the dimensions of the new image are 4 x 7.

The formula thus is:

$$(K \text{ (number of rows minus 1)} + 1) \times (K \text{ (number of cols minus 1)} + 1)$$

Advantages and disadvantages

The one of the clear advantage that k time zooming algorithm has that it is able to compute zoom of any factor which was the power of pixel replication algorithm , also it gives improved result (less blurry) which was the power of zero order hold method. So hence It comprises the power of the two algorithms.

The only difficulty this algorithm has that it has to be sort in the end, which is an additional step, and thus increases the cost of computation.

Spatial resolution

Spatial resolution states that the clarity of an image cannot be determined by the pixel resolution. The number of pixels in an image does not matter.

Spatial resolution can be defined as the

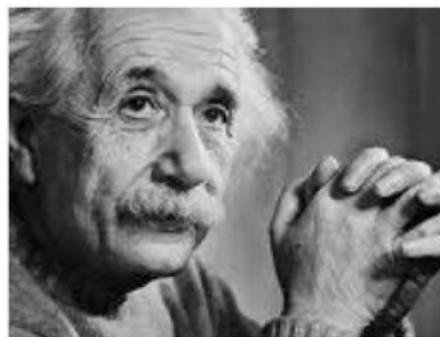
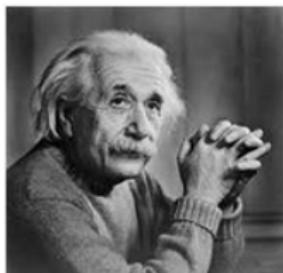
smallest discernible detail in an image. (Digital Image Processing - Gonzalez, Woods - 2nd Edition)

Or in other way we can define spatial resolution as the number of independent pixels values per inch.

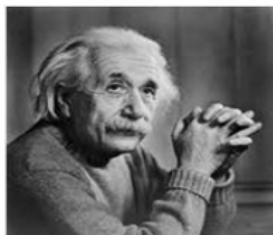
In short what spatial resolution refers to is that we cannot compare two different types of images to see that which one is clear or which one is not. If we have to compare the two images, to see which one is more clear or which has more spatial resolution, we have to compare two images of the same size.

For example:

You cannot compare these two images to see the clarity of the image.



So in order to measure spatial resolution , the pictures below would server the purpose.



Gray level resolution

Gray level resolution refers to the predictable or deterministic change in the shades or levels of gray in an image.

In short gray level resolution is equal to the number of bits per pixel.

We have already discussed bits per pixel in our tutorial of bits per pixel and image storage requirements. We will define bpp here briefly.

BPP

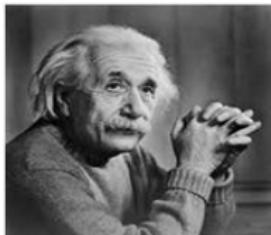
The number of different colors in an image is depends on the depth of color or bits per pixel.

Mathematically

The mathematical relation that can be established between gray level resolution and bits per pixel can be given as.

$$L = 2^k$$

For example:



The above image of Einstein is a gray scale image. Means it is an image with 8 bits per pixel or 8bpp.

Now if we were to calculate the gray level resolution, here's how we gonna do it.

$$L = 2^k$$

Where $k = 8$

$$L = 2^8$$

$$L = 256.$$

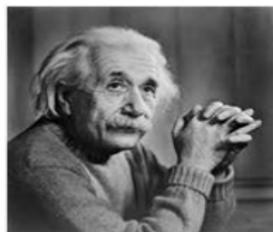
Reducing the gray level

Now we will reduce the gray levels of the image to see the effect on the image.

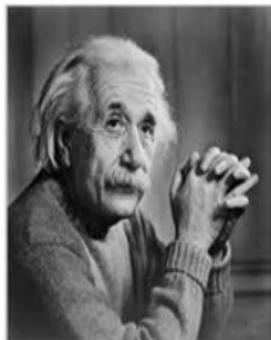
For example

Lets say you have an image of 8bpp, that has 256 different levels. It is a grayscale image and the image looks something like this.

256 Gray Levels

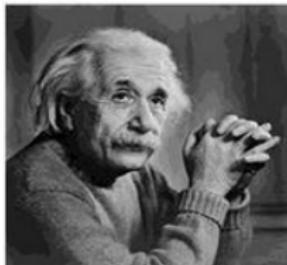


128 Gray Levels



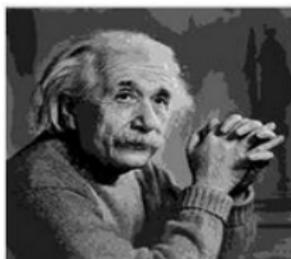
There is not much effect on an image after decrease the gray levels to its half. Lets decrease some more.

16 Gray Levels



Boom here, we go, the image finally reveals, that it is effected by the levels.

8 Gray Levels



Contouring

There is an interesting observation here, that as we reduce the number of gray levels, there is a special type of effect start appearing in the image, which can be seen clear in 16 gray level picture. This effect is known as Contouring.

Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

September 7, 2021

Low Level Image feature Discussion.

Different color codes

All the colors here are of the 24 bit format, that means each color has 8 bits of red, 8 bits of green, 8 bits of blue, in it. Or we can say each color has three different portions. You just have to change the quantity of these three portions to make any color.

Binary color format

Color: Black

Image:



Decimal Code:

(0,0,0)

Color:White

Image:



Decimal Code:

(255,255,255)

Explanation:

Since each portion of R, G, B is an 8 bit portion. So in 8-bit, the white color is formed by 255. It is explained in the tutorial of pixel. So in order to make a white color we set each portion to 255 and that's how we got a white color. By setting each of the value to 255, we get overall value of 255, that's make the color white.

RGB color model:

Color:Red

Image:



Decimal Code:

(255,0,0)

Explanation:

Since we need only red color, so we zero out the rest of the two portions which are green and blue, and we set the red portion to its maximum which is 255.

Color:Green

Image:



Decimal Code:

(0,255,0)

Explanation:

Since we need only green color, so we zero out the rest of the two portions which are red and blue, and we set the green portion to its maximum which is 255.

Color: Blue

Image:



Decimal Code:

(0,0,255)

Explanation:

Since we need only blue color, so we zero out the rest of the two portions which are red and green, and we set the blue portion to its maximum which is 255

Gray color:

Color: Gray

Image:



Decimal Code:

(128,128,128)

Explanation

As we have already defined in our tutorial of pixel, that gray color is actually the mid point. In an 8-bit format, the mid point is 128 or 127. In this case we choose 128. So we set each of the portion to its mid point which is 128, and that results in overall mid value and we got gray color.

CMYK color model:

CMYK is another color model where c stands for cyan, m stands for magenta, y stands for yellow, and k for black. CMYK model is commonly used in color printers in which there are two carters of color is used. One consist of CMY and other consist of black color.

The colors of CMY can also made from changing the quantity or portion of red, green and blue.

Color: Cyan

Image:



Decimal Code:

(0,255,255)

Explanation:

Cyan color is formed from the combination of two different colors which are Green and blue. So we set those two to maximum and we zero out the portion of red. And we get cyan color.

Color: Magenta

Image:



Decimal Code:

(255,0,255)

Explanation:

Magenta color is formed from the combination of two different colors which are Red and Blue. So we set those two to maximum and we zero out the portion of green. And we get magenta color.

Color: Yellow

Image:



Decimal Code:

(255,255,0)

Explanation:

Yellow color is formed from the combination of two different colors which are Red and Green. So we set those two to maximum and we zero out the portion of blue. And we get yellow color.

Average method

Average method is the most simple one. You just have to take the average of three colors. Since its an RGB image, so it means that you have add r with g with b and then divide it by 3 to get your desired grayscale image.

Its done in this way.

$$\text{Grayscale} = (R + G + B / 3)$$

For example:



If you have an color image like the image shown above and you want to convert it into grayscale using average method. The following result would appear.



Explanation

There is one thing to be sure, that something happens to the original works. It means that our average method works. But the results were not as expected. We wanted to convert the image into a grayscale, but this turned out to be a rather black image.

Problem

This problem arise due to the fact, that we take average of the three colors. Since the three different colors have three different wavelength and have their own contribution in the formation of image, so we have to take average according to their contribution, not done it averagely using average method. Right now what we are doing is this,

33% of Red, 33% of Green, 33% of Blue

We are taking 33% of each, that means, each of the portion has same contribution in the image. But in reality that's not the case. The solution to this has been given by luminosity method.

Weighted method or luminosity method

You have seen the problem that occur in the average method. Weighted method has a solution to that problem. Since red color has more wavelength of all the three colors, and green is the color that has not only less wavelength than red color but also green is the color that gives more soothing effect to the eyes.

It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two.

So the new equation that form is:

$$\text{New grayscale image} = ((0.3 * R) + (0.59 * G) + (0.11 * B)).$$

According to this equation, Red has contributed 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

Applying this equation to the image, we get this

Original Image:



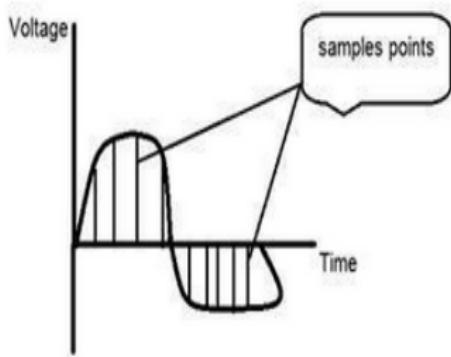
Grayscale Image:



Digitizing a signal

As we have seen in the previous tutorials, that digitizing an analog signal into a digital, requires two basic steps. Sampling and quantization. Sampling is done on x axis. It is the conversion of x axis (infinite values) to digital values.

The below figure shows sampling of a signal.



If you will look at the above figure, you will see that there are some random variations in the signal. These variations are due to noise. In sampling we reduce this noise by taking samples. It is obvious that more samples we take, the quality of the image would be more better, the noise would be more removed and same happens vice versa.

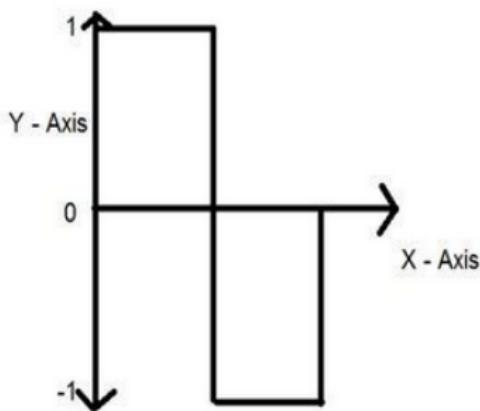
However, if you take sampling on the x axis, the signal is not converted to digital format, unless you take sampling of the y-axis too which is known as quantization. The more samples eventually means you are collecting more data, and in case of image, it means more pixels.

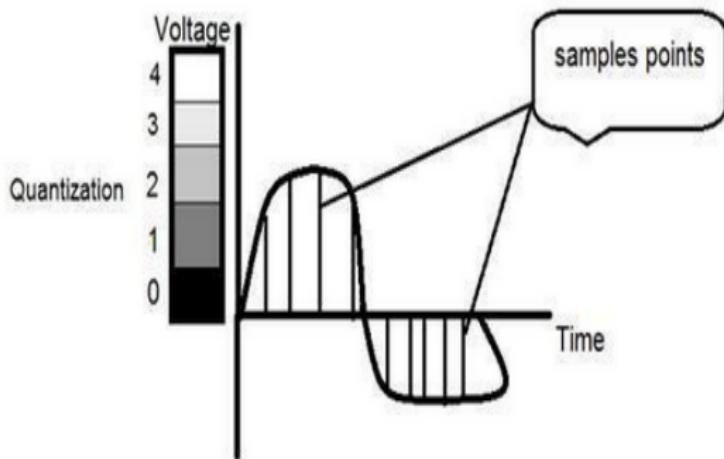
What is quantization

Quantization is opposite to sampling. It is done on y axis. When you are quantizing an image, you are actually dividing a signal into quanta(partitions).

On the x axis of the signal, are the co-ordinate values, and on the y axis, we have amplitudes. So digitizing the amplitudes is known as Quantization.

Here how it is done





In the figure shown in sampling, although the samples have been taken, but they were still spanning vertically to a continuous range of gray level values. In the figure shown above, these vertically ranging values have been quantized into 5 different levels or partitions. Ranging from 0 black to 4 white. This level could vary according to the type of image you want.

Pixel

We have already defined a pixel in our tutorial of concept of pixel, in which we define a pixel as the smallest element of an image. We also defined that a pixel can store a value proportional to the light intensity at that particular location.

Now since we have defined a pixel, we are going to define what is resolution.

Resolution

The resolution can be defined in many ways. Such as pixel resolution, spatial resolution, temporal resolution, spectral resolution. Out of which we are going to discuss pixel resolution.

You have probably seen that in your own computer settings, you have monitor resolution of 800 x 600, 640 x 480 e.t.c

In pixel resolution, the term resolution refers to the total number of count of pixels in an digital image. For example. If an image has M rows and N columns, then its resolution can be defined as $M \times N$.

If we define resolution as the total number of pixels, then pixel resolution can be defined with set of two numbers. The first number the width of the picture, or the pixels across columns, and the second number is height of the picture, or the pixels across its width.

We can say that the higher is the pixel resolution, the higher is the quality of the image.

We can define pixel resolution of an image as 4500 X 5500.

Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

September 7, 2021

Concept of Dithering.

Dithering

Dithering is the process by which we create illusions of the color that are not present actually. It is done by the random arrangement of pixels.

For example. Consider this image.



This is an image with only black and white pixels in it. Its pixels are arranged in an order to form another image that is shown below. Note at the arrangement of pixels has been changed, but not the quantity of pixels.

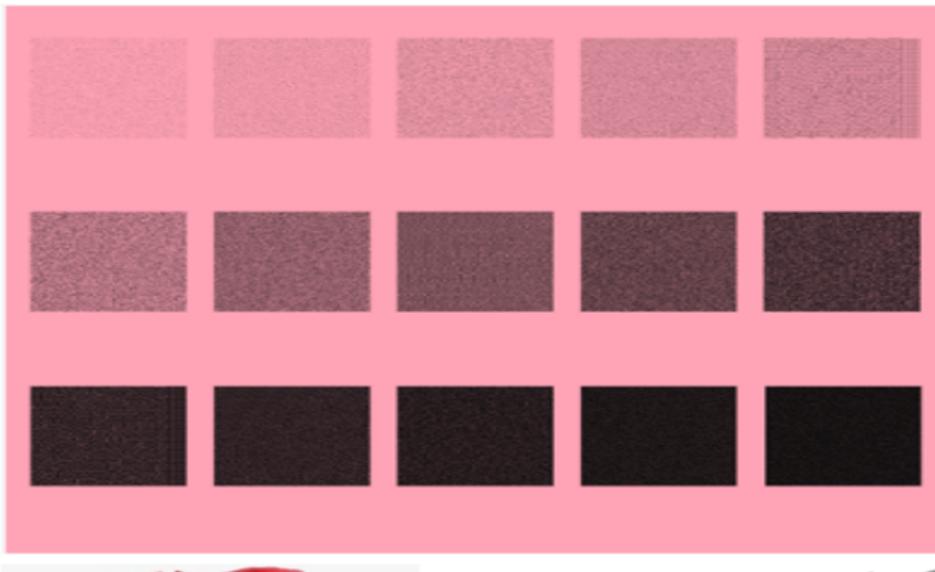


Dithering is a technique to convert a grayscale image to black and white. It is used to create an illusion of the color which is actually not present. Dithering is done by randomly arranging the pixels.

Dither is applied in the form of noise to prevent quantization error.

Dither is commonly used in the processing of both digital audio and video data. It is the last stage of mastering audio to a CD.

Below image have different levels of dithering.





Original image



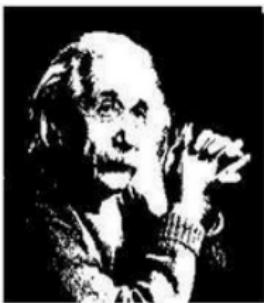
Image having Dithering effect

Why Dithering?

Why do we need dithering, the answer of this lies in its relation with quantization.

Dithering with quantization

When we perform quantization, to the last level, we see that the image that comes in the last level (level 2) looks like this.



Now as we can see from the image here, that the picture is not very clear, especially if you will look at the left arm and back of the image of the Einstein. Also this picture does not have much information or detail of the Einstein.

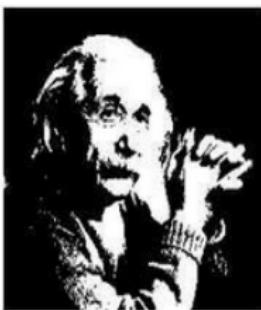
Now if we were to change this image into some image that gives more detail then this, we have to perform dithering.

Performing dithering

First of all, we will work on thresholding. Dithering is usually working to improve thresholding. During thresholding, the sharp edges appear where gradients are smooth in an image.

In thresholding, we simply choose a constant value. All the pixels above that value are considered as 1 and all the value below it are considered as 0.

We got this image after thresholding.



Since there is not much change in the image, as the values are already 0 and 1 or black and white in this image.

Now we perform some random dithering to it. Its some random arrangement of pixels.



We got an image that gives slighter of the more details, but its contrast is very low.

So we do some more dithering that will increase the contrast. The image that we got is this:



Now we mix the concepts of random dithering, along with threshold and we got an image like this.



Now you see, we got all these images by just re-arranging the pixels of an image. This re-arranging could be random or could be according to some measure.

Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

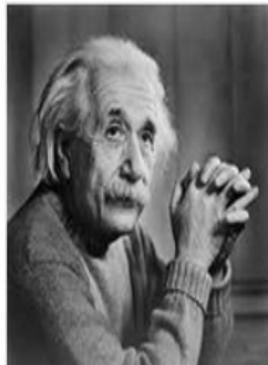
September 7, 2021

Concept of Histogram.

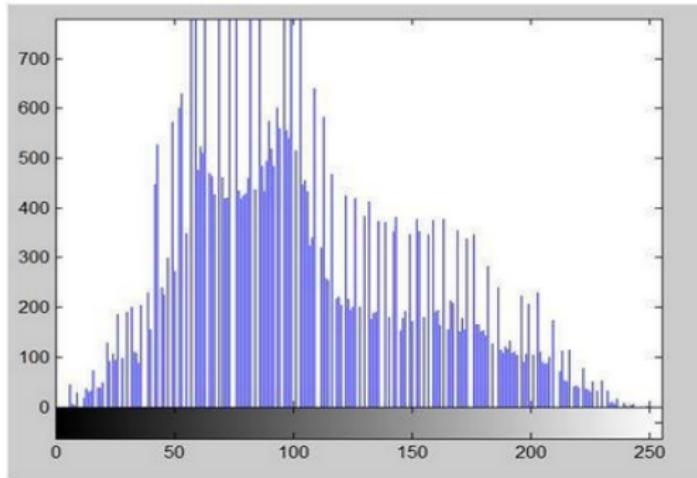
Histogram of an image

Histogram of an image, like other histograms also shows frequency. But an image histogram, shows frequency of pixels intensity values. In an image histogram, the x axis shows the gray level intensities and the y axis shows the frequency of these intensities.

For example



The histogram of the above picture of the Einstein would be something like this



The x axis of the histogram shows the range of pixel values. Since its an 8 bpp image, that means it has 256 levels of gray or shades of gray in it. Thats why the range of x axis starts from 0 and end at 255 with a gap of 50. Whereas on the y axis, is the count of these intensities.

As you can see from the graph, that most of the bars that have high frequency lies in the first half portion which is the darker portion. That means that the image we have got is darker. And this can be proved from the image too.

Applications of Histograms

Histograms has many uses in image processing. The first use as it has also been discussed above is the analysis of the image. We can predict about an image by just looking at its histogram. Its like looking an x ray of a bone of a body.

The second use of histogram is for brightness purposes. The histograms has wide application in image brightness. Not only in brightness, but histograms are also used in adjusting contrast of an image.

Another important use of histogram is to equalize an image.

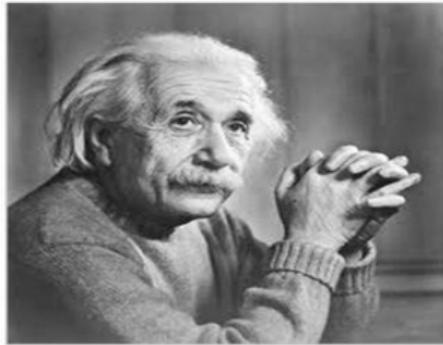
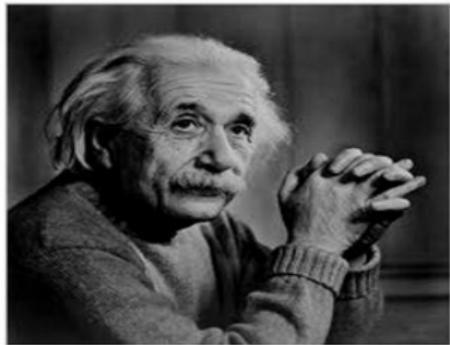
And last but not the least, histogram has wide use in thresholding. This is mostly used in computer vision.

Brightness

Brightness is a relative term. It depends on your visual perception. Since brightness is a relative term, so brightness can be defined as the amount of energy output by a source of light relative to the source we are comparing it to. In some cases we can easily say that the image is bright, and in some cases, its not easy to perceive.

For example

Just have a look at both of these images, and compare which one is brighter.



How to make an image brighter.

Brightness can be simply increased or decreased by simple addition or subtraction, to the image matrix.

Consider this black image of 5 rows and 5 columns



Since we already know, that each image has a matrix at its behind that contains the pixel values. This image matrix is given below.

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

What we will do is, that we will simply add a value of 1 to each of the matrix value of image 1. After adding the image 1 would something like this.



Now we will again compare it with image 2, and see any difference.



We see, that still we cannot tell which image is brighter as both images looks the same.

Now what we will do, is that we will add 50 to each of the matrix value of the image 1 and see what the image has become.

The output is given below.



Now again, we will compare it with image 2.

Image 1



Image 2



Image 1



Image 2



Now when you compare it, you can see that this image1 is clearly brighter than the image 2.

Even it is brighter than the old image1. At this point the matrix of the image1 contains 100 at each index as first add 5, then 50, then 45. So $5 + 50 + 45 = 100$.

Computer Vision-IT813

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

September 7, 2021

Concept of Histogram.

Contrast

Contrast can be simply explained as the difference between maximum and minimum pixel intensity in an image.

For example.

Consider the final image1 in brightness.



The matrix of this image is:

100	100	100	100	100
100	100	100	100	100
100	100	100	100	100
100	100	100	100	100
100	100	100	100	100

Rectangular Snp

The maximum value in this matrix is 100.

The minimum value in this matrix is 100.

Contrast = maximum pixel intensity (subtracted by) minimum pixel intensity

= 100 (subtracted by) 100

= 0

0 means that this image has 0 contrast.

Transformation

Transformation is a function. A function that maps one set to another set after performing some operations.

Digital Image Processing system

We have already seen in the introductory tutorials that in digital image processing, we will develop a system that whose input would be an image and output would be an image too. And the system would perform some processing on the input image and gives its output as an processed image. It is shown below.



Image transformation.

Consider this equation

$$G(x,y) = T\{ f(x,y) \}$$

In this equation,

$F(x,y)$ = input image on which transformation function has to be applied.

$G(x,y)$ = the output image or processed image.

T is the transformation function.

This relation between input image and the processed output image can also be represented as.

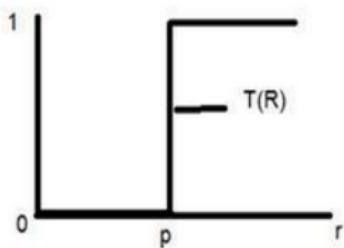
$$s = T(r)$$

where r is actually the pixel value or gray level intensity of $f(x,y)$ at any point. And s is the pixel value or gray level intensity of $g(x,y)$ at any point.

The basic gray level transformation has been discussed in our tutorial of basic gray level transformations.

Examples

Consider this transformation function.



Lets take the point r to be 256, and the point p to be 127. Consider this image to be a one bpp image. That means we have only two levels of intensities that are 0 and 1. So in this case the transformation shown by the graph can be explained as.

All the pixel intensity values that are below 127 (point p) are 0, means black. And all the pixel intensity values that are greater than 127, are 1, that means white. But at the exact point of 127, there is a sudden change in transmission, so we cannot tell that at that exact point, the value would be 0 or 1.

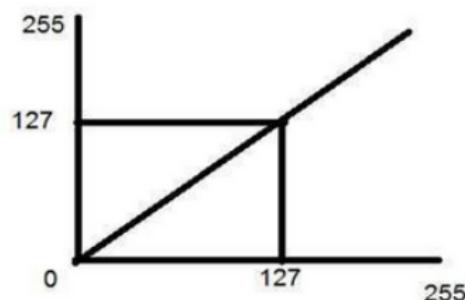
Mathematically this transformation function can be denoted as:

$$\{0 \quad f(x,y) < 127$$

$$g(x,y) =$$

$$\{1 \quad f(x,y) > 127$$

Consider another transformation like this



Now if you will look at this particular graph, you will see a straight transition line between input image and output image.

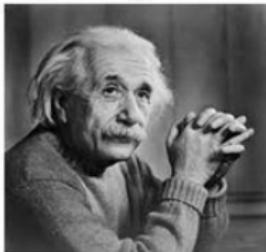
It shows that for each pixel or intensity value of input image, there is a same intensity value of output image. That means the output image is exact replica of the input image.

It can be mathematically represented as:

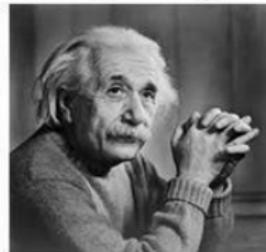
$$g(x,y) = f(x,y)$$

the input and output image would be in this case are shown below.

Input image



Output image



Histogram

Histogram is nothing but a graph that shows frequency of occurrence of data. Histograms has many use in image processing, out of which we are going to discuss one user here which is called histogram sliding.

Histogram sliding

In histogram sliding, we just simply shift a complete histogram rightwards or leftwards. Due to shifting or sliding of histogram towards right or left, a clear change can be seen in the image. In this tutorial we are going to use histogram sliding for manipulating brightness.

The term i-e: Brightness has been discussed in our tutorial of introduction to brightness and contrast. But we are going to briefly define here.

Brightness

Brightness is a relative term. Brightness can be defined as intensity of light emit by a particular light source.

Contrast

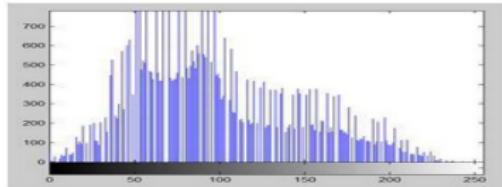
Contrast can be defined as the difference between maximum and minimum pixel intensity in an image.

Sliding Histograms

Increasing brightness using histogram sliding

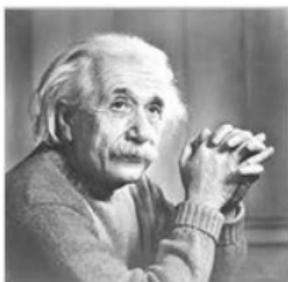


Histogram of this image has been shown below.

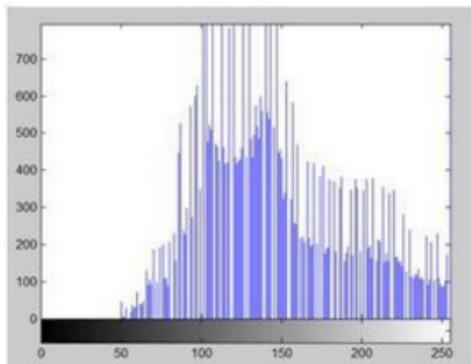


Here what we got after adding 50 to each pixel intensity.

The image has been shown below.

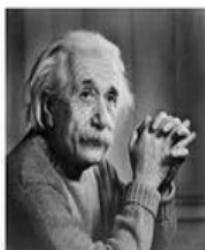


And its histogram has been shown below.



Lets compare these two images and their histograms to see that what change have to got.

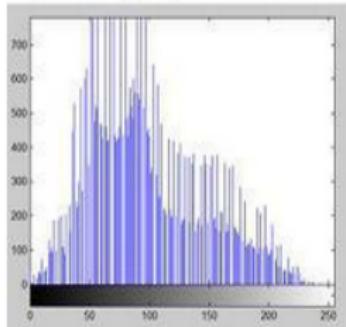
Old image



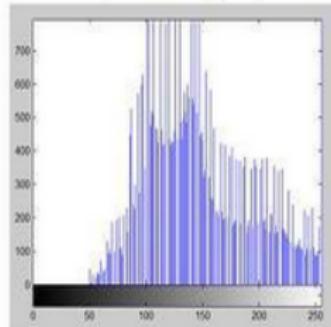
New image



Old histogram



New Histogram

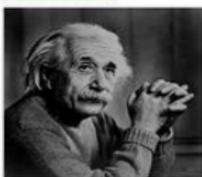


Decreasing brightness using histogram sliding

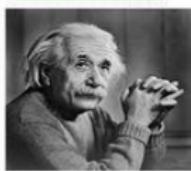
Now if we were to decrease brightness of this new image to such an extent that the old image look brighter, we got to subtract some value from all the matrix of the new image. The value which we are going to subtract is 80. Because we already add 50 to the original image and we got a new brighter image, now if we want to make it darker, we have to subtract at least more than 50 from it.

And this what we got after subtracting 80 from the new image.

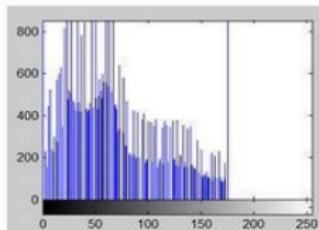
New image.



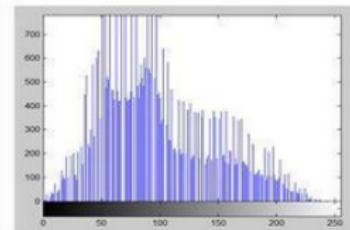
Original image.



New Histogram.



Original Histogram.



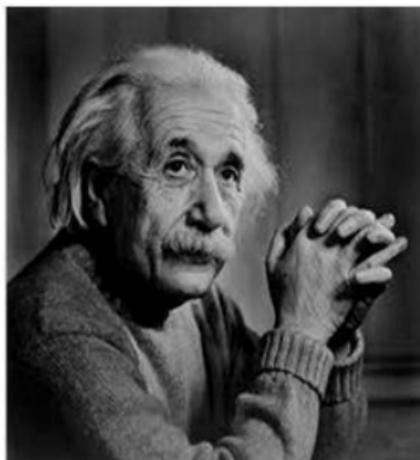
There are two methods of enhancing contrast. The first one is called Histogram stretching that increase contrast. The second one is called Histogram equalization that enhance contrast and it has been discussed in our tutorial of histogram equalization.

Before we will discuss the histogram stretching to increase contrast, we will briefly define contrast.

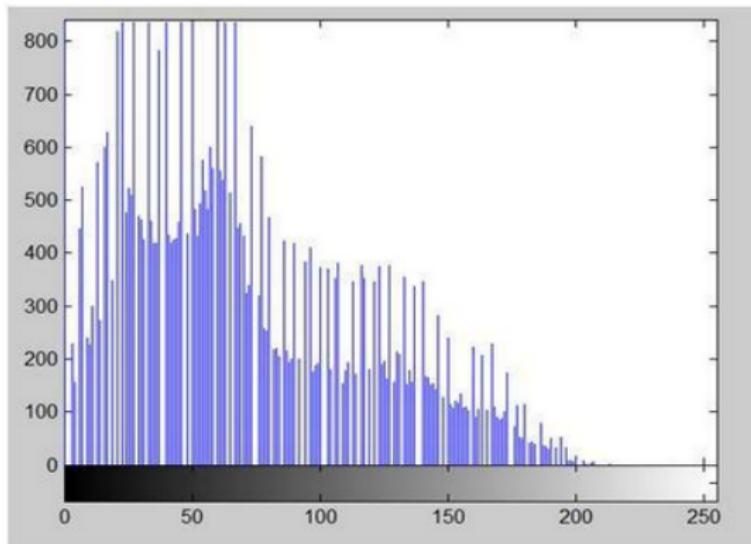
Contrast

Contrast is the difference between maximum and minimum pixel intensity.

Consider this image.



The histogram of this image is shown below.



Now we calculate contrast from this image.

Contrast = 225.

Now we will increase the contrast of the image.

Increasing the contrast of the image

The formula for stretching the histogram of the image to increase the contrast is

$$g(x,y) = \frac{f(x,y) - f_{min}}{f_{max} - f_{min}} * 2^{bpp}$$

The formula requires finding the minimum and maximum pixel intensity multiply by levels of gray. In our case the image is 8bpp, so levels of gray are 256.

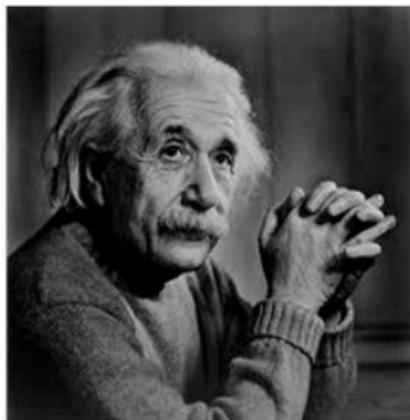
The minimum value is 0 and the maximum value is 225. So the formula in our case is

$$g(x,y) = \frac{f(x,y) - 0}{225 - 0} * 255$$

where $f(x,y)$ denotes the value of each pixel intensity. For each $f(x,y)$ in an image , we will calculate this formula.

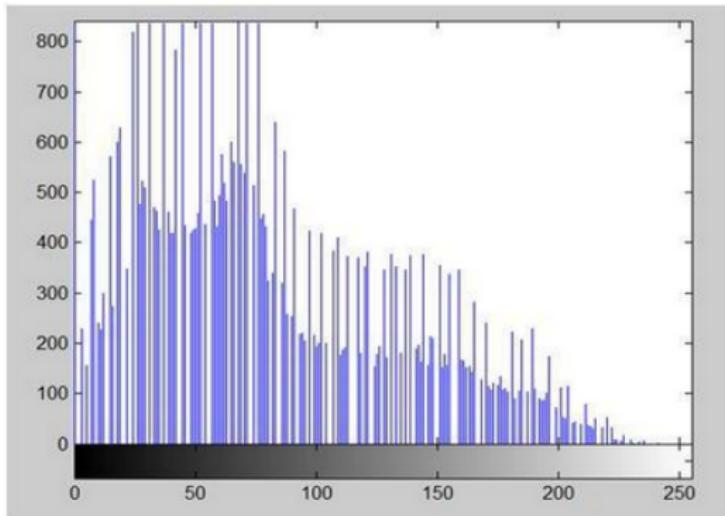
After doing this, we will be able to enhance our contrast.

The following image appear after applying histogram stretching.



The stretched histogram of this image has been shown below.

Note the shape and symmetry of histogram. The histogram is now stretched or in other means expand. Have a look at it.



In this case the contrast of the image can be calculated as

Contrast = 240

Hence we can say that the contrast of the image is increased.

Computer Vision-IT813

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

September 14, 2021

Concept of Histogram Equalization

What is PMF?

PMF stands for probability mass function. As its name suggests, it gives the probability of each number in the data set or you can say that it basically gives the count or frequency of each element.

How PMF is calculated

We will calculate PMF from two different ways. First from a matrix, because in the next tutorial, we have to calculate the PMF from a matrix, and an image is nothing more than a two dimensional matrix.

Then we will take another example in which we will calculate PMF from the histogram.

Consider this matrix.

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

Now if we were to calculate the PMF of this matrix, here how we are going to do it.

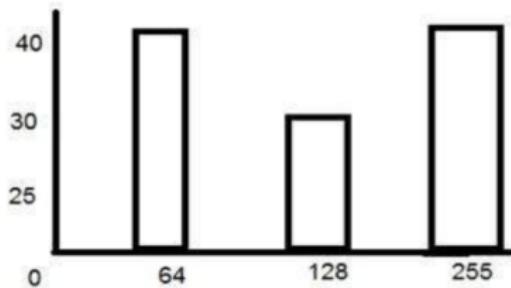
At first, we will take the first value in the matrix , and then we will count, how much time this value appears in the whole matrix. After count they can either be represented in a histogram, or in a table like this below.

PMF

0	2	2/25
1	4	4/25
2	3	3/25
3	3	3/25
4	2	2/25
5	4	4/25
6	3	3/25
7	4	4/25

Note that the sum of the count must be equal to total number of values.

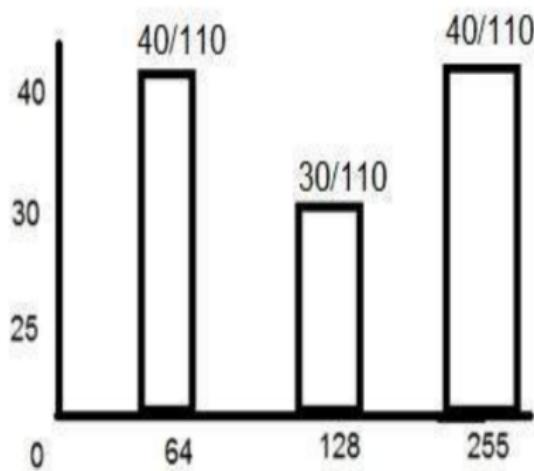
Calculating PMF from histogram



The above histogram shows frequency of gray level values for an 8 bits per pixel image.

Now if we have to calculate its PMF, we will simply look at the count of each bar from vertical axis and then divide it by total count.

So the PMF of the above histogram is this.



Another important thing to note in the above histogram is that it is not monotonically increasing. So in order to increase it monotonically, we will calculate its CDF.

What is CDF?

CDF stands for cumulative distributive function. It is a function that calculates the cumulative sum of all the values that are calculated by PMF. It basically sums the previous one.

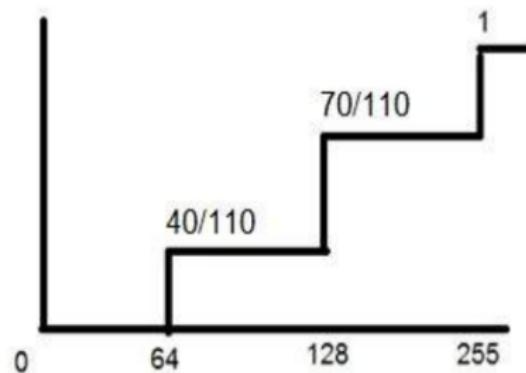
How it is calculated?

We will calculate CDF using a histogram. Here how it is done. Consider the histogram shown above which shows PMF.

Since this histogram is not increasing monotonically, so will make it grow monotonically.

We will simply keep the first value as it is, and then in the 2nd value , we will add the first one and so on.

Here is the CDF of the above PMF function.



Now as you can see from the graph above, that the first value of PMF remain as it is. The second value of PMF is added in the first value and placed over 128. The third value of PMF is added in the second value of CDF , that gives 110/110 which is equal to 1.

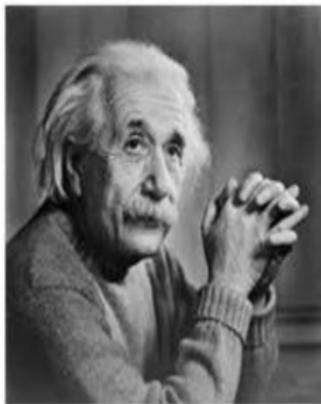
And also now, the function is growing monotonically which is necessary condition for histogram equalization.

Histogram Equalization

Histogram equalization is used to enhance contrast. It is not necessary that contrast will always be increased in this. There may be some cases where histogram equalization can be worse. In those cases the contrast is decreased.

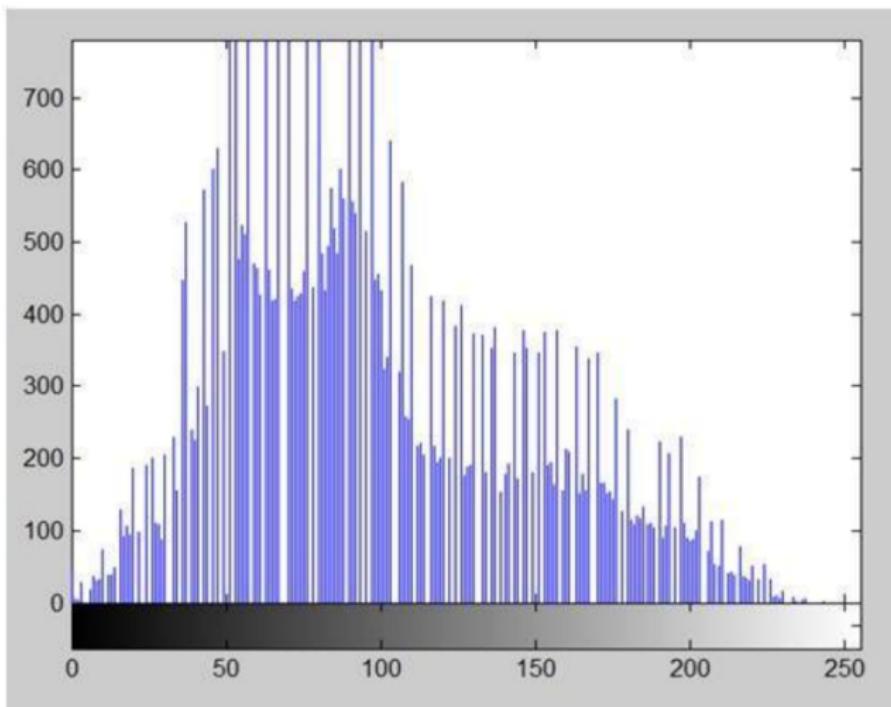
Let's start histogram equalization by taking this image below as a simple image.

Image

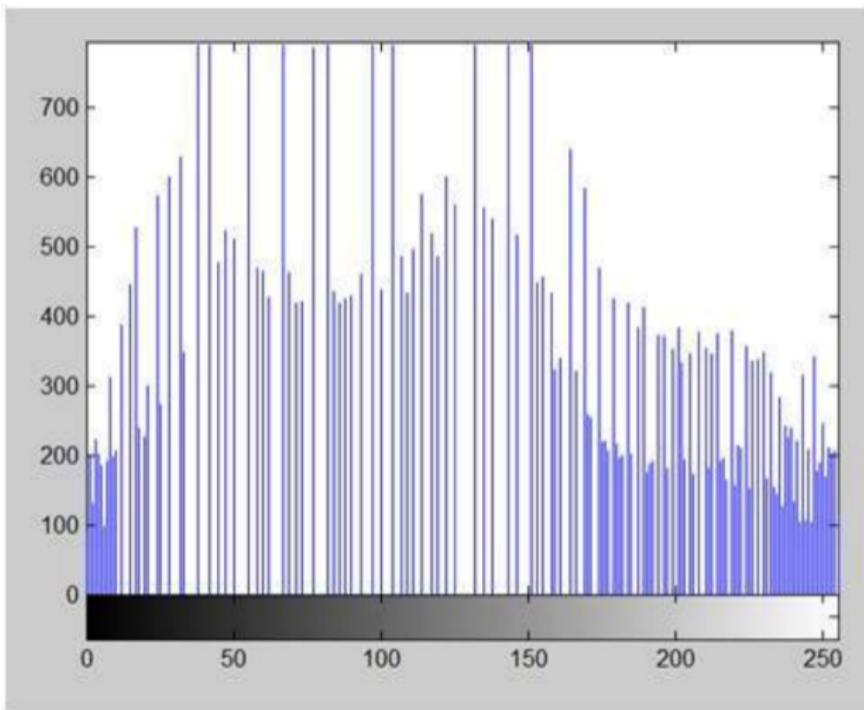


Histogram of this image

The histogram of this image has been shown below.

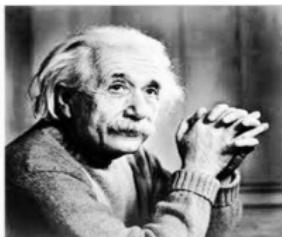


Histogram Equalization histogram

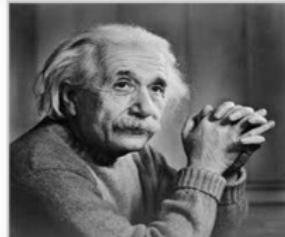


Comparing both the histograms and images

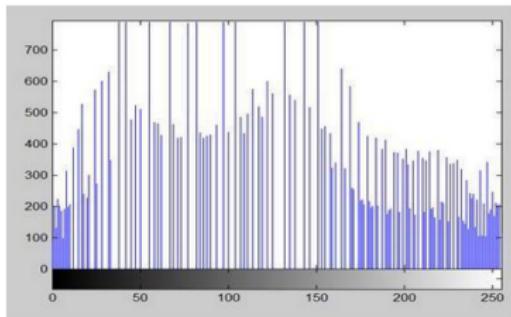
New Image



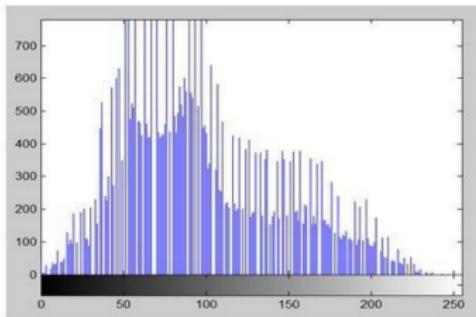
Old image



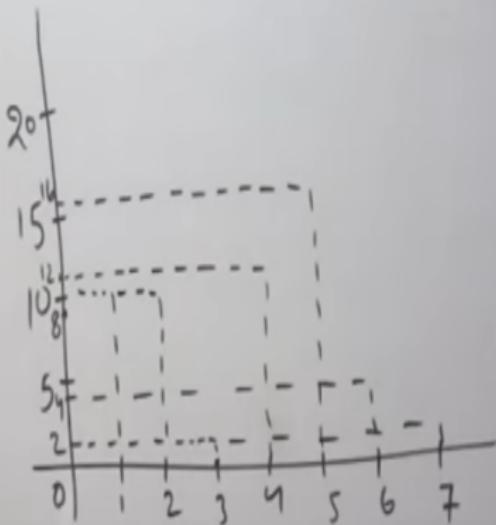
New Histogram



Old Histogram



Gray level (r_k)	0	1	2	3	4	5	6	7
No. of pixels (p_k)	8	10	10	2	12	16	4	2



Gray level (y_k)	0	1	2	3	4	5	6	7
No. of pixels (p_k)	8	10	10	2	12	16	4	2

$$\text{Total no. of pixels} = 8 + 10 + 10 + 2 + 12 + 16 + 4 + 2 = 64.$$

y_k	p_k	p_m	C_m	$L = 7 \times C_m$	Round off	No. of pixels

Q. Perform histogram equalization on the given 8x8 image. The gray level distribution of the image is given below -

Gray level (r_k)	0	1	2	3	4	5	6	7
No. of pixels (p_k)	8	10	10	2	12	16	4	2

$$\text{Total no. of pixels} = 8 + 10 + 10 + 2 + 12 + 16 + 4 + 2 = 64$$

r_k	p_k	p_m	C_m	$L = 7 \times C_m$	Round off	No. of pixels
0	8	0.125	0.125	0.875	1	8
1	10	0.15625	0.25	1.96875	2	10
2	10	0.15625	0.4375	3.0625	3	12
3	2	0.03125	0.46875	3.28125	3	12
4	12	0.1875	0.65625	4.59375	5	12
5	16	0.25	0.90625	6.34375	6	16
6	4	0.0625	0.76875	6.78125	7	6
7	2	0.03125	1	7	7	2

Computer Vision-IT813

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

MORPHOLOGICAL OPERATIONS

Dilation AND Erosion

Morphological operation

- It is a collection of non-linear operations related to the shape or morphology of features in an image.
- ie , we process images according to its shape
- 2 fundamental operations
 - Dilation
 - Erosion

DILATION AND EROSION

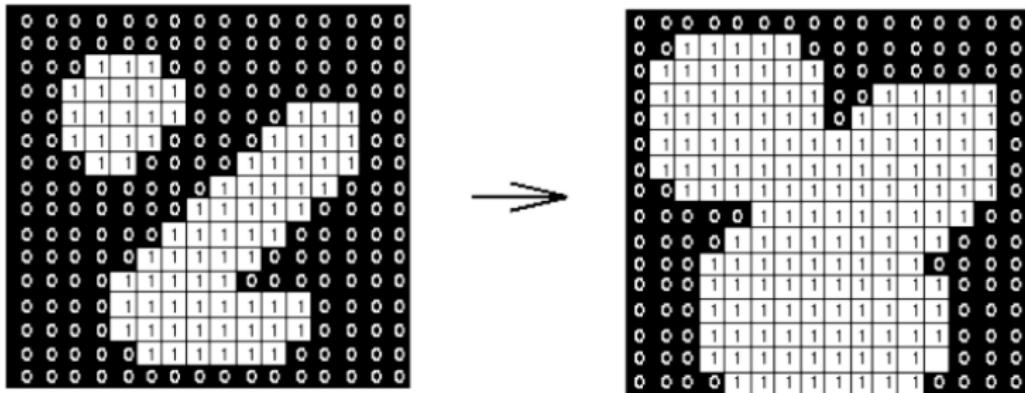
- Dilation adds pixels to the boundaries of objects in an image
- Erosion removes pixels on object boundaries

DILATION

- It grows or thicken objects in a binary image
- Thickening is controlled by a shape referred to as **structuring element**
- **Structuring element** is a matrix of 1's and 0's

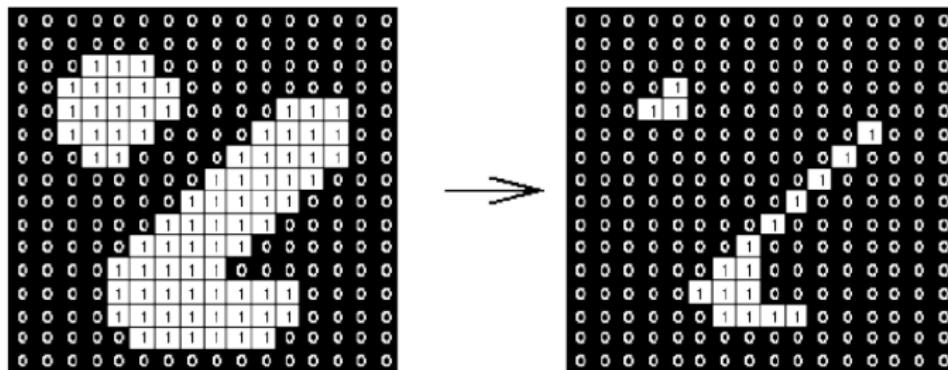
Applications of dilation

For bridging gaps in an image



Effect of dilation using a 3×3 square structuring element

Applications of erosion : Eliminating unwanted detail



Effect of erosion using a 3×3 square structuring element

Opening and Closing operations

- Opening → An Erosion followed by a dilation
- Closing → A dilation followed by an erosion

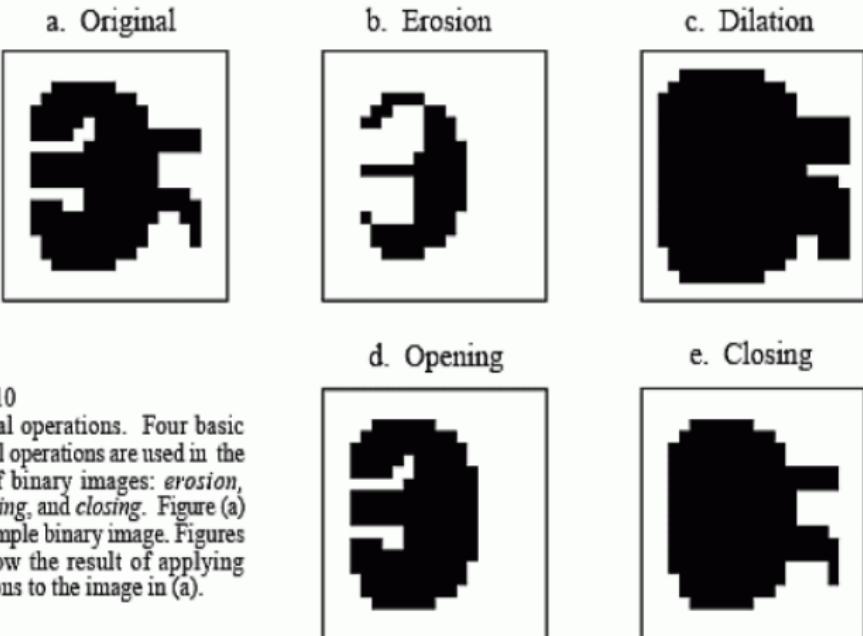
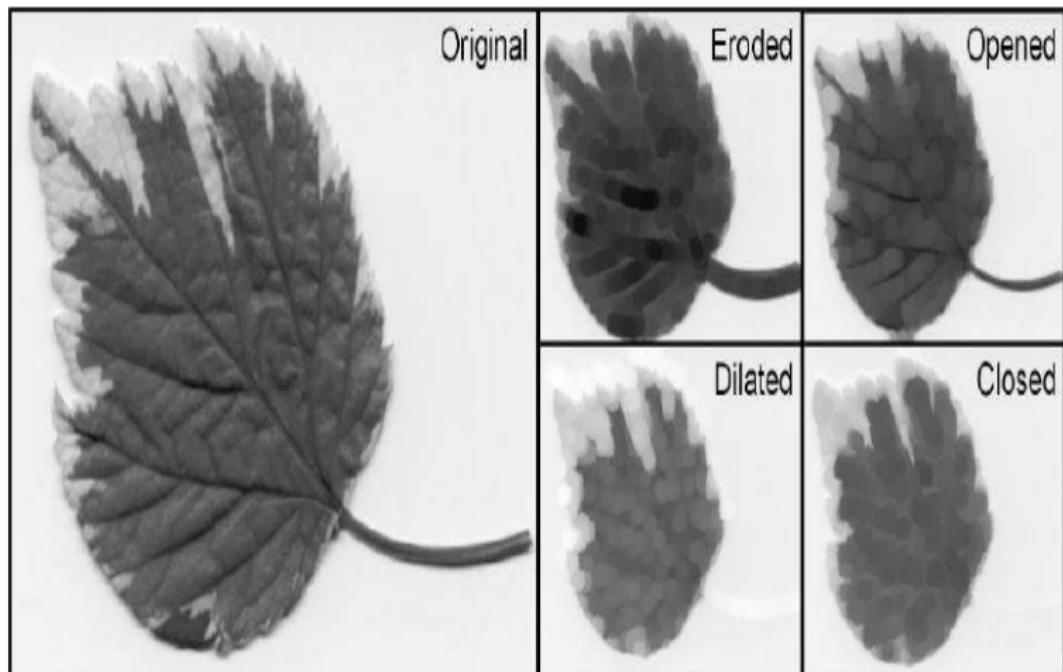


FIGURE 25-10

Morphological operations. Four basic morphological operations are used in the processing of binary images: *erosion*, *dilation*, *opening*, and *closing*. Figure (a) shows an example binary image. Figures (b) to (e) show the result of applying these operations to the image in (a).



morphologySet theory

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

$$V = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)$$

New concepts

1) Reflection

Reflection of v

$$\hat{V} = \{ \omega | \omega = -v \text{ for } v \in V \}$$

e.g.

$$\hat{V} = \{ (-1, -3), (-1, -1), (-2, -2), (-2, -3), (-3, -3), (-3, -1) \}$$

					0 0 0 0
				0	1 1
				0 0 1	1
				0 1 0	1
-4	-3	-2	-1	0 0 0 0	0
0 0 0 0	0 1 2 3				
0 1 0 1	-1				
0 1 1 0	-2				
0 1 1 1	-3				
0 0 0 0	-4				

$$V = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)\}$$

Translation. $L_2 = \{z_1, z_2\}$

$$(V)_2 = \{ c | c = v + z, \text{ for } v \in V \}$$

$$\text{eg. } z = \{1, 1\}$$

$$(V)_2 = \{(2,4), (2,2), (3,3), (3,4), (4,4), (4,3), (4,2)\}$$

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

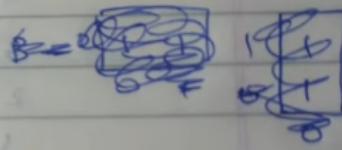
Dilated Dilation

- ① $X \oplus B = \{ p \in \mathbb{Z}^2 \mid p = x + b, x \in X, b \in B \}$
- ② $X \oplus B = \{ p \mid (\hat{B})_p \cap X \neq \emptyset \}$

eg.

Let

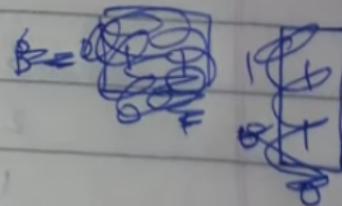
$$X = \begin{matrix} & \begin{matrix} 0 & 0 & 0 & 10 \\ 3 & 0 & 1 & 11 & 1 \\ 2 & 0 & 0 & 11 & 11 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \end{matrix} \end{matrix}$$



$$B = \begin{matrix} 0 & \boxed{1} & 1 \\ 0 & 1 \end{matrix}$$

let

$x = 4$	0	0	0	0
3	0	1	1	1
2	0	0	1	1
1	0	1	0	1
0	0	0	0	0
	0	1	2	3



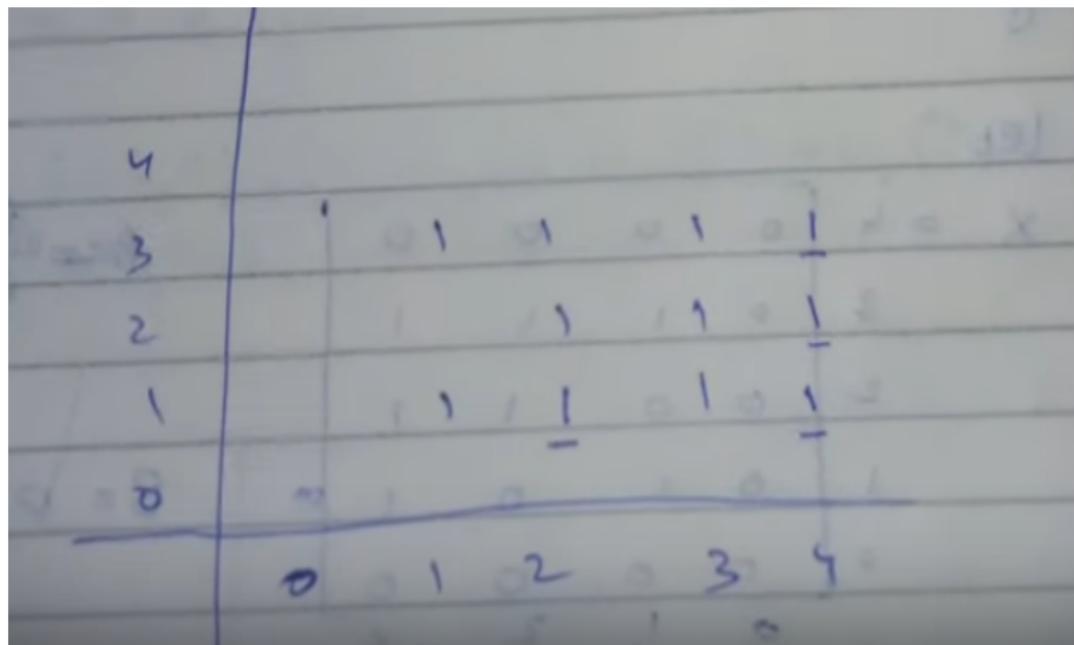
$$B = \{ \begin{matrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{matrix} \}$$

$$X = \{ (1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1) \}$$

$$B = \{ (0,0), (1,0) \}$$

$$X = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2)\} \\ B = \{(0,0), (1,0)\}$$

$$X \oplus B = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1), (2,3), (2,1), (3,2), (3,3), (4,3), (4,2), (4,1)\}$$



Erosion.

$$x \ominus B = \{ p \mid (B)p \subseteq x \}$$

$$x \ominus B = \{ p \in \mathbb{Z}^2 \mid p + b \in x \text{ for every } b \in B \}$$

$$x = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

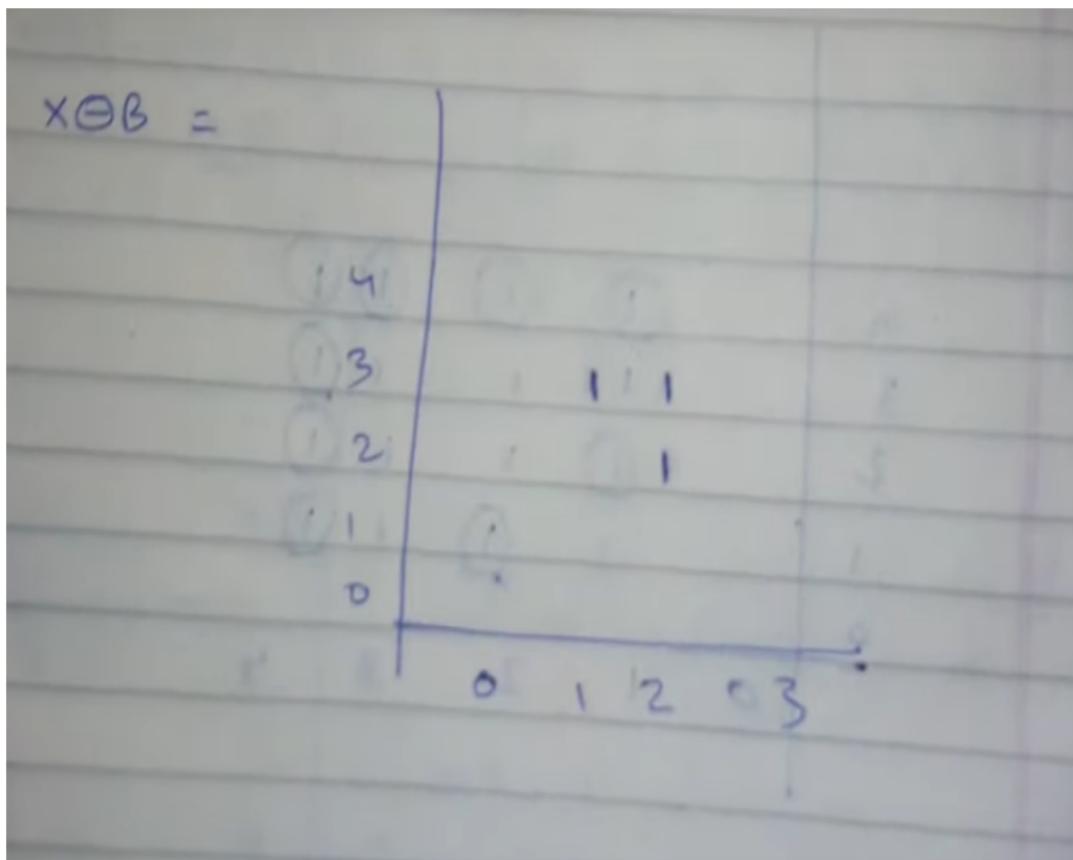
$$B = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$X = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)\}$$

~~X ⊕ B ⊂ X~~

$$(*) B = \{(0,0), (1,0)\}$$

$$X \ominus B = \{(1,3), (2,3), (2,2),\}$$



Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

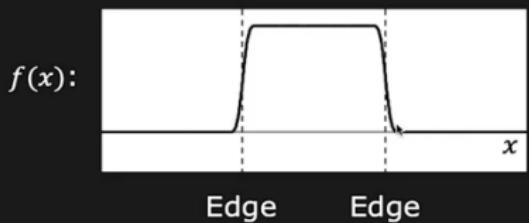
April 5, 2022

Edge detection Using Gradients

- An image gradient is a directional change in the intensity or color in an image.
- The gradient of the image is one of the fundamental building blocks in image processing.
- Mathematically, the gradient of a two-variable function (here the image intensity function) at each image point is a 2D vector with the components given by the derivatives in the horizontal and vertical directions.
- The most common way to approximate the image gradient is to convolve an image with a kernel, such as the Sobel operator or Prewitt operator.

1D Edge Detection

Edge is a rapid change in image intensity in a small region

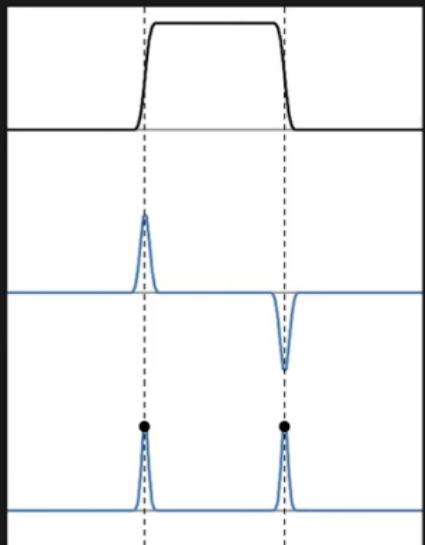


Basic Calculus: **Derivative** of a continuous function represents the amount of change in the function.

Edge Detection Using 1st Derivative

First Derivative: $\frac{\partial f}{\partial x}$

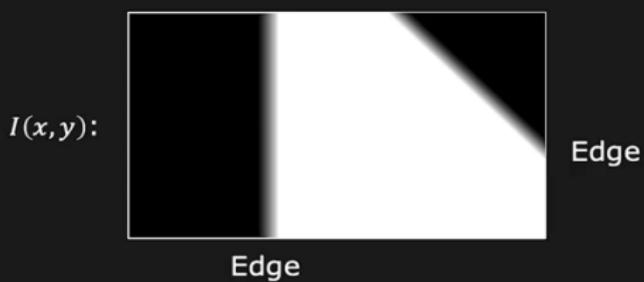
First Derivative Absolute Value: $|\frac{\partial f}{\partial x}|$



Local Extrema
Indicate Edges

Local Maxima
Indicate Edges

2D Edge Detection



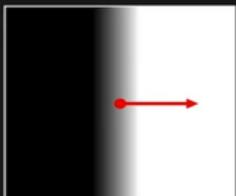
Basic Calculus: Partial Derivatives of a 2D continuous function represents the amount of change along each dimension.

Gradient (∇)

Gradient (Partial Derivatives) represents the direction of most rapid change in intensity

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

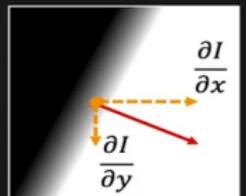
Pronounced as "Del I"



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



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



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

Gradient (∇) as Edge Detector

Gradient Magnitude $S = \|\nabla I\| = \sqrt{\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2}$

Gradient Orientation $\theta = \tan^{-1} \left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x} \right)$

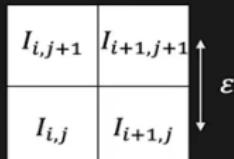


Discrete Gradient (∇) Operator

Finite difference approximations:

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

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



Can be implemented as Convolution!

$$\frac{\partial}{\partial x} \approx \frac{1}{2\varepsilon} \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}$$

$$\frac{\partial}{\partial y} \approx \frac{1}{2\varepsilon} \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}$$

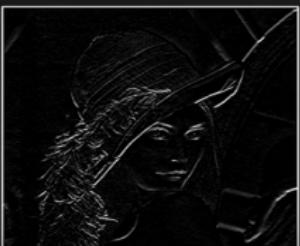
Comparing Gradient (∇) Operators

Gradient	Roberts	Prewitt	Sobel (3x3)	Sobel (5x5)
$\frac{\partial I}{\partial x}$	$\begin{matrix} 0 & 1 \\ -1 & 0 \end{matrix}$	$\begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$	$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$	$\begin{matrix} -1 & -2 & 0 & 2 & 1 \\ -2 & -3 & 0 & 3 & 2 \\ -3 & -5 & 0 & 5 & 3 \\ -2 & -3 & 0 & 3 & 2 \\ -1 & -2 & 0 & 2 & 1 \end{matrix}$
$\frac{\partial I}{\partial y}$	$\begin{matrix} 1 & 0 \\ 0 & -1 \end{matrix}$	$\begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$	$\begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$	$\begin{matrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 3 & 5 & 3 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & -3 & -5 & -3 & -2 \\ -1 & -2 & -3 & -2 & -1 \end{matrix}$

Gradient (∇) Using Sobel Filter



Image (I)



Edge Thresholding

Standard: (Single Threshold T)

$\|\nabla I(x, y)\| < T$ Definitely Not an Edge

$\|\nabla I(x, y)\| \geq T$ Definitely an Edge

Hysteresis Based: (Two Thresholds $T_0 < T_1$)

$\|\nabla I(x, y)\| < T_0$ Definitely Not an Edge

$\|\nabla I(x, y)\| \geq T_1$ Definitely an Edge

$T_0 \leq \|\nabla I(x, y)\| < T_1$ Is an Edge if a Neighboring Pixel is Definitely an Edge

Canny Edge Detection Algorithm

- Step 1 - Grayscale Conversion.
- Step 2 - Gaussian Blur.
- Step 3 - Determine the Intensity Gradients.
- Step 4 - Non Maximum Suppression.
- Step 5 - Double Thresholding.
- Step 6 - Edge Tracking by Hysteresis.
- Step 7 - Cleaning Up.

Computer Vision-IT416

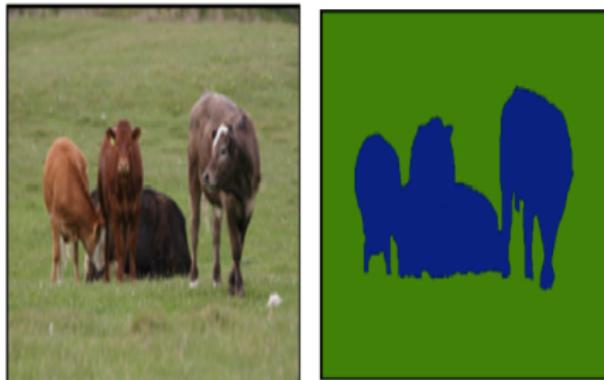
Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

Image segmentation and clustering methods

- Image segmentation may be viewed as a clustering approach in which the pixels, that are satisfying a criterion, are grouped into a cluster while dissatisfying pixels are placed in different groups.



Broad Classification of Image Segmentation

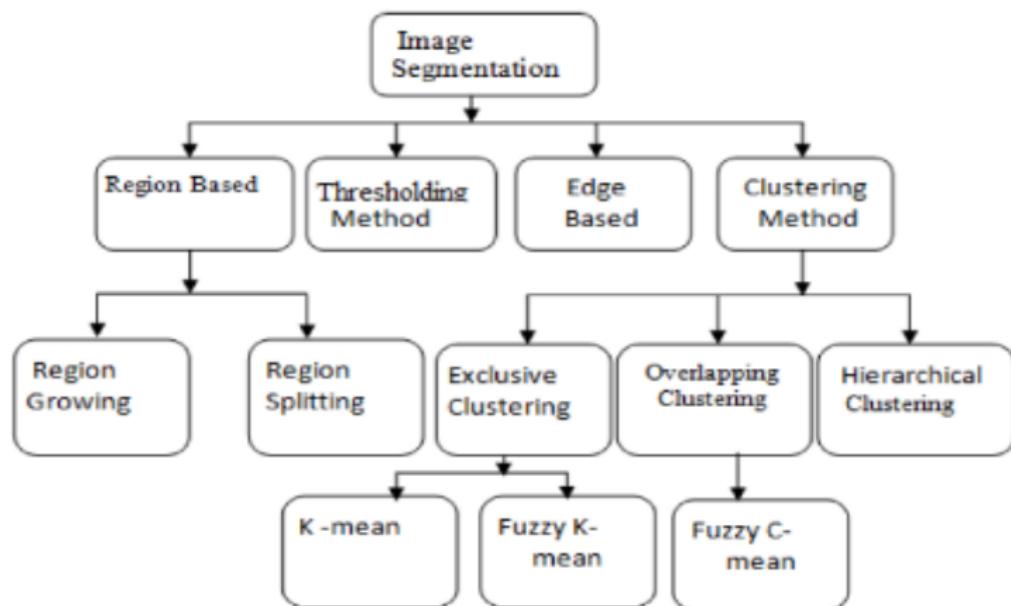


Image segmentation and clustering methods

- Image segmentation is an essential phase of computer vision in which useful information is extracted from an image that can range from finding objects while moving across a room to detect abnormalities in a medical image.
- As image pixels are generally unlabelled, the commonly used approach for the same is clustering.
- Two main clustering methods have been surveyed, namely hierarchical and partitional based clustering methods.
- As partitional clustering is computationally better, further, the partitional based clustering methods into three categories, namely K-means based methods, histogram-based methods, and meta-heuristic based methods.

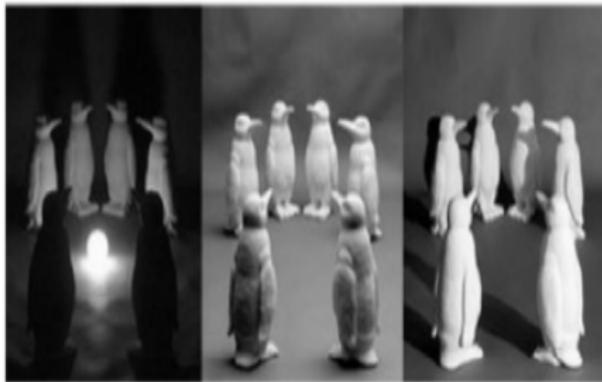
- “A picture is worth a thousand words” is a famous idiom which signifies that processing an image may relieve more information than processing the textual data.
- In computer vision, image segmentation is the prime research area which corresponds to partitioning of an image into its constituent objects or region of interests (ROI)
- Generally, it assembles the image pixels into similar regions. It is a pre-processing phase of many image-based applications like biometric identification, medical imaging, object detection and classification, and pattern recognition. Some of the prominent applications are as follows.
- Content-based image retrieval: It corresponds to the searching of query-relevant digital images from large databases. The retrieval results are obtained as per the contents of the query image. To extract the contents from an image, image segmentation is performed.

- “A picture is worth a thousand words” is a famous idiom which signifies that processing an image may relieve more information than processing the textual data.
- In computer vision, image segmentation is the prime research area which corresponds to partitioning of an image into its constituent objects or region of interests (ROI)
- Generally, it assembles the image pixels into similar regions. It is a pre-processing phase of many image-based applications like biometric identification, medical imaging, object detection and classification, and pattern recognition. Some of the prominent applications are as follows.
- Content-based image retrieval: It corresponds to the searching of query-relevant digital images from large databases. The retrieval results are obtained as per the contents of the query image. To extract the contents from an image, image segmentation is performed.

- Machine vision: It is the image-based technology for robotic inspection and analysis, especially at the industrial level. Here, segmentation extracts the information from the captured image related to a machine or processed material.
- Medical imaging: Today, image segmentation helps medical science in a number of ways from medical diagnosis to medical procedures. Some of the examples include segmentation of tumors for locating them, segmenting tissue to measure the corresponding volumes, and segmentation of cells for performing various digital pathological tasks like cell count, nuclei classification and many others.

- Object recognition and detection: Object recognition and detection is an important application of computer vision. Here, an object may be referred to as a pedestrian or a face or some aerial objects like roads, forests, crops, etc. This application is indispensable to image segmentation as the extraction of the indented object from the image is priorly required.
- Video surveillance: In this, the video camera captures the movements of the region of interests and analysis them to perform an indented task such as identification of the action being performed in the captured video or controlling the traffic movement, counting the number of objects and many more. To perform the analysis, segmentation of the region of interest is foremost required.

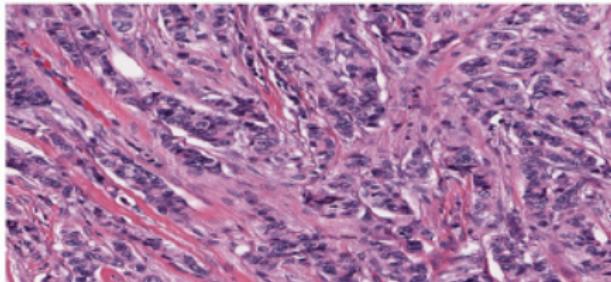
- Though segmenting an image into the constituent ROI may end up as a trivial task for humans, it is relatively complex from the perspective of the computer vision. There are number of challenges which may affects the performance of an image segmentation method.
- Illumination variation: It is a fundamental problem in image segmentation and has severe effects on pixels. This variation occurs due to the different lighting conditions during the image capturing.



- Intra-class variation: One of the major issues in this field is the existence of the region of interest in a number of different forms or appearances. Figure 1b depicts an example of chairs that are shown in different shapes, each having a different appearance. Such intra-class variation often makes the segmentation procedure difficult. Thus, a segmentation method should be invariant to such kind of variations.



- Background complexity: Image with a complex background is a major challenge. Segmenting an image as the region of interests may mingle with the complex environment and constraints. The dark blue color regions in the image represent the nuclei region which is generally defined as the region of interests in histopathological applications like nuclei count or cancer detection. It can be observed that the background is too complex due to which the nuclei regions do not have clearly defined boundaries. Therefore, such background complexities degrade the performance of segmentation methods.



Clustering Method

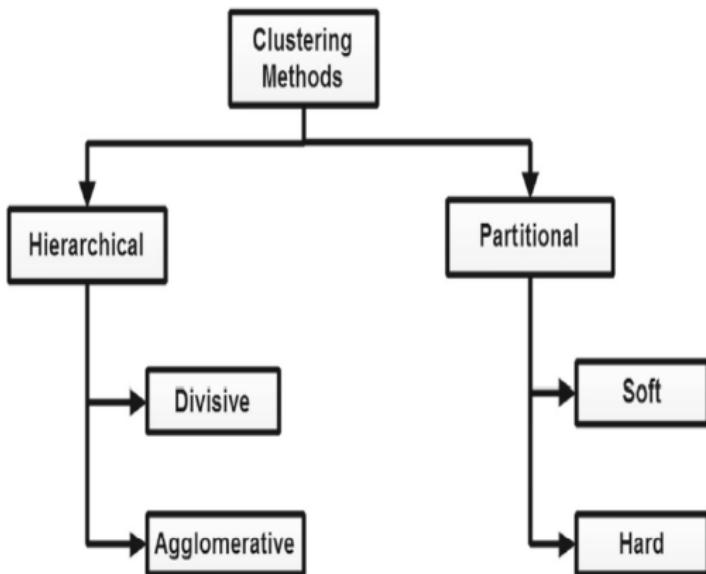
- To mitigate the challenges and learning from unlabelled data, the common approach is to group the data based on certain similarity or dissimilarity measures followed by labelling of each group. This approach of grouping the data is generally termed as clustering.
- The main objective of a clustering method is to classify the unlabelled pixels into homogeneous groups that have maximum resemblance, i.e. to achieve maximum similarity within the clusters and minimum dissimilarity among the clusters.

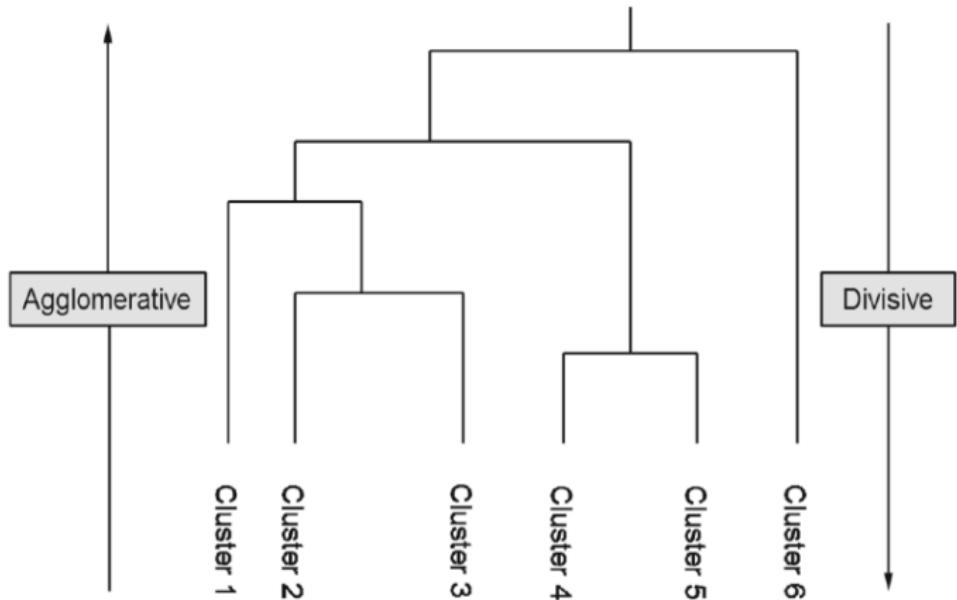
Mathematically, the clustering procedure on an image (X) of size ($m \times n$), defined over d -dimensions, generates K clusters $\{C_1, C_2, \dots, C_K\}$ subject to the following conditions:

- $C_i \neq \emptyset$, for $i = 1, 2, \dots, K$
- $C_i \cap C_j = \emptyset$, for i and $j = 1, 2, \dots, K$ and $i \neq j$
- $\cup_{i=1}^K C_i = X$

- The first condition ensures that there will be at least one pixel in every formed cluster.
- The next condition implies that all the formed clusters will be mutually exclusive, i.e. a pixel will not be assigned to two clusters.
- The last condition states that the data values assigned to all the clusters will represent the complete image.

- In hierarchical clustering, the grouping of data is performed at different similarity levels which is schematized by a tree-like structure termed as a dendrogram





- In divisive clustering, recursive hierarchical data splitting is performed in a top-down fashion to generate the clusters.
- All the data items belong to a single cluster initially. This single cluster is further split into smaller clusters until a termination criteria is satisfied or until each data item forms its own cluster.
- On the other side, the agglomerative clustering is performed in the bottom-up fashion where data points are merged hierarchically to produce clusters.
- Initially, each data item defines itself as a cluster which is further merged into bigger clusters, until a termination criteria is met or until a single cluster is formed consisting of all the data items.

- In general, divisive clustering is more complex than the agglomerative approach, as it partitions the data until each cluster contains a single data item.
- The divisive clustering will be computationally efficient if the each cluster is not partitioned to individual data leaves.
- The time complexity of a naive agglomerative clustering is $O(n^3)$ which can be reduced to $O(n^2)$ using optimization algorithms.
- On the contrary, the time complexity of divisive clustering is $\Omega(n^2)$
- Moreover, divisive clustering is also more accurate since it considers the global distribution of data while partitioning data in top-level partitioning.

- The pseudocode of the divisive and agglomerative clustering approaches are presented in Algorithms 1 and 2 respectively.

Algorithm 1 Agglomerative clustering.

Input: Initialize the number of clusters to be formed.

Output: Clustered data.

Consider each data point as a cluster;

while clustering condition is not satisfied **do**

 Perform merging of two clusters having minimum inter-cluster distance;

end while

Algorithm 2 Divisive clustering.

Input: Initialize the number of clusters to be formed.

Output: Clustered data.

Consider all data points as a single cluster;

while clustering condition is not satisfied **do**

 Divide the cluster into two clusters resulting in the largest inter-cluster distance;

end while

Partitional clustering

- The partitional clustering is relatively popular and preferred over the hierarchical clustering, especially for a large dataset, due to its computational efficiency .
- In this clustering approach, the notion of similarity is used as the measurement parameter.
- Generally, partitional clustering groups the data items into clusters according to some objective function such that data items in a cluster are more similar than the data items in the other clusters.
- To achieve this, the similarity of each data item is measured with every cluster. Moreover, in partitional clustering, the general notion of the objective function is the minimization of the within-cluster similarity criteria which is usually computed by using Euclidean distance.
- The objective function expresses the goodness of each formed cluster and returns the best representation from the generated clusters.

Partitional clustering methods for image segmentation

Sub-categories	Methods	Remarks
Soft	FCM, FCS,	Grouping based on objective function;
	FLAME	Preferred for large datasets.
Hard	Kmeans-based	Low time complexity;
	histogram-based	Number of clusters need to be known priorly;
	Metaheuristic-based	Dependance over the initial clusters.

- Soft clustering methods assign each data to either two or more clusters with a degree of belongingness (or membership) iteratively.
- The degree of belongingness illustrates the level of association among data more reasonably. The belongingness of a data item with a cluster is a continuous value in the interval [0, 1] and depends upon the objective function.

- Particularly, FCM is the most widely used and popular method of this approach. It returns a set of K fuzzy clusters by minimizing the objective function defined in (1)

$$\sum_{i=1}^N \sum_{k=1}^K \mu_{ik}^m \|x_i - v_k\|^2, m \geq 1. \quad (1)$$

where, $\mu_{ik} \in [0, 1]$ and corresponds to membership degree for i^{th} pixel with k^{th} cluster. Equation (1) is optimized iteratively by updating μ_{ik} and v_k according to (2) and (3) respectively.

$$\mu_{ik} = \frac{1}{\sum_{j=1}^K \left(\frac{\|x_i - v_k\|}{\|x_i - v_j\|} \right)^{\frac{2}{m-1}}} \quad (2)$$

$$v_k = \frac{\sum_{i=1}^N \mu_{ik}^m x_i}{\sum_{i=1}^N \mu_{ik}^m} \quad (3)$$

- Normally, the exponent of the fuzzy partition matrix (m) is kept as $m \geq 1$. This regulates the number of pixels that can have membership with more than one cluster.

The pseudo-code of FCM is presented in Algorithm 3

Algorithm 3 Fuzzy C-Means (FCM).

Input: Initialize the number of clusters to be formed.

Output: Clustered data.

Randomly initialize the cluster centroids;

while clustering condition is not satisfied or centroids do not change **do**

 Use (2) to compute the fuzzy partition matrix (U);

 Compute the objective function;

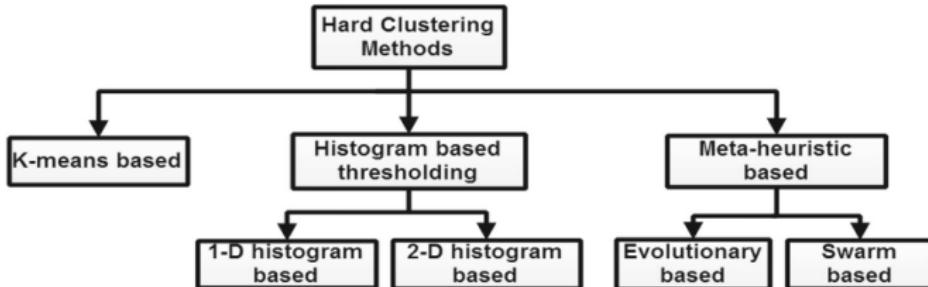
 Update the cluster centroids by (3);

end while

- The inclusion of degree of belongingness benefits soft clustering methods in a number of ways like, relatively high clustering accuracy, faster in generating approximate solutions, and efficient in handling incomplete or heterogeneous data.

Hard clustering methods

- Hard clustering methods iteratively partition the data into disjoint clusters according to the objective function. Generally, the objective function is the sum of squared Euclidean distance between data and associated centroid which is to be minimized.
- Usually, the centre of the clustered data is considered as the centroid of the clusters in these methods. Moreover, in contrast to soft clustering, hard clustering assigns data to a single cluster only i.e., each data will have the degree of belongingness as either 0 or 1.
- Classification of hard clustering methods



Kmeans-based methods

- In Kmeans-based methods, the cluster centroid is updated by taking the mean of all the data items assigned to the corresponding cluster.
- This is iteratively continued until some defined convergence criterion is met.
- Although methods of this category have merits like relatively low time complexity, simple in nature, and guaranteed convergence, there are number of limitations which need to be handled.
- These limitations include the number of clusters to be formed needs to be known priorly, solution quality depends on the initial clusters and number of formed clusters, not appropriate on data having non-convex distribution, follow hill-climbing strategy and hence usually traps into local optima, and relatively sensitive to outliers, noise, and initialization phase.

K-means: K-means [1] partitions a set of data points, $X = \{x_1, \dots, x_n\}$, into a number of clusters (k). It performs partition based on the similarity criteria which is usually the sum of squared error defined in (4).

$$J = \sum_{i=1}^k \text{sum}_{x_j \in X} \|x_j - m_i\|^2 \quad (4)$$

where, m_i is the centroid of cluster i which is collectively represented as $M = \{m_1, \dots, m_k\}$ for corresponding clusters, $C = \{c_1, \dots, c_k\}$. This method iteratively minimizes the criterion function J . Further, the formed clusters C and corresponding centroids M are updated as given by (5) and (6) respectively.

$$x_i \in c_l, \text{ if } l = \operatorname{argmin}_{l=1}^k \|x_j - m_i\|^2 \quad (5)$$

$$m_i = \frac{\sum_{x_j \in c_l} x_j}{|c_l|} \quad (6)$$

for $1 \leq i \leq N$ and $1 \leq l \leq k$.

- K-means method has the time complexity of $O(nkt)$ where, n , k , and t correspond to the number of data items, number of clusters to be formed, and maximum iterations respectively. However, this method is biased towards initial cluster centroids and usually traps into local minima. Moreover, solutions vary with the number of clusters.
- The pseudo-code of the K-means method is presented in Algorithm 4

Algorithm 4 K-means.

Input: Initialize the number of clusters to be formed.

Output: Clustered data.

Randomly select distinct data points as initial cluster centroids;

while clustering condition is not satisfied or centroids do not change **do**

 Compute the objective function, defined in (4);

 Assign each data point to the cluster which is closest;

 Update the cluster centroids;

end while

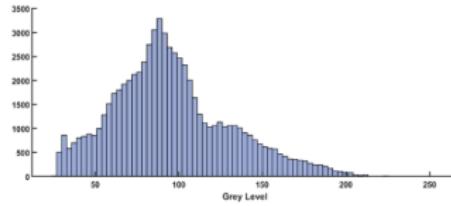
Histogram-based methods

- This category includes methods which perform segmentation by constructing histogram according to the frequency of the intensity values in an image.
- These methods identify a set of optimal threshold values which partitions the histogram. This partitioning results in grouping of intensity values in different clustering. Therefore, histogram-based segmentation methods can be considered as clustering approach.
- Generally, a single feature of an image is considered, usually the intensity, to define a 1-dimensional (1D) histogram.

Multimedia Tools and Applications



a



b

Fig. 6 1D view of grey level histogram of an image taken from BSDS300 [30] a Image b 1D histogram

1D histogram-based segmentation method

To understand the mathematical formulation of 1D histogram-based segmentation method, consider an RGB image of N number of pixels each having intensity from $\{0, 1, 2, \dots, L - 1\}$ in each plane. Suppose, n_i corresponds to the number of pixels for i^{th} intensity level. Therefore, the probability distribution (p_i) of i^{th} intensity level in the image is defined as the probability of occurrence of the image pixels with i^{th} intensity level which is formulated as (7).

$$p_i = \frac{n_i}{N}, \quad 0 \leq i \leq L - 1 \quad (7)$$

The mean intensity of an image plane is determined by (8).

$$\mu = \sum_{i=1}^L i p_i \quad (8)$$

To group the image pixels into n clusters $\{C_1, C_2, \dots, C_n\}$, $n - 1$ thresholds, $(t_1, t_2, \dots, t_{n-1})$, are required which are represented as (9).

$$v(x, y) = \begin{cases} 0, & v(x, y) \leq t_1 \\ \frac{t_1+t_2}{2}, & t_1 < v(x, y) \leq t_2 \\ \vdots & \vdots \\ \frac{t_{n-2}+t_{n-1}}{2}, & t_{n-2} < v(x, y) \leq t_{n-1} \\ L - 1, & v(x, y) > t_{n-1} \end{cases} \quad (9)$$

where $v(x, y)$ corresponds to pixel intensity at (x, y) location of a $M \times N$ image. The partition C_j , $1 \leq j \leq n$ consists of pixels with intensity value more than t_{j-1} and less than equals to t_j . The frequency of cluster C_j for each plane is computed as (10).

$$w_j = \sum_{i=t_{j-1}+1}^{t_j} p_i \quad (10)$$

The mean for the cluster C_j can be calculated by (11) and the inter-class variance is formulated in (12).

$$\mu_j = \sum_{i=t_{j-1}+1}^{t_j} i p_i / w_j \quad (11)$$

$$\sigma^2 = \sum_{j=1}^n w_j (\mu_j - \mu)^2 \quad (12)$$

To group the pixels into clusters based on intensity, the maximization of inter-class variance [(12)] is considered. Therefore, the objective function tries to maximize the fitness function, defined in (13):

$$\phi = \max_{1 < t_1 < \dots < t_{n-1} < L} \{\sigma^2(t)\} \quad (13)$$

- The performance of 1D histogram-based methods is generally unsatisfactory as they consider only single feature like intensity level information of an image and do not deal with spatial correlation among the pixels which is an important parameter for segmentation.
- To achieve the same, a new histogram has been introduced for image segmentation, termed as 2D-histogram, which has been proven to be better.
- A 2D-histogram considers two features of an image at a time. In literature, these features are selected from a set of three image features, namely pixel intensity, average pixel intensity, and pixel gradient.
- Generally, the segmentation methods based on the histogram are efficient as they require only a single pass through the pixels to construct the histogram.

Metaheuristic-based methods

- This category involves the use of metaheuristic-based approaches to obtain optimal clusters by updating random solutions according to mathematical formulation and optimality criteria (or objective function).
- Generally, a meta-heuristic method solves an optimization problem whose objective function is to perform either maximization or minimization of a cost function ($f(x)$) with a given set of constraints.
- The mathematically formulation of a maximization optimization problem is presented in (14).

$$\text{Maximize}_{\{x \in \mathbb{R}^d\}} f(x) \quad (14)$$

$$\text{such that : } a_i(x) \geq 0, \quad (15)$$

$$b_j(x) = 0, \quad (16)$$

$$c_k(x) \leq 0 \quad (17)$$

where $x = (x_1, x_2, x_3, \dots, x_d)^T$ is the set of decision variables which is defined over d -dimensions and \mathbb{R}^d is the search space of the problem. $a_i(x)$, $b_j(x)$, and $c_k(x)$ correspond to the different constraints applicable to an optimization problem. Actual constraints depend on the considered optimization problem.

Metaheuristic-based methods

- Each meta-heuristic algorithm mimics a particular natural phenomena which may belong to evolutionary, physical, or biological.
- In literature, two common aspects that are often found in these algorithms are exploration and exploitation.
- Exploration represents the diversification in the search space wherein the existing solutions are updated with the intention of exploring the search space. This helps in exploring the new solutions, prevents the stagnation problem, and responsible for achieving the global solution.
- The exploitation, which corresponds to the intensification of the current solution, performs the local search around the currently generated solutions. In this, the goal is to exploit the search space and responsible for convergence to the optimal solution.
- Generally, meta-heuristic algorithms may broadly be classified into two categories, namely evolutionary and swarm algorithms.

Metaheuristic-based methods

- Evolutionary-based algorithms are based on evolution theories such as Darwin's evolutionary theory. The evolutionary algorithms work on the principle of generating better individuals with the course of generation by combining the best individuals of the current generation.
- Genetic algorithm (GA), evolutionary strategy (ES), differential evolution (DE) , biogeography-based optimization (BBO), and probability-based incremental learning (PBIL).
- On the other side, swarm-based algorithms behave like the swarm of agents, such as fishes or birds, to achieve optimal results.
- Some algorithms of this category are particle swarm optimization (PSO) , ant colony optimization (ACO) , gravitational search algorithm (GSA) ,spider monkey optimization (SMO), grey-wolf optimizer (GWO) , cuckoo search (CS) , and military dog based optimizer (MDO).

The pseudo-code for a metaheuristic-based clustering method is presented in Algorithm 5

- Generally, these clustering-based methods are better than other clustering methods in terms of independence from the initial parameter settings and return global optimal solution.

Algorithm 5 Metaheuristic-based clustering method.

Input: Initialize the number of clusters to be formed.

Output: Clustered data.

Choose a set of random population where each individual of the population corresponds to a cluster centroid.

while clustering condition is not satisfied or centroids do not change **do**

 Compute the fitness value of each solution through the considered objective function;

 Update the solution according to the meta-heuristic algorithm;

 Assign each data point to the nearest cluster centroid;

end while

Performance evaluation parameters

- The performance evaluation of a method is necessary to assure the validity of a method. This section lists out various performance measures that researchers have found useful for the quantitative evaluation of an image segmentation method.
- Confusion matrix (CM): It is a widely used representation for the assessing the efficiency of a classification method. However, it can be used to analyze the results of a clustering method too.
- In context of clustering, the number of correctly clustered patterns (true positive (TP), true negative (TN)) and wrongly clustered patterns (false positive (FP), false negative (FN)) can be easily identified from the confusion matrix.
- The confusion matrix (CM) of size NxN represents that there are N classes (or clusters).

- Based on CM, precision, recall and accuracy can also be computed using (18) – (20).

$$\text{Precision} = \frac{TP}{TP + FP} \quad (18)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (19)$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (20)$$

- Intersection of Union (IoU): Intersection of Union is the ratio of the number of common pixels between X and Y to the total number of pixels in X and Y. Here, X and Y correspond to the segmented image and ground truth respectively. The formulation of IoU is depicted in (21).

$$IoU = \frac{|X \cap Y|}{|X| + |Y|} \quad (21)$$

Dice-Coefficient (DC): Dice-Coefficient is defined as twice the number of common pixels divided by total number of pixels in X and Y, where X corresponds to the segmented image and Y is the ground truth. DC is mathematically defined as (22).

$$DC = \frac{2|X \cap Y|}{|X| + |Y|} \quad (22)$$

Boundary Displacement Error (BDE): This parameter computes the average boundary pixels displacement error between two segmented images as depicted in (23). The error of one boundary pixel is defined as it's distance from the closest pixel in the other boundary image.

$$\mu_{LA}(u, v) = \begin{cases} \frac{u-v}{L-1} & 0 < u - v \\ 0 & u - v < 0 \end{cases} \quad (23)$$

Probability Rand Index (PRI): Probability Rand Index finds labelling consistency between the segmented image and its ground truth. It counts such fraction of pairs of pixels and average the result across all ground truths of a given image as shown in (24).

$$R = \frac{a + b}{a + b + c + d} = \frac{a + b}{n} \quad (24)$$

Variation of Information (VOI): Variation of Information as shown in (25) computes the randomness in one segmentation in terms of the distance from given segmentation.

$$VOI(X; Y) = H(X) + H(Y) - 2I(X, Y) \quad (25)$$

Given a set of n elements $S = \{o_1, \dots, o_n\}$ and two partitions of S to compare, $X = \{X_1, \dots, X_r\}$, a partition of S into r subsets, and $Y = \{Y_1, \dots, Y_s\}$, a partition of S into s subsets, define the following:

- a , the number of pairs of elements in S that are in the **same** subset in X and in the **same** subset in Y
- b , the number of pairs of elements in S that are in **different** subsets in X and in **different** subsets in Y
- c , the number of pairs of elements in S that are in the **same** subset in X and in **different** subsets in Y
- d , the number of pairs of elements in S that are in **different** subsets in X and in the **same** subset in Y

The Rand index, R , is:^{[1][2]}

$$R = \frac{a + b}{a + b + c + d} = \frac{a + b}{\binom{n}{2}}$$

Intuitively, $a + b$ can be considered as the number of agreements between X and Y and $c + d$ as the number of disagreements between X and Y .

Since the denominator is the total number of pairs, the Rand index represents the frequency of occurrence of agreements over the total pairs, or the probability that X and Y will agree on a randomly chosen pair.

$\binom{n}{2}$ is calculated as $n(n - 1)/2$.

Similarly, one can also view the Rand index as a measure of the percentage of correct decisions made by the algorithm. It can be computed using the following formula:

$$RI = \frac{TP + TN}{TP + FP + FN + TN}$$

where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives, and FN is the number of false negatives.

Variation of Information (VOI): Variation of Information as shown in (25) computes the randomness in one segmentation in terms of the distance from given segmentation.

$$VOI(X; Y) = H(X) + H(Y) - 2I(X, Y) \quad (25)$$

Global Consistency Error (GCE): A refinement in one segmentation over the other is depicted by the value of GCE as given in (26). If two segmentations are related this way then they are considered as consistent, i.e. both can represent the same natural image segmentation at different scales.

$$GCE = \frac{1}{n} \left\{ \sum_i E(s_1, s_2, p_i), \sum_i E(s_2, s_1, p_i) \right\} \quad (26)$$

Structural Similarity Index (SSIM): Structural Similarity Index measures the similarity between two images by taking initial uncompressed or distortion-free image as the reference and computed as (27). It incorporates important perceptual phenomena such as luminance masking and contrast masking.

$$SSIM = \frac{(2 \times \bar{x} \times \bar{y} + c_1)(2 \times \sigma_{xy} + c_2)}{(\sigma_x^2 + \sigma_y^2) \times ((\bar{x})^2 + (\bar{y})^2 + c_1)} \quad (27)$$

Feature Similarity Index (FSIM): Feature Similarity Index is a quality score that uses the phase congruency (PC), which is a dimensionless measure and shows the significance of a local structure. It is calculated by (28).

$$FSIM = \frac{\sum_{x \in \Omega} S_L(x) \cdot PC_m(x)}{\sum_{x \in \Omega} PC_m(x)} \quad (28)$$

Root Mean squared error (RMSE): The root-mean-squared error computes the difference between sample value predicted by a model or an estimator and actual value. The formulation is shown in (29).

$$RMSE(\hat{\theta}) = \sqrt{MSE(\hat{\theta})} = \sqrt{E((\hat{\theta} - \theta)^2)} \quad (29)$$

Peak Signal to Noise Ratio (PSNR in dB): Peak signal-to-noise ratio is defined as the ratio between the maximum possible power of a signal and the power of corrupting noise and is calculated using (30). In general, a higher value of PSNR represents high quality reconstruction and is defined via MSE.

$$PSNR = 10 \log_{10} \frac{(2^n - 1)^2}{\sqrt{MSE}} \quad (30)$$

Normalized Cross-Correlation (NCC): Normalized cross-correlation is used for template matching where images are first normalized due to lighting and exposure conditions. It is calculated by subtracting the mean of original and segmented images from the corresponding images, and divided by their standard deviations. Let $t(x, y)$ is the segmented image of $f(x, y)$, then NCC is calculated by (31).

$$\frac{1}{n} \sum_{x,y} \frac{1}{\sigma_f \sigma_t} (f(x, y) - \bar{f})(t(x, y) - \bar{t}) \quad (31)$$

Average Difference (AD): This parameter represents the average difference between the pixel values and is computed by (32).

$$AD = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N (x(i, j) - y(i, j)) \quad (32)$$

Maximum Difference (MD): This parameter finds the maximum of error signal by taking the difference between original image and segmented image and defined in (33).

$$MD = \max |x(i, j) - y(i, j)| \quad (33)$$

Normalized Absolute Error (NAE): The normalized absolute difference between the original and corresponding segmented image gives NAE and is calculated by (34).

$$NAE = \frac{\sum_{i=1}^M \sum_{j=1}^N |x(i, j) - y(i, j)|}{\sum_{i=1}^M \sum_{j=1}^N |x(i, j)|} \quad (34)$$

- In the above mentioned parameters, IoU, DC, SSIM, FSIM, PRI, PSNR, and NCC show better segmentation on high values. A high value indicates that the segmented image is more close to the ground truth.
- To measure the same, these parameters compute values by considering the region of intersection between the segmented image and the ground truth which corresponds to matching the number of similar pixels.
- On the contrary, other indices prefer lower values for better segmentation as these measures compute error between the segmented image and the ground truth and aim at reducing the difference between them.

Computer Vision-IT813

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

Image Segmentation

- Region-based segmentation is a technique for determining the region directly.
- Region growing is a simple region-based image segmentation method. It is also classified as a pixel-based image segmentation method since it involves the selection of initial seed points.

- Applications-Finding tumors ,veins etc in medical images , finding targets in satellite/aerial images , finding people in surveillance images , summarizing video etc
- Methods- Thresholding , k-means clustering etc

- Region growing is a procedure that groups pixels or sub regions into larger regions.
- The simplest of these approaches is pixel aggregation, which starts with a set of seed points and from these grows regions by appending to each seed points those neighboring pixels that have similar properties (such as gray level, texture, color, shape).
- Region growing based techniques are better than the edge-based techniques in noisy images where edges are difficult to detect

- Region growing methods can correctly separate the regions that have the same properties we define.
- Region growing methods can provide the original images which have clear edges with good segmentation results.
- The concept is simple. We only need a small number of seed points to represent the property we want, then grow the region.

- Computationally expensive
- It is a local method with no global view of the problem.
- Sensitive to noise.
- Unless the image has had a threshold function applied to it, a continuous path of points related to color may exist which connects any two points in the image.

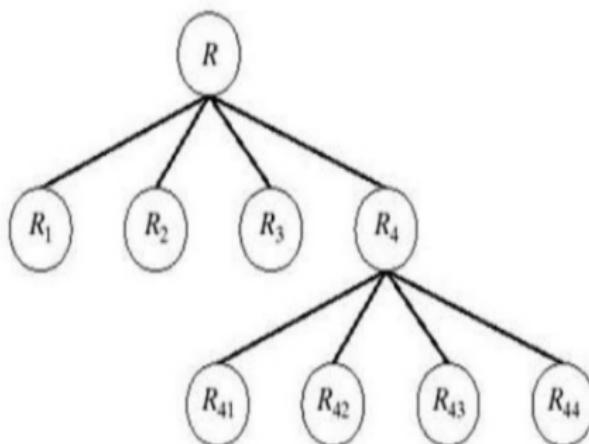
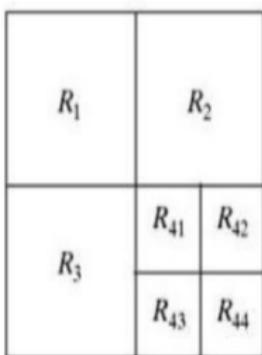
REGION SPLITTING

- Region growing starts from a set of seed points.
- An alternative is to start with the whole image as a single region and subdivide the regions that do not satisfy a condition of homogeneity.

REGION MERGING

- Region merging is the opposite of region splitting.
- Start with small regions (e.g. 2x2 or 4x4 regions) and merge the regions that have similar characteristics (such as gray level, variance).
- Typically, splitting and merging approaches are used iteratively

REGION SPLITTING AND MERGING



Region growing

2	3	F	5	3	3	3
5	N	2	3	7	2	F A
5	6	6	7	7	6	P
6	7	6	7	5	5	4
6	6	F	4	4	3	2
5	4	5	4	2	3	4
0	3	2	3	2	5	6
0	0	0	0	2	2	5
1	1	0	1	0	3	4
1	0	1	0	2	3	5
						4

Region growing

threshold = 3

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

$$\begin{array}{r} a/a \\ \hline 566 \\ 6a \quad 6 \\ 6a \quad 6 \\ 6a \quad 6 \\ 5a \quad a \\ \hline 0 \end{array} \quad \begin{array}{r} a/a \\ \hline 777 \\ 5a \quad 5a \\ 5a \quad 5a \\ 5a \quad 5a \\ 23a \quad 4 \\ \hline 32 \end{array} \quad \begin{array}{r} a/a \\ \hline 66 \\ 5a \quad 5a \\ 5a \quad 5a \\ 4a \quad 4 \\ \hline 4 \end{array}$$

Handwritten annotations:

- A circled '7' is written above the first column of the first row.
- A circled '4' is written above the second column of the third row.
- A circled '0' is written above the second column of the fifth row.
- A circled '1' is written above the first column of the sixth row.
- A circled '1' is written above the second column of the sixth row.

Below the first two columns of the first row, there is a small square containing '0 0 0 0'. Below the first two columns of the fifth row, there is a small square containing '2 3'. Below the first two columns of the sixth row, there is a small square containing '1 0'.

Region split & merge.

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

Region split & merge.

$$7 - 4 = 3$$

$$\text{Inrest} \leq 3$$

5

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

Region split & merge.

$$7 - 4 = 3$$

$$\text{unrest} \leq 3$$

5

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

morphology

Set theory.

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

$$V = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)\}$$

New concepts

1) Reflection

Reflection of v

$$\hat{V} = \{ \omega | \omega = -v \text{ for } v \in V \}$$

Ex.

$$\hat{V} = \{ (-1, -3), (-1, -1), (-2, -2), (-2, -3), (-3, -3), (-3, -1) \}$$

					0 0 0 0
				0	1 1
				0 0 1	1
				0 1 0	1
-4	-3	-2	-1	0 0 0 0	0
0 0 0 0	0 1 2 3				
0 1 0 1	-1				
0 1 1 0	-2				
0 1 1 1	-3				
0 0 0 0	-4				

$$V = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)\}$$

Translation. $L_2 = \{z_1, z_2\}$

$$(V)_2 = \{ c | c = v + z, \text{ for } v \in V \}$$

$$\text{eg. } z = \{1, 1\}$$

$$(V)_2 = \{(2,4), (2,2), (3,3), (3,4), (4,4), (4,3), (4,2)\}$$

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

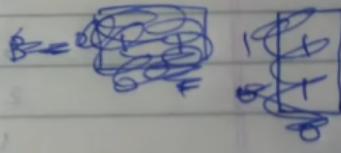
Dilated Dilation

- $$\textcircled{1} \quad X \oplus B = \{ p \in \mathbb{Z}^2 \mid p = x + b, x \in X, b \in B \}$$
- $$\textcircled{2} \quad X \oplus B = \{ p \mid (\hat{B})_p \cap X \neq \emptyset \}$$

eg.

Let

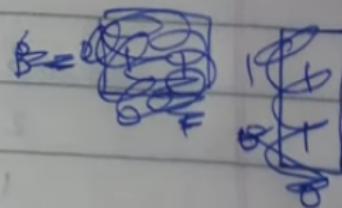
$$X = \begin{matrix} & \begin{matrix} 0 & 0 & 0 & 10 \end{matrix} \\ \begin{matrix} 3 \\ 2 \\ 1 \\ 0 \end{matrix} & \left| \begin{matrix} 0 & 1 & 11 & 1 \\ 0 & 0 & 11 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{matrix} \right| \end{matrix}$$



$$B = \begin{matrix} & \begin{matrix} 1 & 1 \\ 0 & 1 \end{matrix} \\ \begin{matrix} 0 \\ 1 \end{matrix} & \end{matrix}$$

let

$x = 4$	0	0	0	0
3	0	1	1	1
2	0	0	1	1
1	0	1	0	1
0	0	0	0	0
	0	1	2	3



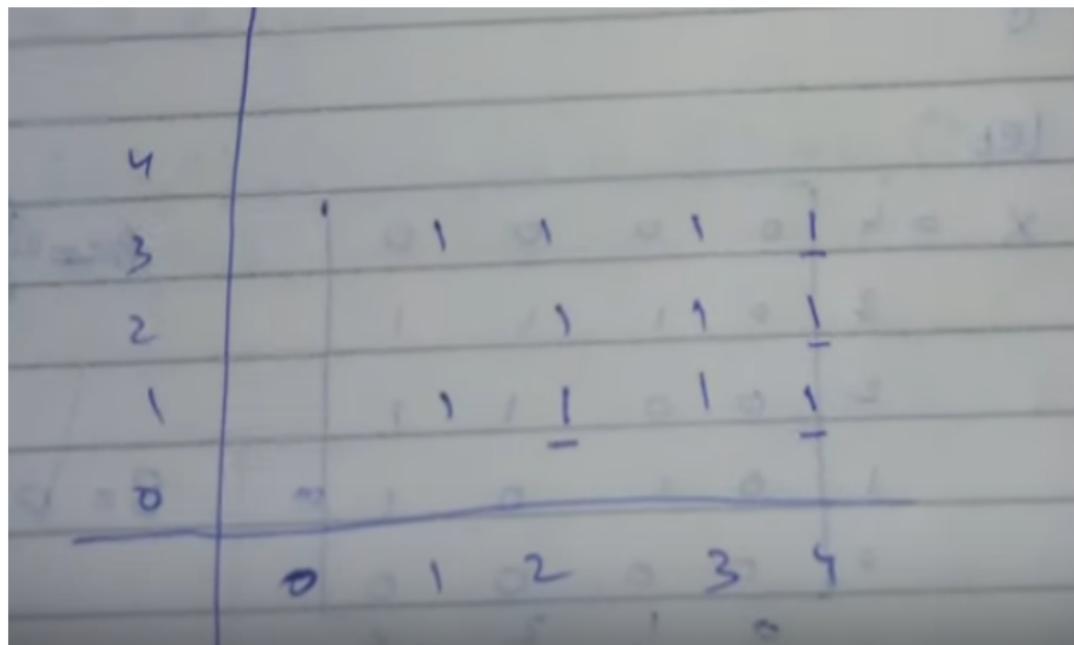
$$B = \{ \begin{matrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{matrix} \}$$

$$X = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)\}$$

$$B = \{(0,0), (1,0)\}$$

$$X = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2)\} \\ B = \{(0,0), (1,0)\}$$

$$X \oplus B = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1), (2,3), (2,1), (3,2), (3,3), (4,3), (4,2), (4,1)\}$$



Erosion.

$$x \ominus B = \{ p \mid (B)p \subseteq x \}$$

$$x \ominus B = \{ p \in \mathbb{Z}^2 \mid p + b \in x \text{ for every } b \in B \}$$

$$x = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

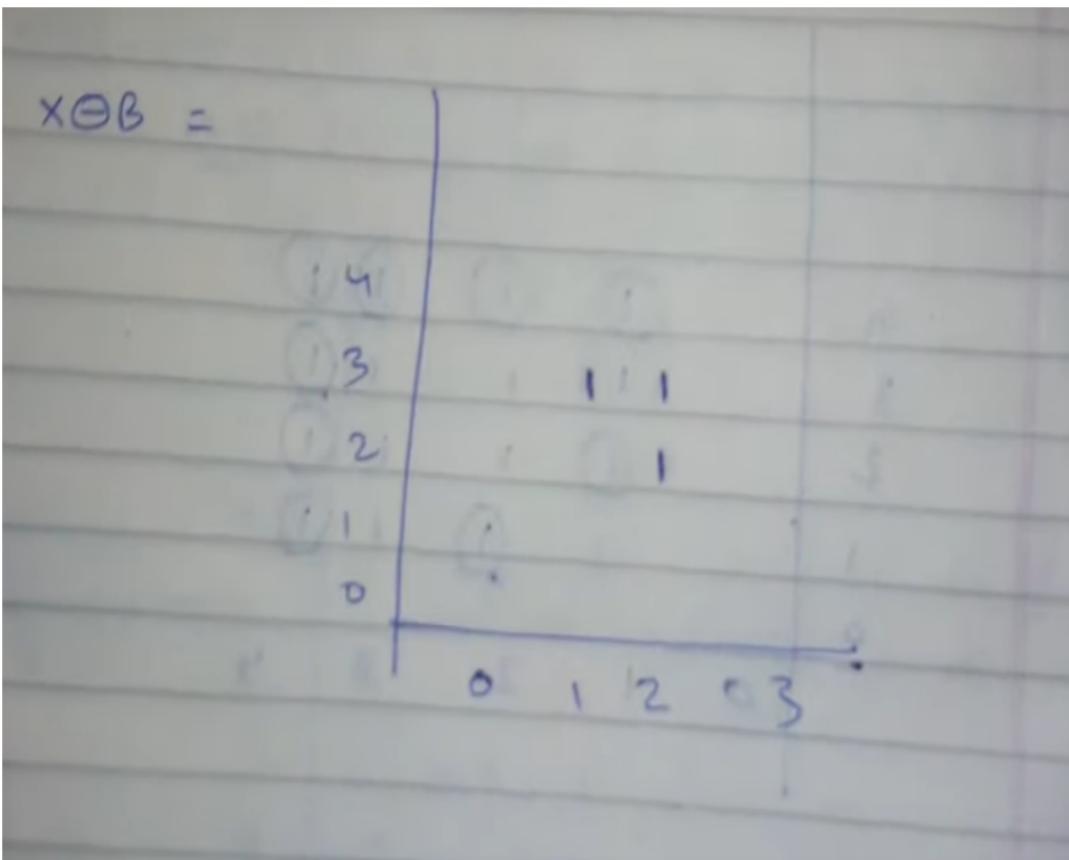
$$B = \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$X = \{(1,3), (1,1), (2,2), (2,3), (3,3), (3,2), (3,1)\}$$

~~X ⊕ B ⊂ X~~

$$\text{B} = \{(0,0), (1,0)\}$$

$$X \ominus B = \{(1,3), (2,3), (2,2)\}$$



Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

Boundary Detection

We need to find Object Boundaries from Edge Pixels.

Topics:

- (1) Fitting Lines and Curves to Edges
- (2) Active Contours (Snakes)
- (3) The Hough Transform
- (4) Generalized Hough Transform

Preprocessing Edge Images



Edge
Detection



Thresholding



Shrink
& Expand

Manually Sketched

Boundary
Detection

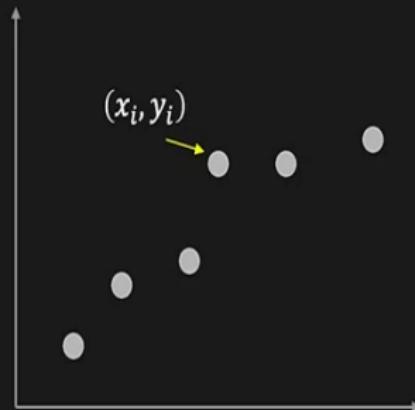


Thinning



Fitting Lines to Edges

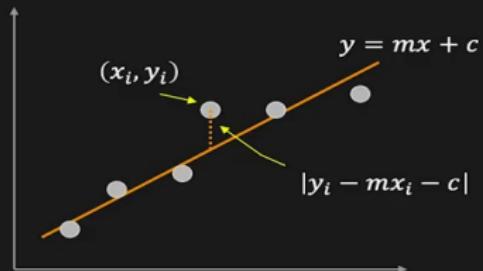
Given: Edge Points (x_i, y_i)



Fitting Lines to Edges

Given: Edge Points (x_i, y_i)

Task: Find (m, c)



Minimize: Average Squared Vertical Distance

$$E = \frac{1}{N} \sum_i (y_i - mx_i - c)^2$$

Least Squares Solution:

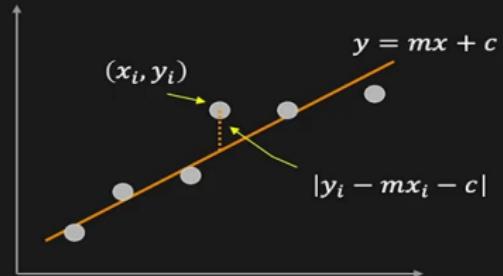
$$\frac{\partial E}{\partial m} = \frac{-2}{N} \sum_i x_i (y_i - mx_i - c) = 0$$

$$\frac{\partial E}{\partial c} = \frac{-2}{N} \sum_i (y_i - mx_i - c) = 0$$

Fitting Lines to Edges

Given: Edge Points (x_i, y_i)

Task: Find (m, c)



Solution:

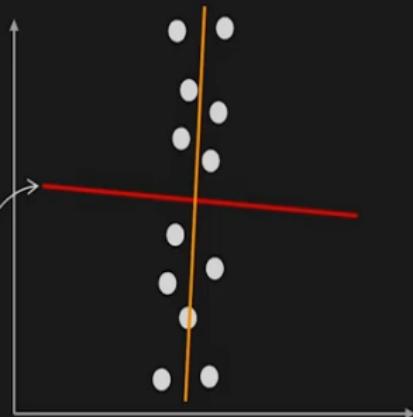
$$m = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \quad c = \bar{y} - m\bar{x}$$

where: $\bar{x} = \frac{1}{N} \sum_i x_i$ $\bar{y} = \frac{1}{N} \sum_i y_i$

Fitting Lines to Edges

Problem: When the points represent a vertical line.

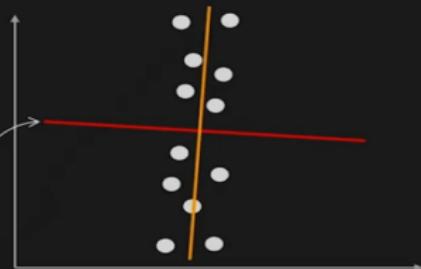
Line that minimizes E !



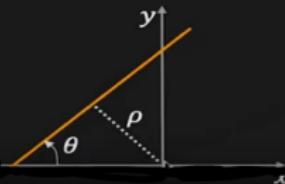
Fitting Lines to Edges

Problem: When the points represent a vertical line.

Line that minimizes E!



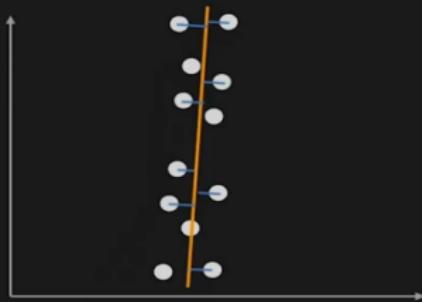
Solution: Use a different line equation



$$x \sin \theta - y \cos \theta + \rho = 0$$

Fitting Lines to Edges

Problem: When the points represent a vertical line.



Minimize: Average Squared Perpendicular Distance

$$E = \frac{1}{N} \sum_i \left(\frac{x_i \sin \theta - y_i \cos \theta + \rho}{\text{Perpendicular Distance}} \right)^2$$

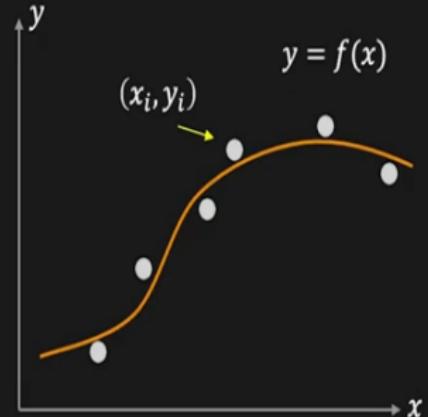
Fitting Curves to Edges

Given: Edge Points (x_i, y_i)

Task: Find polynomial

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

that best fits the points



Fitting Curves to Edges

Given: Edge Points (x_i, y_i)

Task: Find polynomial

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

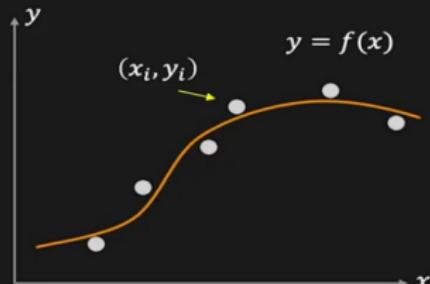
that best fits the points

Minimize:

$$E = \frac{1}{N} \sum_i (y_i - ax_i^3 - bx_i^2 - cx_i - d)^2$$

Solve the Linear System Using Least Squares Fit by:

$$\frac{\partial E}{\partial a} = 0 \quad \frac{\partial E}{\partial b} = 0 \quad \frac{\partial E}{\partial c} = 0 \quad \frac{\partial E}{\partial d} = 0$$



Fitting Curves to Edges

Solving as a Linear System:

$$y_0 = ax_0^3 + bx_0^2 + cx_0 + d$$

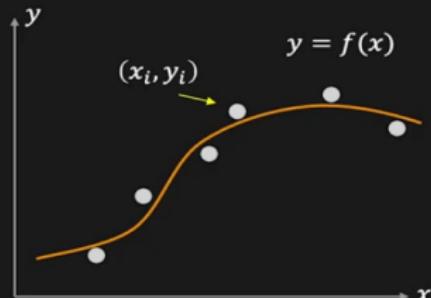
$$y_1 = ax_1^3 + bx_1^2 + cx_1 + d$$

⋮

$$y_i = ax_i^3 + bx_i^2 + cx_i + d$$

⋮

$$y_n = ax_n^3 + bx_n^2 + cx_n + d$$



Given many (x_i, y_i) 's, this is an over-determined linear system with four unknowns (a, b, c, d) .

Solving a Linear System

An over-determined linear system with m unknowns $\{a_j\}$ ($j = 0, \dots, m$) and n observations $\{(x_{ij}, y_i)\}$ ($i = 0, \dots, n$) ($n > m$) can be written in a matrix form.

$$\left[\begin{array}{cccc} x_{00} & x_{01} & \dots & x_{0m} \\ x_{10} & x_{11} & \dots & x_{1m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n0} & x_{n1} & \dots & x_{nm} \end{array} \right] \left[\begin{array}{c} a_0 \\ a_1 \\ \vdots \\ a_m \end{array} \right] = \left[\begin{array}{c} y_0 \\ y_1 \\ \vdots \\ y_n \end{array} \right]$$

$X_{n \times m}$
Known

$\mathbf{a}_{m \times 1}$
Unknown

$\mathbf{y}_{n \times 1}$
Known

$\left. \right\} X\mathbf{a} = \mathbf{y}$
 $X_{n \times m}$ is not a square matrix and hence not invertible.

Least Squares Solution:

$$X^T X \mathbf{a} = X^T \mathbf{y} \Rightarrow \mathbf{a} = (X^T X)^{-1} X^T \mathbf{y} \quad X^+ = (X^T X)^{-1} X^T$$

$\mathbf{a} = X^+ \mathbf{y}$

(Pseudo Inverse)

Difficulties for the Fitting Approach

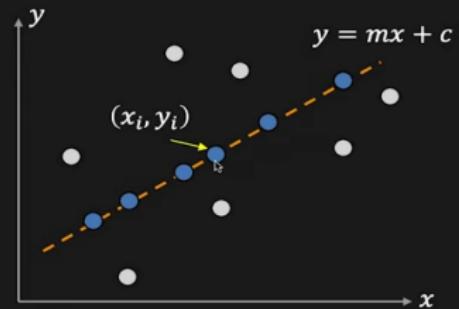


- Extraneous Data: Which points to fit to?
- Incomplete Data: Only part of the model is visible.
- Noise

Hough Transform: Line Detection

Given: Edge Points (x_i, y_i)

Task: Detect line
 $y = mx + c$



Consider point (x_i, y_i)

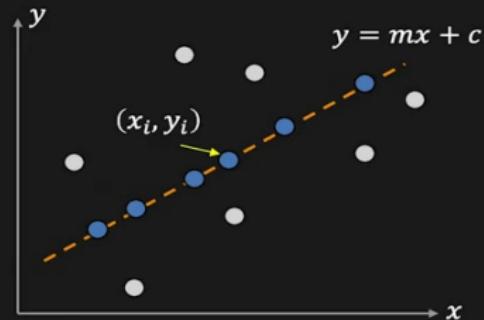
$$y_i = mx_i + c$$

Hough Transform: Line Detection

Given: Edge Points (x_i, y_i)

Task: Detect line

$$y = mx + c$$



Consider point (x_i, y_i)

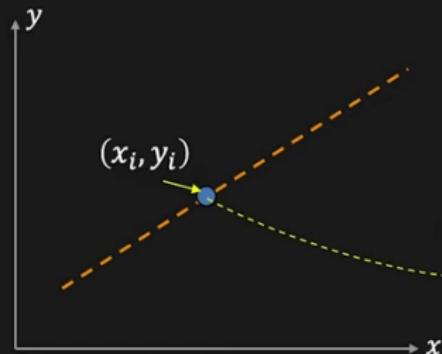
$$y_i = mx_i + c$$

$$\iff$$

$$c = -mx_i + y_i$$

Hough Transform: Concept

Image Space



$$y_i = mx_i + c$$

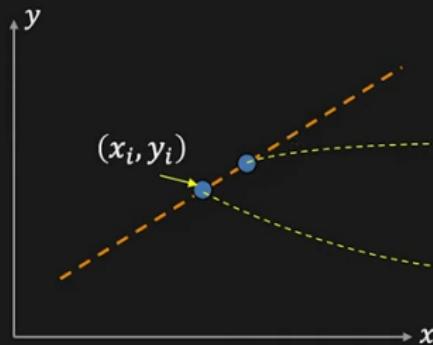
Parameter Space



$$c = -mx_i + y_i$$

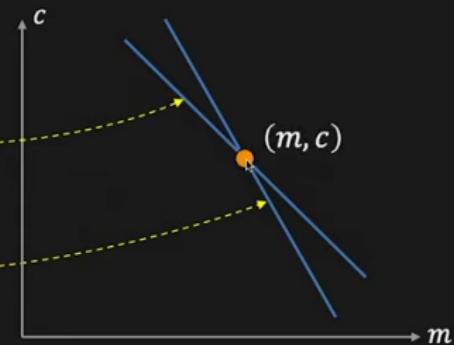
Hough Transform: Concept

Image Space



$$y_i = mx_i + c$$

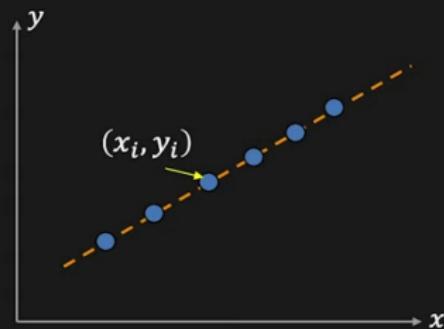
Parameter Space



$$c = -mx_i + y_i$$

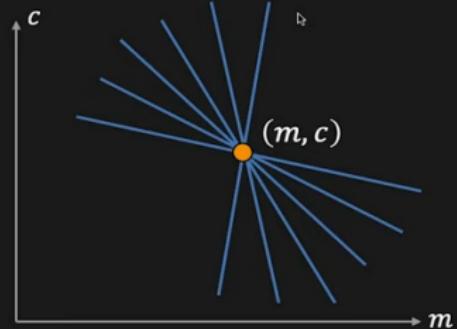
Hough Transform: Concept

Image Space



$$y_i = mx_i + c$$

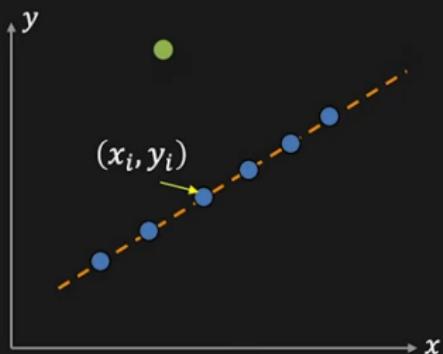
Parameter Space



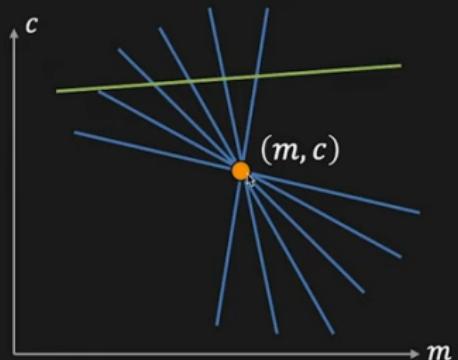
$$c = -mx_i + y_i$$

Hough Transform: Concept

Image Space

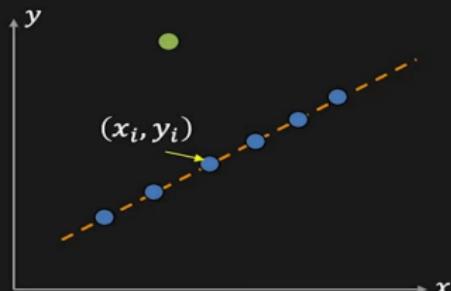


Parameter Space



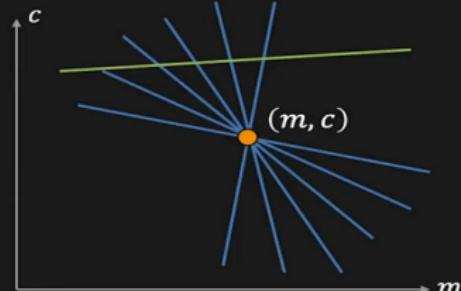
Hough Transform: Concept

Image Space



$$y_i = mx_i + c$$

Parameter Space



$$c = -mx_i + y_i$$

Point

Line

Line

Point

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

Step 4. For each edge point (x_i, y_i) ,

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

if (m, c) lies on the line: $c = -mx_i + y_i$



m	$A(m, c)$				
	0	0	0	0	0
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

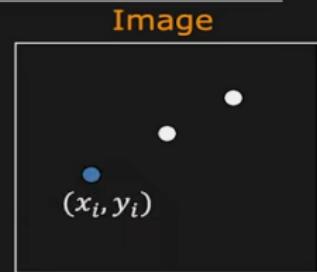
Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

Step 4. For each edge point (x_i, y_i) ,

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

if (m, c) lies on the line: $c = -mx_i + y_i$



$A(m, c)$

m	1	0	0	0	0
0	1	0	0	0	
0	0	1	0	0	
0	0	0	1	0	
0	0	0	0	1	

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

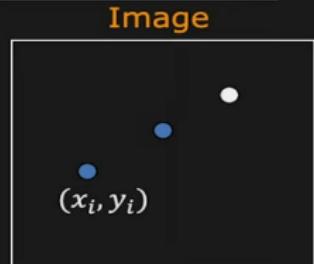
Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

Step 4. For each edge point (x_i, y_i) ,

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

if (m, c) lies on the line: $c = -mx_i + y_i$



c	$A(m, c)$				
1	0	0	0	1	
0	1	0	1	0	
0	0	2	0	0	
0	1	0	1	0	
1	0	0	0	1	

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

Step 4. For each edge point (x_i, y_i) ,

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

if (m, c) lies on the line: $c = -mx_i + y_i$



c	$A(m, c)$				
1	0	0	0	1	
0	1	0	1	0	
1	1	3	1	1	
0	1	0	1	0	
1	0	0	0	1	

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

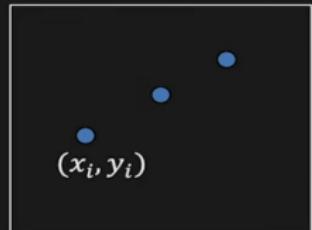
Step 4. For each edge point (x_i, y_i) ,

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

if (m, c) lies on the line: $c = -mx_i + y_i$

Step 5. Find local maxima in $A(m, c)$

Image



$A(m, c)$

c	1	0	0	0	1
1	0	1	0	1	0
0	1	1	3	1	1
1	0	0	0	1	0
0	1	0	0	0	1

Line Detection Algorithm

Step 1. Quantize parameter space (m, c)

Step 2. Create **accumulator array** $A(m, c)$

Step 3. Set $A(m, c) = 0$ for all (m, c)

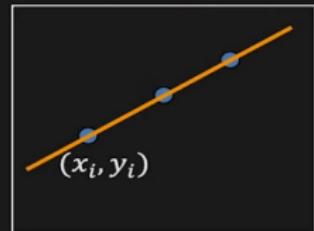
Step 4. For each edge point (x_i, y_i) ,

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

if (m, c) lies on the line: $c = -mx_i + y_i$

Step 5. Find local maxima in $A(m, c)$

Image

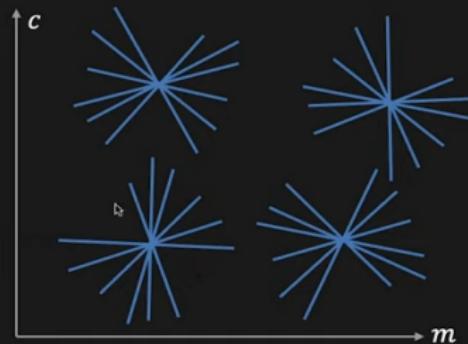


	$A(m, c)$				
c	1	0	0	0	1
0	0	1	0	1	0
1	1	1	3	1	1
0	1	0	0	1	0
1	0	0	0	0	1

Multiple Line Detection



Image Space



Parameter Space

Multiple Line Detection

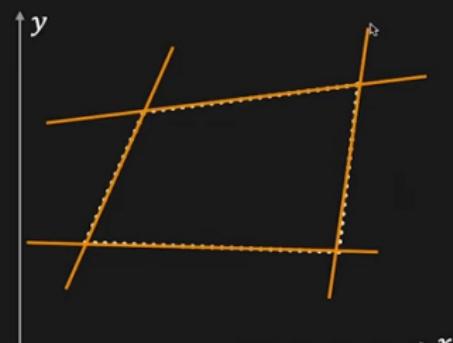
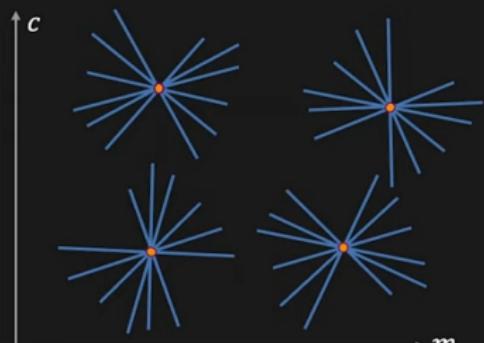


Image Space



Parameter Space

Better Parameterization

Issue: Slope of the line $-\infty \leq m \leq \infty$

- Large Accumulator
- More Memory and Computation

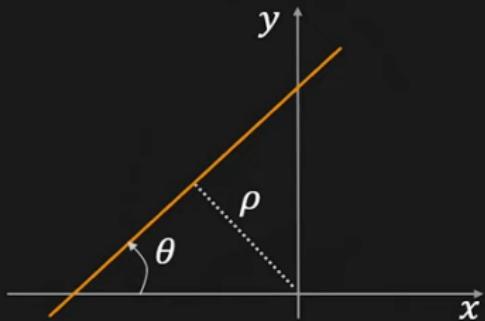
↳

Solution: Use $x \sin \theta - y \cos \theta + \rho = 0$

- Orientation θ is finite: $0 \leq \theta < \pi$
- Distance ρ is finite

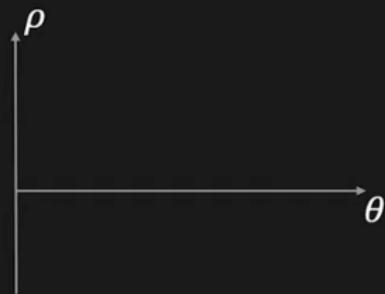
Better Parameterization

Image Space



$$\textcolor{brown}{x} \sin \theta - \textcolor{brown}{y} \cos \theta + \rho = 0$$

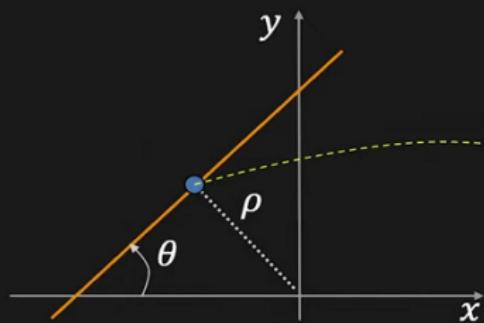
Parameter Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

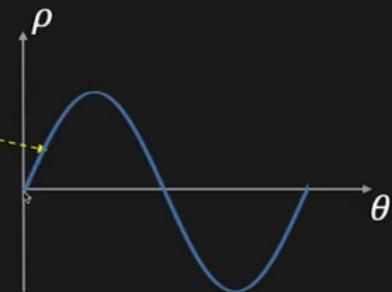
Better Parameterization

Image Space



$$\textcolor{brown}{x} \sin \theta - \textcolor{brown}{y} \cos \theta + \rho = 0$$

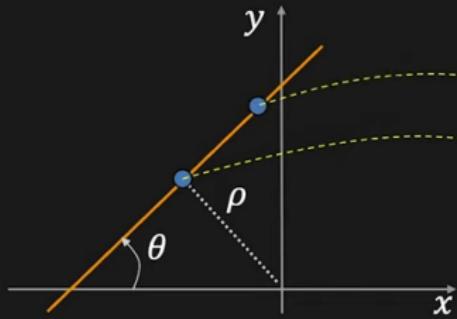
Parameter Space



$$x \sin \theta - y \cos \theta + \textcolor{brown}{\rho} = 0$$

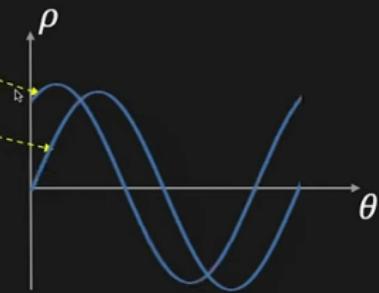
Better Parameterization

Image Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

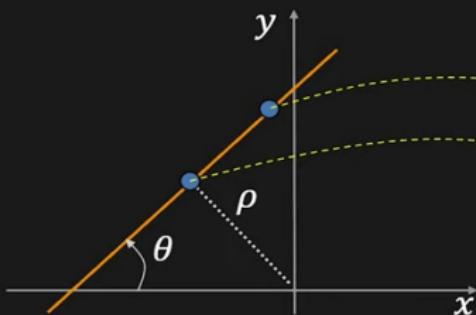
Parameter Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

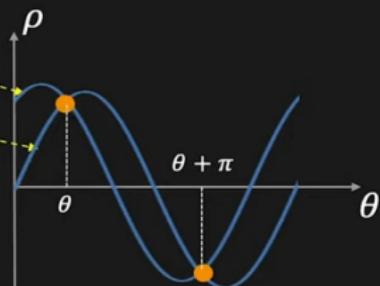
Better Parameterization

Image Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

Parameter Space



$$x \sin \theta - y \cos \theta + \rho = 0$$

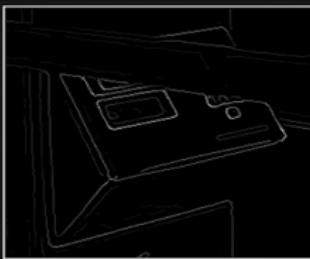
Hough Transform Mechanics

- How big should the accumulator cells be?
 - Too big, and different lines may be merged
 - Too small, and noise causes lines to be missed

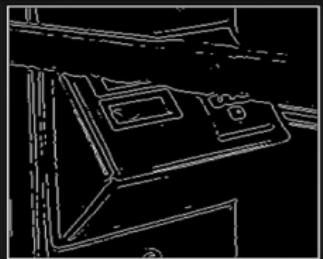
Line Detection Results



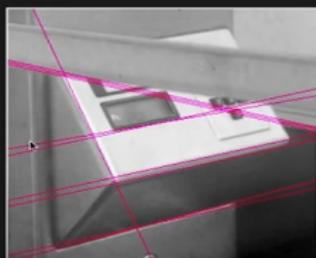
Original Image



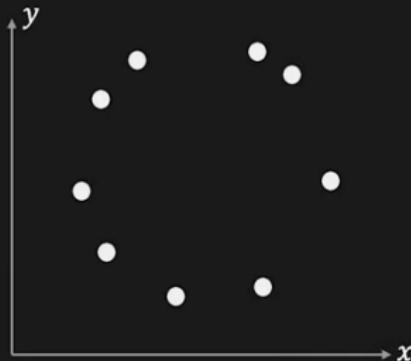
Gradient



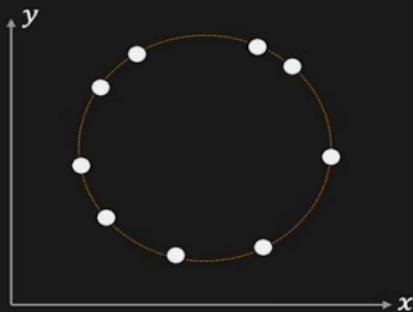
Edge (Threshold)



Hough Transform: Circle Detection

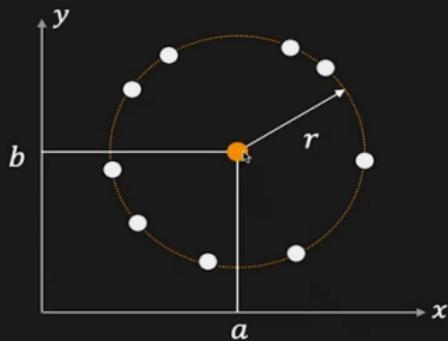


Hough Transform: Circle Detection



$$\text{Equation of Circle: } (x_i - a)^2 + (y_i - b)^2 = r^2$$

Hough Transform: Circle Detection

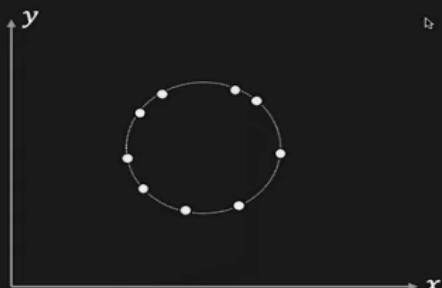


$$\text{Equation of Circle: } (x_i - a)^2 + (y_i - b)^2 = r^2$$

Hough Transform: Circle Detection

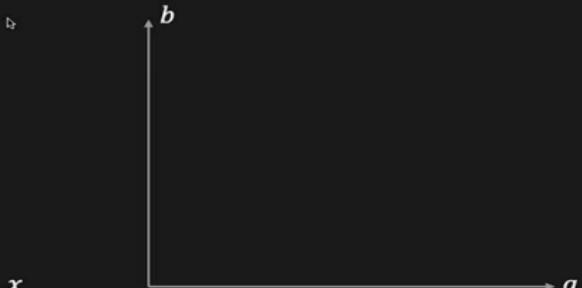
If radius r is known: Accumulator Array: $A(a, b)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space

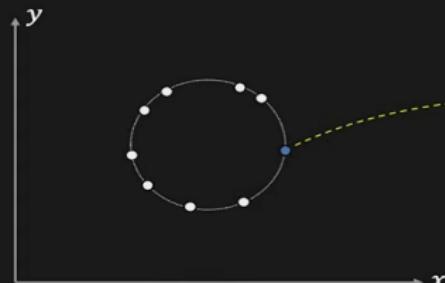


$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Hough Transform: Circle Detection

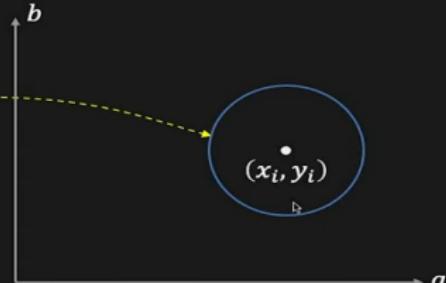
If radius r is known: Accumulator Array: $A(a, b)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space

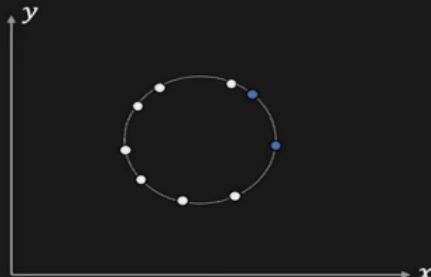


$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Hough Transform: Circle Detection

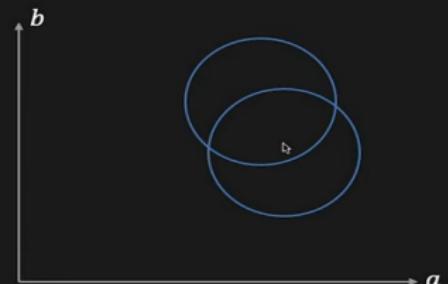
If radius r is known: Accumulator Array: $A(a, b)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space

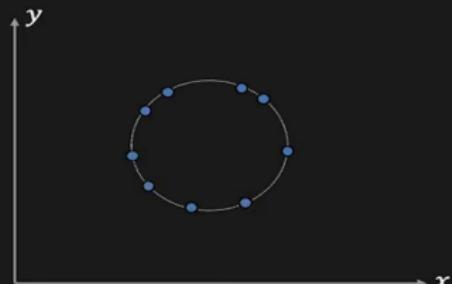


$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Hough Transform: Circle Detection

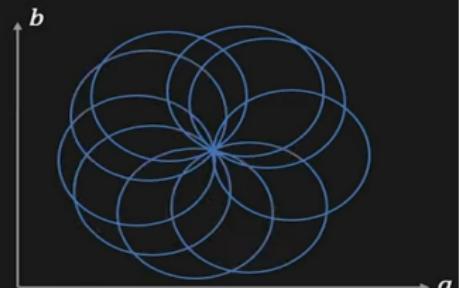
If radius r is known: Accumulator Array: $A(a, b)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Hough Transform: Circle Detection

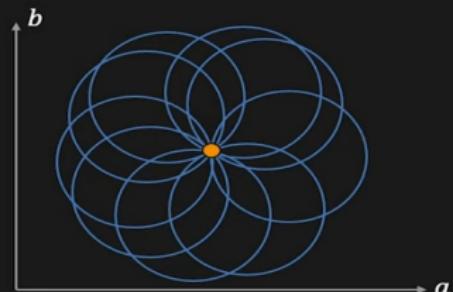
If radius r is known: Accumulator Array: $A(a, b)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

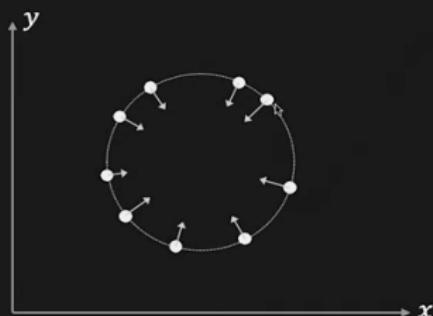
Circle Detection Results



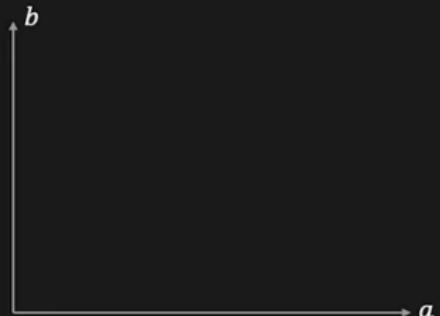
Using Gradient Information

Given: Edge Location (x_i, y_i) , Edge Direction φ_i and Radius r

Image Space



Parameter Space

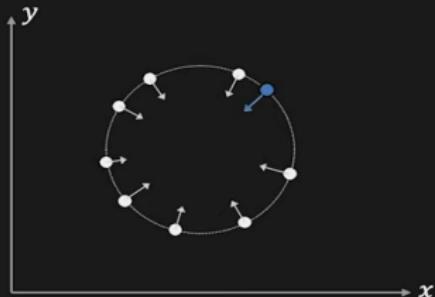


$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Using Gradient Information

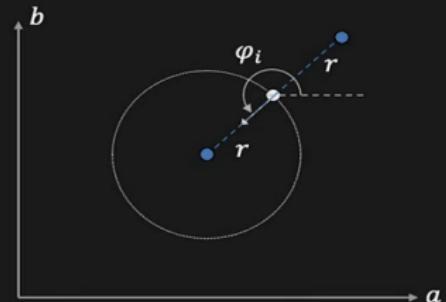
Given: Edge Location (x_i, y_i) , Edge Direction φ_i and Radius r

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



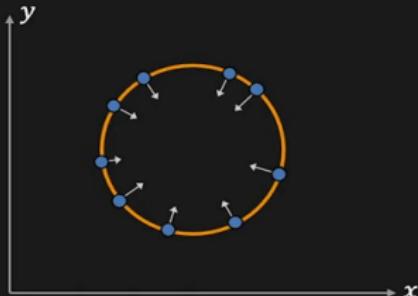
$$a = x_i \pm r \cos \varphi_i$$

$$b = y_i \pm r \sin \varphi_i$$

Using Gradient Information

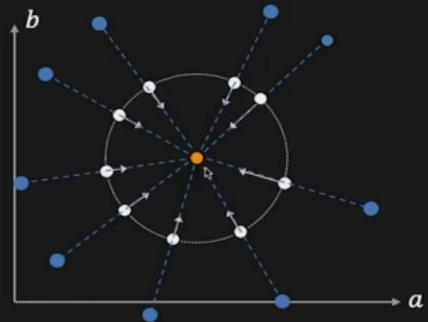
Given: Edge Location (x_i, y_i) , Edge Direction φ_i and Radius r

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



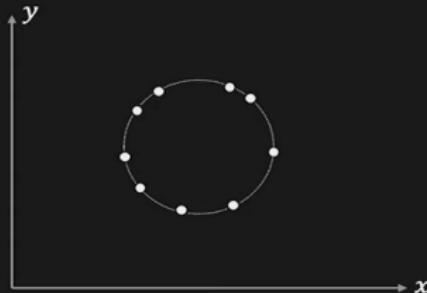
$$a = x_i \pm r \cos \varphi_i$$

$$b = y_i \pm r \sin \varphi_i$$

Hough Transform: Circle Detection

If radius r is NOT known: Accumulator Array: $A(a, b, r)$

Image Space

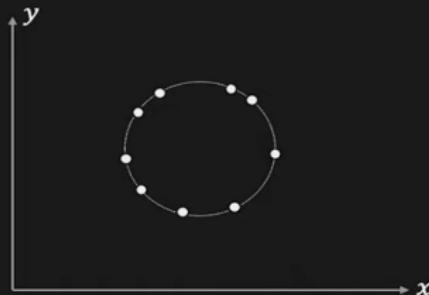


$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Hough Transform: Circle Detection

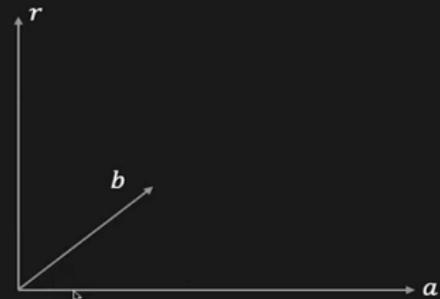
If radius r is NOT known: Accumulator Array: $A(a, b, r)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space

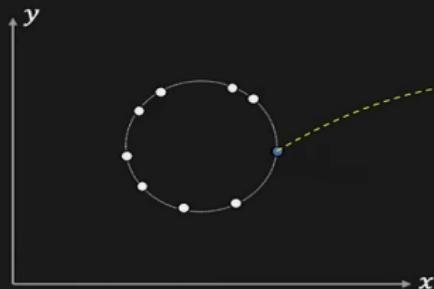


$$(a - x_i)^2 + (b - y_i)^2 = r^2$$

Hough Transform: Circle Detection

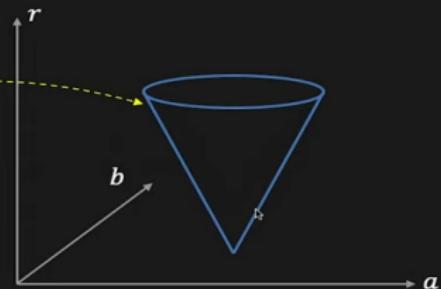
If radius r is NOT known: Accumulator Array: $A(a, b, r)$

Image Space



$$(x_i - a)^2 + (y_i - b)^2 = r^2$$

Parameter Space



$$(\alpha - x_i)^2 + (\beta - y_i)^2 = \gamma^2$$

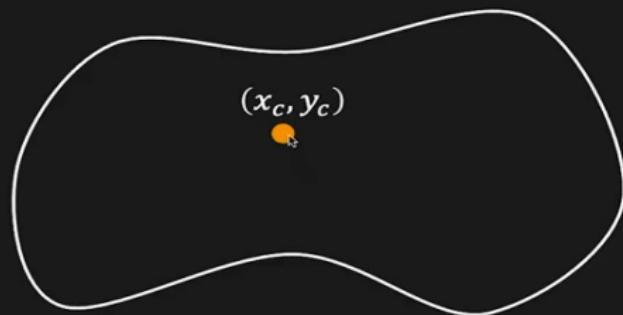
Generalized Hough Transform

Find shapes that cannot be described by equations



Generalized Hough Transform

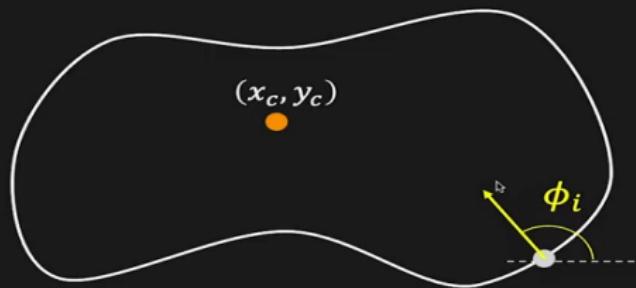
Find shapes that cannot be described by equations



Reference point: (x_c, y_c)

Generalized Hough Transform

Find shapes that cannot be described by equations

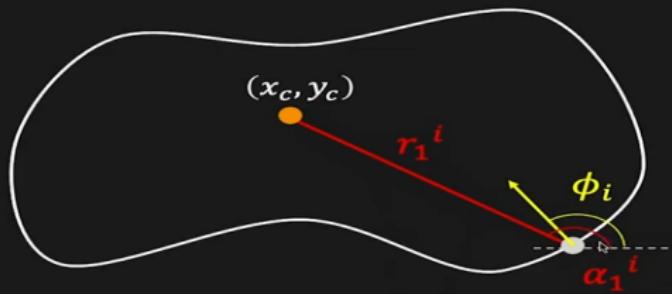


Reference point: (x_c, y_c)

Edge direction: ϕ_i $0 \leq \phi_i < 2\pi$

Generalized Hough Transform

Find shapes that cannot be described by equations



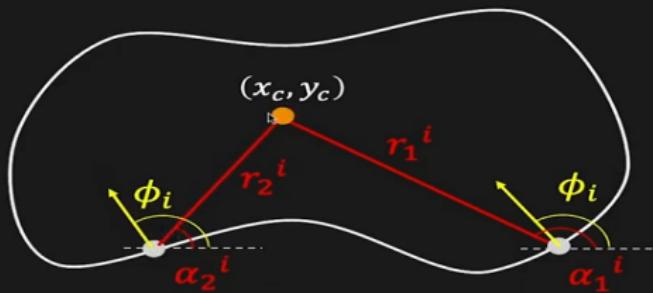
Reference point: (x_c, y_c)

Edge direction: ϕ_i $0 \leq \phi_i < 2\pi$

Edge location: $\vec{r}_k^i = (r_k^i, \alpha_k^i)$

Generalized Hough Transform

Find shapes that cannot be described by equations

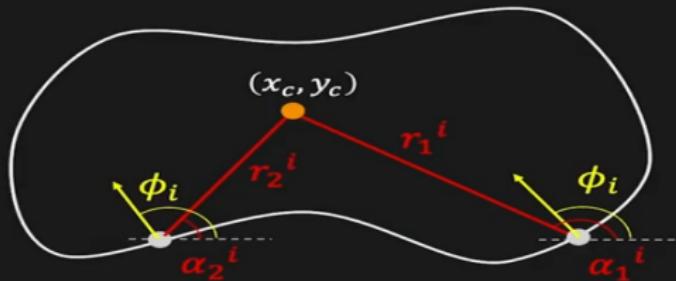


Reference point: (x_c, y_c)

Edge direction: $\phi_i \quad 0 \leq \phi_i < 2\pi$

Edge location: $\vec{r}_k^i = (r_k^i, \alpha_k^i)$

Hough Model

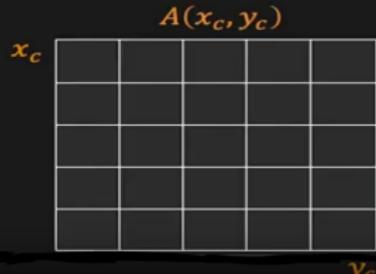
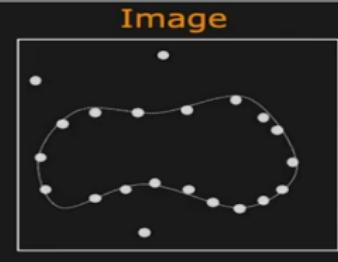


ϕ -Table:

Edge Direction	$\vec{r} = (r, \alpha)$
ϕ_1	$\vec{r}_1^1, \vec{r}_2^1, \vec{r}_3^1$
ϕ_2	\vec{r}_1^2, \vec{r}_2^2
\vdots	\vdots
ϕ_n	$\vec{r}_1^n, \vec{r}_2^n, \vec{r}_3^n, \vec{r}_4^n$

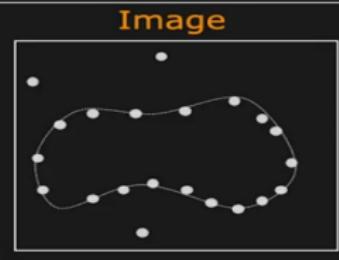
Generalized Hough Algorithm

- Create accumulator array $A(x_c, y_c)$



Generalized Hough Algorithm

- Create accumulator array $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)



$A(x_c, y_c)$				
x_c	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

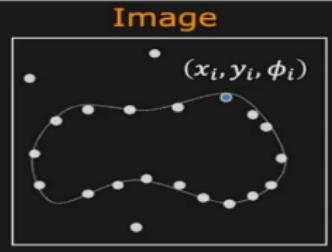
Generalized Hough Algorithm

- Create accumulator array $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)
- For each edge point (x_i, y_i, ϕ_i) ,

For each entry $\phi_i \rightarrow \vec{r}_k^i$ in ϕ - table,

$$x_c = x_i \pm r_k^i \cos(\alpha_k^i)$$

$$y_c = y_i \pm r_k^i \sin(\alpha_k^i)$$



$A(x_c, y_c)$				
x_c	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

y_c

Generalized Hough Algorithm

- Create **accumulator array** $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)
- For each edge point (x_i, y_i, ϕ_i) ,

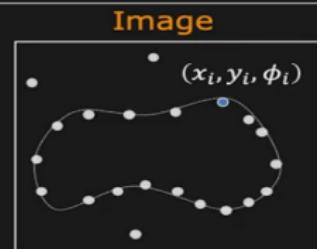
For each entry $\phi_i \rightarrow \vec{r}_k^i$ in ϕ - table,

$$x_c = x_i \pm r_k^i \cos(\alpha_k^i)$$

$$y_c = y_i \pm r_k^i \sin(\alpha_k^i)$$

$$A(x_c, y_c) = A(x_c, y_c) + 1$$

- Find local maxima in $A(x_c, y_c)$



x_c	$A(x_c, y_c)$				
	0	0	0	0	0
0	0	2	0	1	0
0	0	0	4	1	0
0	2	0	0	0	0
0	0	0	1	0	0

y_c

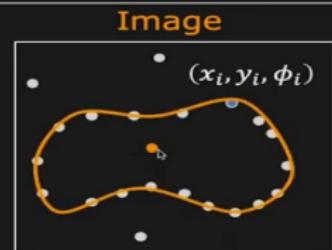
Generalized Hough Algorithm

- Create **accumulator array** $A(x_c, y_c)$
- Set $A(x_c, y_c) = 0$ for all (x_c, y_c)
- For each edge point (x_i, y_i, ϕ_i) ,
For each entry $\phi_i \rightarrow r_k^i$ in ϕ - table,

$$x_c = x_i \pm r_k^i \cos(\alpha_k^i)$$

$$y_c = y_i \pm r_k^i \sin(\alpha_k^i)$$

$$A(x_c, y_c) = A(x_c, y_c) + 1$$
- Find local maxima in $A(x_c, y_c)$



x_c	$A(x_c, y_c)$				
0	0	0	0	0	0
0	2	0	1	0	0
0	0	4	1	0	0
0	2	0	0	0	0
0	0	0	1	0	0

y_c

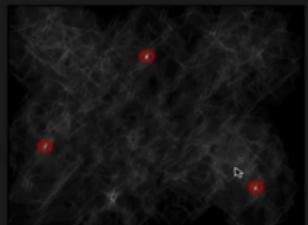
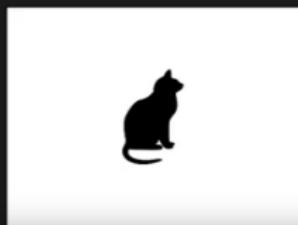
Results



Model



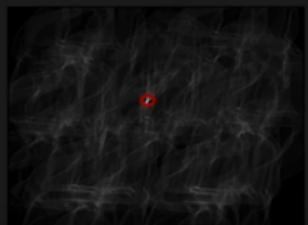
Image

Hough Transform $A(x_c, y_c)$ 

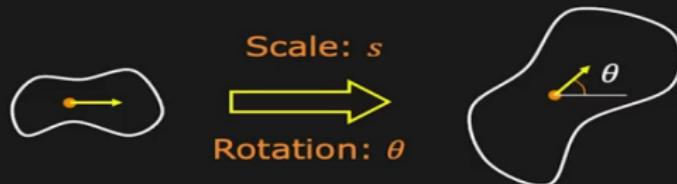
Model



Image

Hough Transform $A(x_c, y_c)$

Handling Scale And Rotation



Use Accumulation Array: $A(x_c, y_c, s, \theta)$

$$x_c = x_i \pm r_k^i \cdot s \cos(\alpha_k^i + \theta)$$

$$y_c = y_i \pm r_k^i \cdot s \sin(\alpha_k^i + \theta)$$

$$A(x_c, y_c, s, \theta) = A(x_c, y_c, s, \theta) + 1$$

Huge Memory and Computationally Expensive!

Hough Transform: Comments

- Works on disconnected edges
- Relatively insensitive to occlusion and noise
- Effective for simple shapes (lines, circles, etc.)
- Complex Shapes: Generalized Hough Transform
- Trade-off between work in image space and parameter space

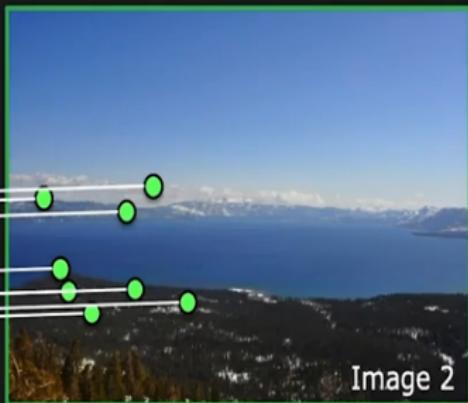
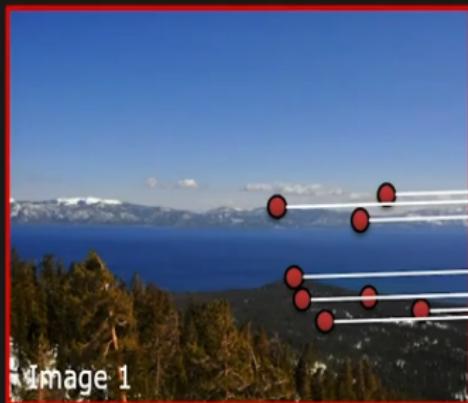
Computer Vision-IT416

Dinesh Naik

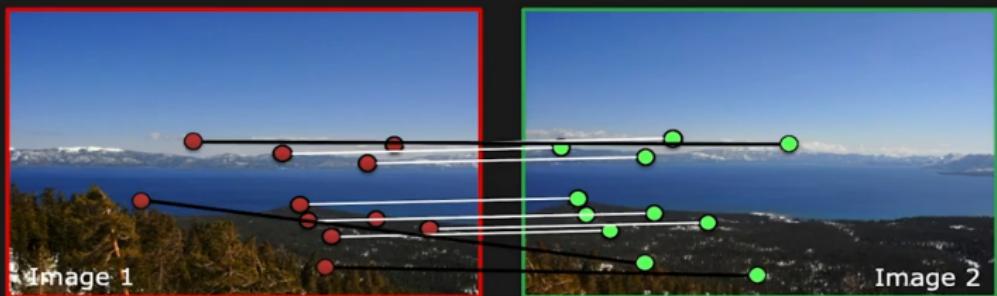
Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

What Could Go Wrong?



What Could Go Wrong?



Outliers!

We need to robustly compute transformation in the presence of wrong matches.

RANDom SAmple Consensus

General RANSAC Algorithm:

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.

For homography:

$s = 4$ points.

RANDom SAmple Consensus

General RANSAC Algorithm:

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples.

For homography:

$s = 4$ points.

RANDom SAmple Consensus

General RANSAC Algorithm:

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples.
3. Count the number M of data points (inliers) that fit the model within a measure of error ε .

For homography:

$s = 4$ points. ε is acceptable alignment error in pixels.

RANdom SAmple Consensus

General RANSAC Algorithm:

1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples.
3. Count the number M of data points (inliers) that fit the model within a measure of error ε .
4. Repeat Steps 1-3 N times

For homography:

$s = 4$ points. ε is acceptable alignment error in pixels.

RANdom SAmple Consensus

General RANSAC Algorithm:

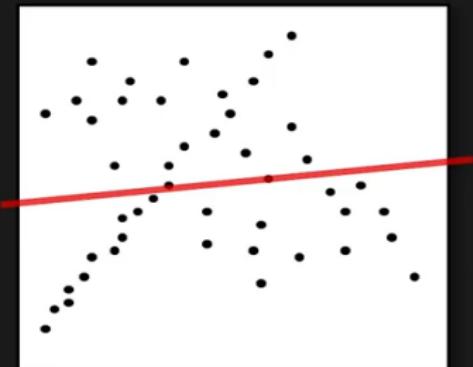
1. Randomly choose s samples. Typically s is the minimum samples to fit a model.
2. Fit the model to the randomly chosen samples.
3. Count the number M of data points (inliers) that fit the model within a measure of error ε .
4. Repeat Steps 1-3 N times
5. Choose the model that has the largest number M of inliers.

For homography:

$s = 4$ points. ε is acceptable alignment error in pixels.

RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting

RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting

Inliers: 2



RANSAC Iteration 1

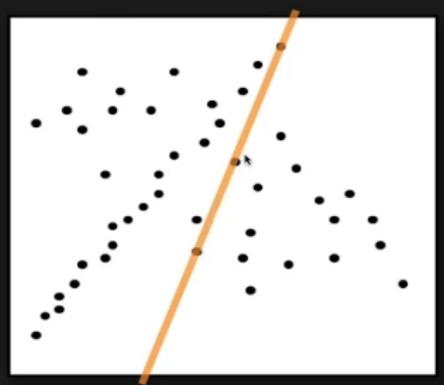
Inliers: 4

RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting



RANSAC Iteration 2

Inliers: 3

RANSAC Example: Line Fitting

Robust line fitting:



Least Squares Fitting



RANSAC Iteration i

Inliers: 20

Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

What is an Active Contour?

Given: Approximate boundary (contour) around the object

Task: Evolve (move) the contour to fit exact object boundary



Image

What is an Active Contour?

Given: Approximate boundary (contour) around the object

Task: Evolve (move) the contour to fit exact object boundary

Active Contour:

Iteratively “deform” the initial contour so that:

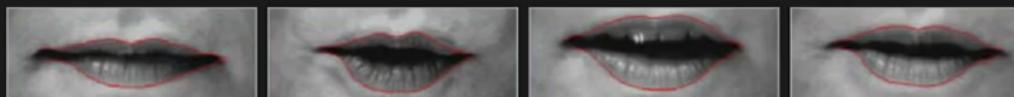
- It is near pixels with high gradient (edges)
- It is smooth



Image

Power of Deformable Contours

Boundaries could deform over time



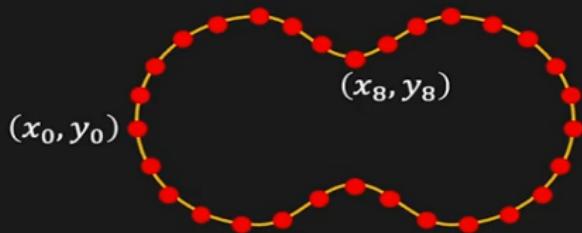
Boundaries could deform with viewpoint



Boundary Tracking: Use the boundary from the current image as initial boundary for the next image.

Representing a Contour

Contour \mathbf{v} : An ordered list of 2D vertices (control points) connected by straight lines of fixed length



$$\mathbf{v} = \{v_i = (x_i, y_i) \mid i = 0, 1, 2, \dots, n - 1\}$$

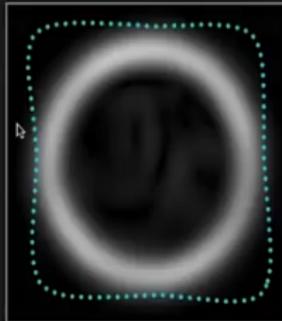
Attracting Contours to Edges



Image with
Initial Contour



Gradient Magnitude
Squared
 $\|\nabla I\|^2$



Blurred Gradient
Magnitude Squared
 $\|\nabla n_\sigma * I\|^2$

Attracting Contours to Edges

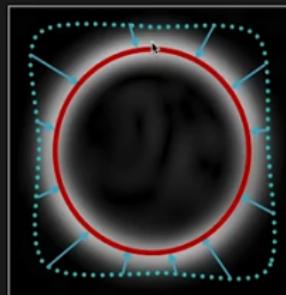


Image with
Initial Contour



Gradient Magnitude
Squared

$$\|\nabla I\|^2$$



Blurred Gradient
Magnitude Squared

$$\|\nabla n_\sigma * I\|^2$$

Maximize Sum of Gradient Magnitude Square

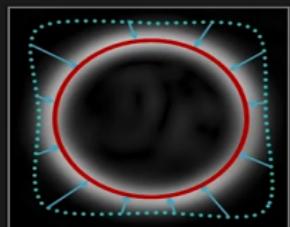
Attracting Contours to Edges



Image with
Initial Contour



Gradient Magnitude
Squared
 $\|\nabla I\|^2$



Blurred Gradient
Magnitude Squared
 $\|\nabla n_\sigma * I\|^2$

Maximize Sum of Gradient Magnitude Square

≡ Minimize -ve (Sum of Gradient Magnitude Square)

≡ Minimize $E_{image} = - \sum_{i=0}^{n-1} \|\nabla n_\sigma * I(v_i)\|^2$

Contour Deformation: Greedy Algorithm

1. For each contour point v_i ($i = 0, \dots, n - 1$), move v_i to a position within a window W where the energy function E_{image} for the contour is minimum.



Contour Deformation: Greedy Algorithm

1. For each contour point v_i ($i = 0, \dots, n - 1$), move v_i to a position within a window W where the energy function E_{image} for the contour is minimum.



Contour Deformation: Greedy Algorithm

1. For each contour point v_i ($i = 0, \dots, n - 1$), move v_i to a position within a window W where the energy function E_{image} for the contour is minimum.

2. If the sum of motions of all the contour points is less than a threshold, stop. Else go to Step 1.



Contour Deformation: Greedy Algorithm

1. For each contour point v_i ($i = 0, \dots, n - 1$), move v_i to a position within a window W where the energy function E_{image} for the contour is minimum.

2. If the sum of motions of all the contour points is less than a threshold, stop. Else go to Step 1.



Greedy solution might be suboptimal and slow.

Sensitivity to Noise and Initialization



Contour fitted to
gradient magnitude

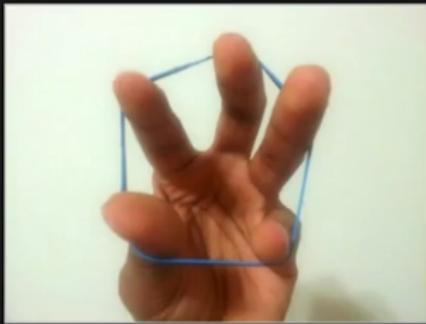
Sensitivity to Noise and Initialization



Contour fitted to
gradient magnitude

Solution: Add constraints that make the contour
contract and remain smooth

Making Contours Elastic and Smooth

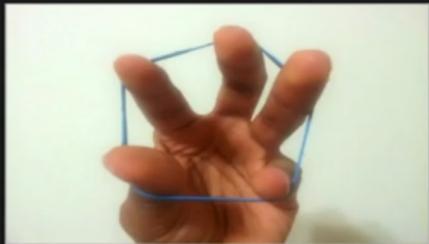


Elastic and contracts
like a rubber band



Smooth
like a metal strip

Making Contours Elastic and Smooth



Elastic and contracts
like a rubber band



Smooth
like a metal strip

Minimize Internal Bending Energy of the Contour:

$$E_{contour} = \alpha E_{elastic} + \beta E_{smooth}$$

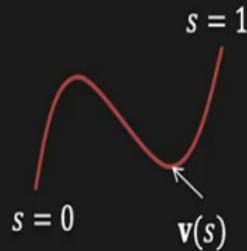
(α, β) : Control the influence of elasticity and smoothness

Elasticity and Smoothness

For point $0 \leq s \leq 1$ on continuous contour $\mathbf{v}(s) = (x(s), y(s))$:

$$E_{elastic} = \left\| \frac{d\mathbf{v}}{ds} \right\|^2$$

$$E_{smooth} = \left\| \frac{d^2\mathbf{v}}{ds^2} \right\|^2$$

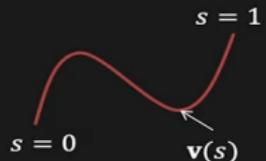


Elasticity and Smoothness

For point $0 \leq s \leq 1$ on continuous contour $\mathbf{v}(s) = (x(s), y(s))$:

$$E_{elastic} = \left\| \frac{d\mathbf{v}}{ds} \right\|^2$$

$$E_{smooth} = \left\| \frac{d^2\mathbf{v}}{ds^2} \right\|^2$$



Discrete approximations at control point \mathbf{v}_i :

$$E_{elastic}(\mathbf{v}_i) = \left\| \frac{d\mathbf{v}}{ds} \right\|^2 \approx \|\mathbf{v}_{i+1} - \mathbf{v}_i\|^2 = (x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2$$

$$E_{smooth}(\mathbf{v}_i) = \left\| \frac{d^2\mathbf{v}}{ds^2} \right\|^2 \approx \|(\mathbf{v}_{i+1} - \mathbf{v}_i) - (\mathbf{v}_i - \mathbf{v}_{i-1})\|^2$$

$$= (x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2$$

Elasticity and Smoothness

Internal bending energy along the entire contour:

$$E_{contour} = \alpha E_{elastic} + \beta E_{smooth}$$

where:

$$E_{elastic} = \sum_{i=0}^{n-1} [(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2]$$

$$E_{smooth} = \sum_{i=0}^{n-1} [(x_{i+1} - 2x_i + x_{i-1})^2 + (y_{i+1} - 2y_i + y_{i-1})^2]$$

Combining the Forces

Image Energy, E_{image} : Measure of how well the contour latches on to edges

Internal Energy, $E_{contour}$: Measure of elasticity and smoothness

Total Energy of Active Contour:

$$E_{total} = E_{image} + E_{contour}$$

Minimize the Total Energy

Contour Deformation: Greedy Algorithm

1. Uniformly sample the contour to get n contour points.
2. For each contour point v_i ($i = 0, \dots, n - 1$), move v_i to a position within a window W where the energy function E_{total} for the entire contour is minimum.

$$E_{total} = E_{image} + E_{contour}$$

3. If the sum of motions of all the contour points is less than a threshold, stop. Else go to Step 1.



Result: Effect of Contour Constraint



Without contour constraint

$$E_{total} = E_{image}$$



With contour constraint

$$E_{total} = E_{image} + E_{contour}$$

Result: Boundary Around Two Objects



Large α

(More like a rubber band)



Small α

(Less like a rubber band)

Computer Vision-IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

Image Stitching



Image 1



Image 2



Image 3

How would you align these images?

Image Stitching

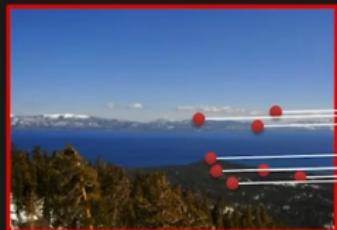


Image 1



Image 2

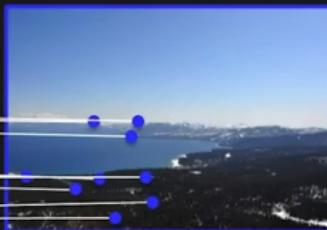


Image 3

Find corresponding points
(using feature detectors like SIFT)

Image Stitching



Image 1

Image 2

Image 3

Find geometric relationship between the images

Image Stitching



Image 1

Image 2

Image 3

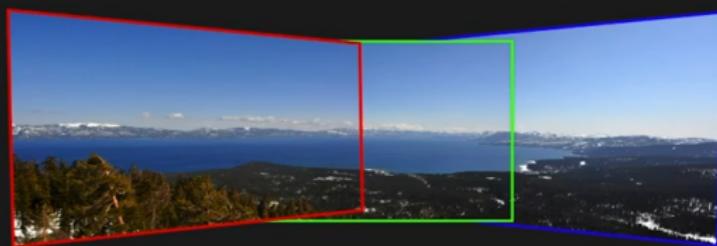
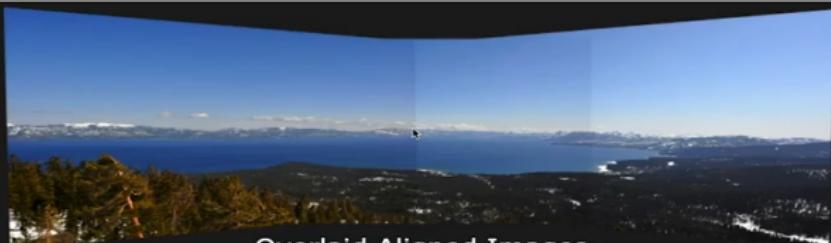


Image Stitching



Overlaid Aligned Images



Blended Images

Image Stitching

Combine multiple photos to create a larger photo

Topics:

- (1) 2x2 Image Transformations
- (2) 3x3 Image Transformations
- (3) Computing Homography
- (4) Dealing with Outliers: RANSAC
- (5) Warping and Blending Images

Image Manipulation

Image Filtering: Change range (brightness)

$$g(x, y) = T_r(f(x, y))$$

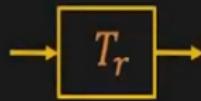


Image Manipulation

Image Filtering: Change range (brightness)

$$g(x, y) = T_r(f(x, y))$$

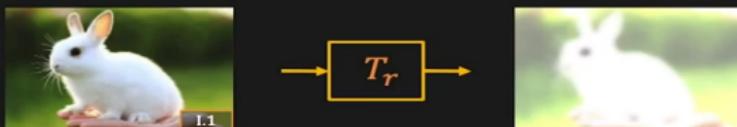
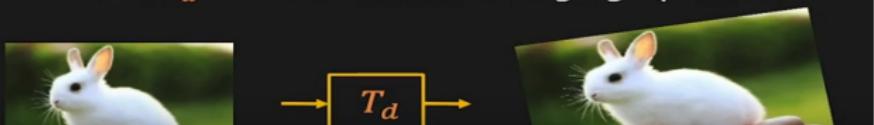


Image Warping: Change domain (location)

$$g(x, y) = f(T_d(x, y))$$

Transformation T_d is a coordinate changing operator



Global Warping/Transformation



Translation



Rotation



Scaling and Aspect

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



Affine



Projective



Barrel

Transformation T is the same over entire domain

Often can be described by just a few parameters

2x2 Linear Transformations



$$\mathbf{p}_1 = (x_1, y_1)$$



$$\mathbf{p}_2 = (x_2, y_2)$$

T can be represented by a matrix.

$$\mathbf{p}_2 = T\mathbf{p}_1 \qquad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = T \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \qquad \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Scaling (Stretching or Squishing)

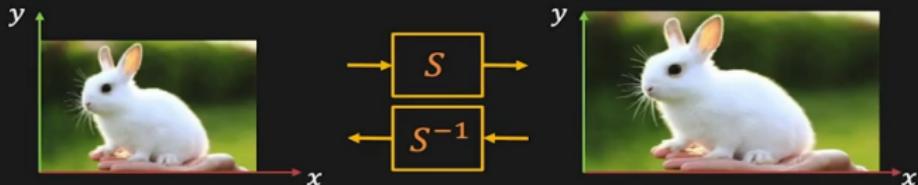


Forward:

$$x_2 = ax_1 \quad y_2 = by_1$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = S \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Scaling (Stretching or Squishing)



Forward:

$$x_2 = ax_1 \quad y_2 = by_1$$

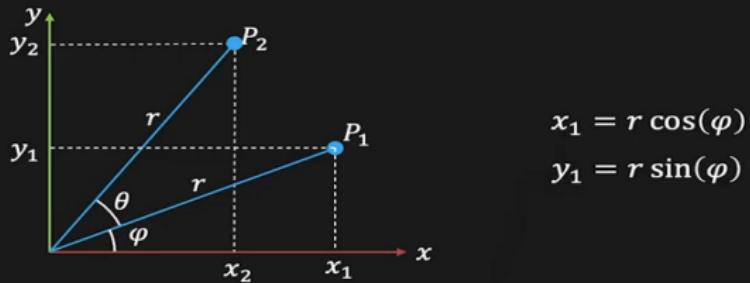
Inverse:

$$x_1 = \frac{1}{a}x_2 \quad y_1 = \frac{1}{b}y_2$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = S \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = S^{-1} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1/a & 0 \\ 0 & 1/b \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

2D Rotation



$$x_2 = r \cos(\varphi + \theta)$$

$$x_2 = r \cos \varphi \cos \theta - r \sin \varphi \sin \theta$$

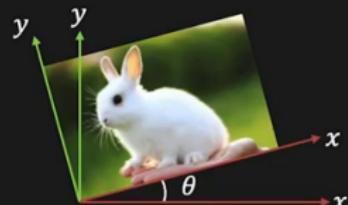
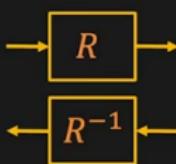
$$x_2 = x_1 \cos \theta - y_1 \sin \theta$$

$$y_2 = r \sin(\varphi + \theta)$$

$$y_2 = r \cos \varphi \sin \theta + r \sin \varphi \cos \theta$$

$$y_2 = x_1 \sin \theta + y_1 \cos \theta$$

Rotation



Forward:

$$x_2 = x_1 \cos\theta - y_1 \sin\theta$$

$$y_2 = x_1 \sin\theta + y_1 \cos\theta$$

Inverse:

$$x_1 = x_2 \cos\theta + y_2 \sin\theta$$

$$y_1 = -x_2 \sin\theta + y_2 \cos\theta$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = R \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} \quad \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = R^{-1} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

Skew



Horizontal Skew:

$$x_2 = x_1 + m_x y_1$$

$$y_2 = y_1$$

Vertical Skew:

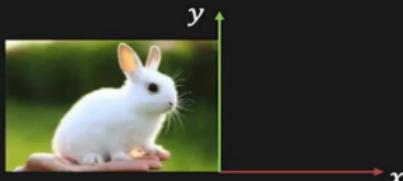
$$x_2 = x_1$$

$$y_2 = m_y x_1 + y_1$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = S_x \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & m_x \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = S_x \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ m_y & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Mirror



Mirror about Y-axis:

$$x_2 = -x_1$$

$$y_2 = y_1$$

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

Mirror about line $y = x$:

$$x_2 = y_1$$

$$y_2 = x_1$$

$$M_{xy} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$



2x2 Matrix Transformations

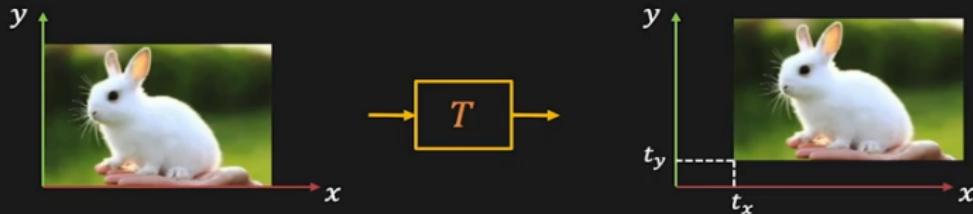
Any transformation of the form:

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

- Origin maps to the origin
- Lines map to lines
- Parallel lines remain parallel
- Closed under composition

$$\left. \begin{array}{l} \mathbf{p}_2 = T_{21}\mathbf{p}_1 \\ \mathbf{p}_3 = T_{32}\mathbf{p}_2 \\ \mathbf{p}_3 = T_{31}\mathbf{p}_1 \end{array} \right\} \quad \mathbf{p}_3 = T_{32}\mathbf{p}_2 = T_{32}T_{21}\mathbf{p}_1 \quad \Rightarrow \quad T_{31} = T_{32}T_{21}$$

Translation



$$x_2 = x_1 + t_x \qquad y_2 = y_1 + t_y$$

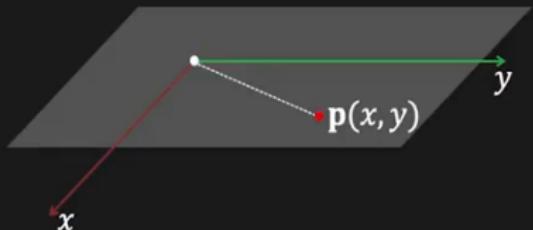
↳

Can translation be expressed as a 2x2 matrix? No.

Homogenous Coordinates

The homogenous representation of a 2D point $\mathbf{p} = (x, y)$ is a 3D point $\tilde{\mathbf{p}} = (\tilde{x}, \tilde{y}, \tilde{z})$. The third coordinate $\tilde{z} \neq 0$ is fictitious such that:

$$x = \frac{\tilde{x}}{\tilde{z}} \quad y = \frac{\tilde{y}}{\tilde{z}}$$

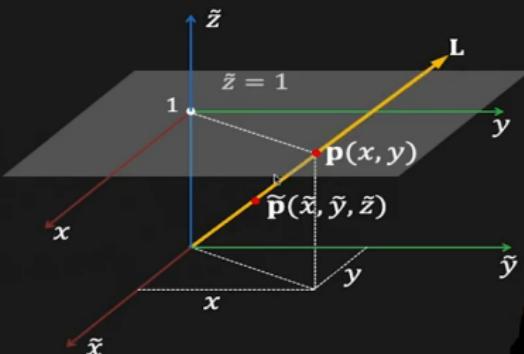


Homogenous Coordinates

The **homogenous representation** of a 2D point $\mathbf{p} = (x, y)$ is a 3D point $\tilde{\mathbf{p}} = (\tilde{x}, \tilde{y}, \tilde{z})$. The third coordinate $\tilde{z} \neq 0$ is fictitious such that:

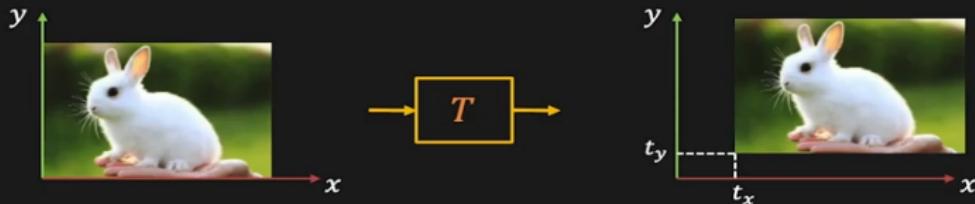
$$x = \frac{\tilde{x}}{\tilde{z}} \quad y = \frac{\tilde{y}}{\tilde{z}}$$

$$\mathbf{p} \equiv \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{z}x \\ \tilde{z}y \\ \tilde{z} \end{bmatrix} \equiv \begin{bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{bmatrix} = \tilde{\mathbf{p}}$$



Every point on line L (except origin) represents the homogenous coordinate of $\mathbf{p}(x, y)$

Translation



$$x_2 = x_1 + t_x \quad y_2 = y_1 + t_y$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Scaling, Rotation, Skew, Translation

$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Scaling

$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} 1 & m_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Skew

$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Translation

$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

Rotation

Affine Transformation

Any transformation of the form:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$



Affine Transformation

Any transformation of the form:

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

- Origin does not necessarily map to the origin
- Lines map to lines
- Parallel lines remain parallel
- Closed under composition

Projective Transformation

Any transformation of the form:

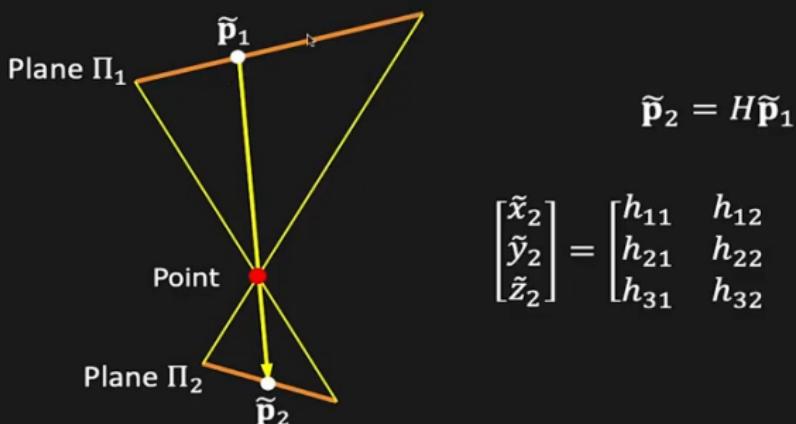
$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ \textcolor{orange}{h_{31}} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} \quad \tilde{\mathbf{p}}_2 = H \tilde{\mathbf{p}}_1$$



Also called Homography

Projective Transformation

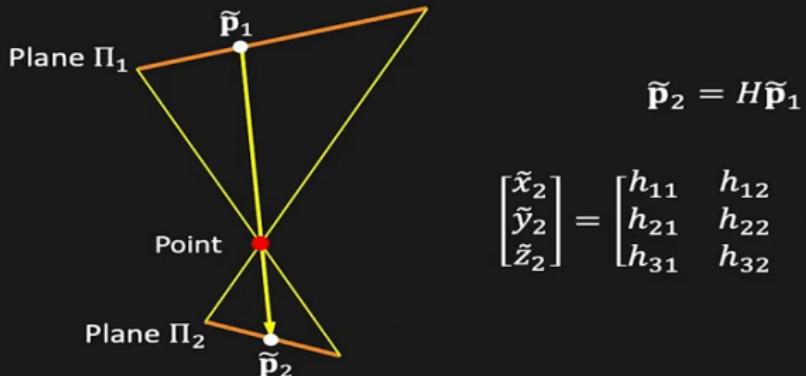
Mapping of one plane to another through a point



$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

Projective Transformation

Mapping of one plane to another through a point



$$\begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

Same as imaging a plane through a pinhole

Projective Transformation

Homography can only be defined up to a scale.

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv k \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

Projective Transformation

Homography can only be defined up to a scale.

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_2 \\ \tilde{y}_2 \\ \tilde{z}_2 \end{bmatrix} \equiv k \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{y}_1 \\ \tilde{z}_1 \end{bmatrix}$$

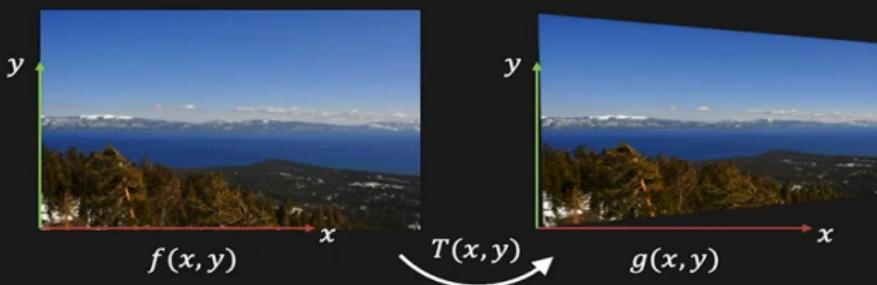
If we fix scale such that $\sqrt{\sum(h_{ij})^2} = 1$ then 8 free parameters

- Origin does not necessarily map to the origin
- Lines map to lines
- Parallel lines do not necessarily remain parallel
- Closed under composition

Warping Images

Given a transformation T and a image $f(x, y)$, compute the transformed image $g(x, y)$

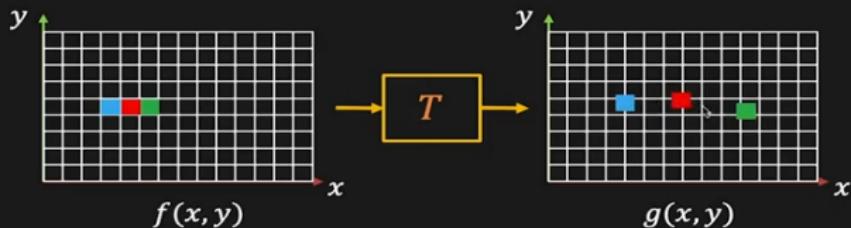
$$g(x, y) = f(T(x, y))$$



Forward Warping

Send each pixel (x, y) in $f(x, y)$ to its corresponding location $T(x, y)$ in $g(x, y)$

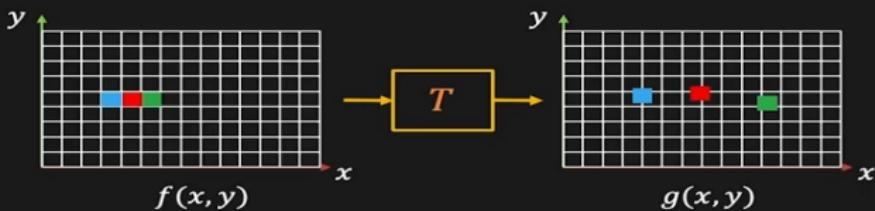
$$g(x, y) = f(T(x, y))$$



Forward Warping

Send each pixel (x, y) in $f(x, y)$ to its corresponding location $T(x, y)$ in $g(x, y)$

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

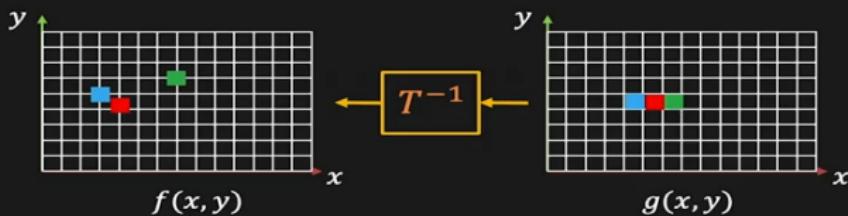


What if pixel lands in between pixels?
What if not all pixels in $g(x, y)$ are filled?

Backward Warping

Get each pixel (x, y) in $g(x, y)$ from its corresponding location $T^{-1}(x, y)$ in $f(x, y)$

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



What if pixel lands between pixels?
Use Nearest Neighbor or Interpolate

Image Alignment Process



Image 1

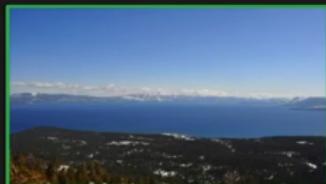


Image 2



Image 3



Reference Image
(Image 2)

Image Alignment Process

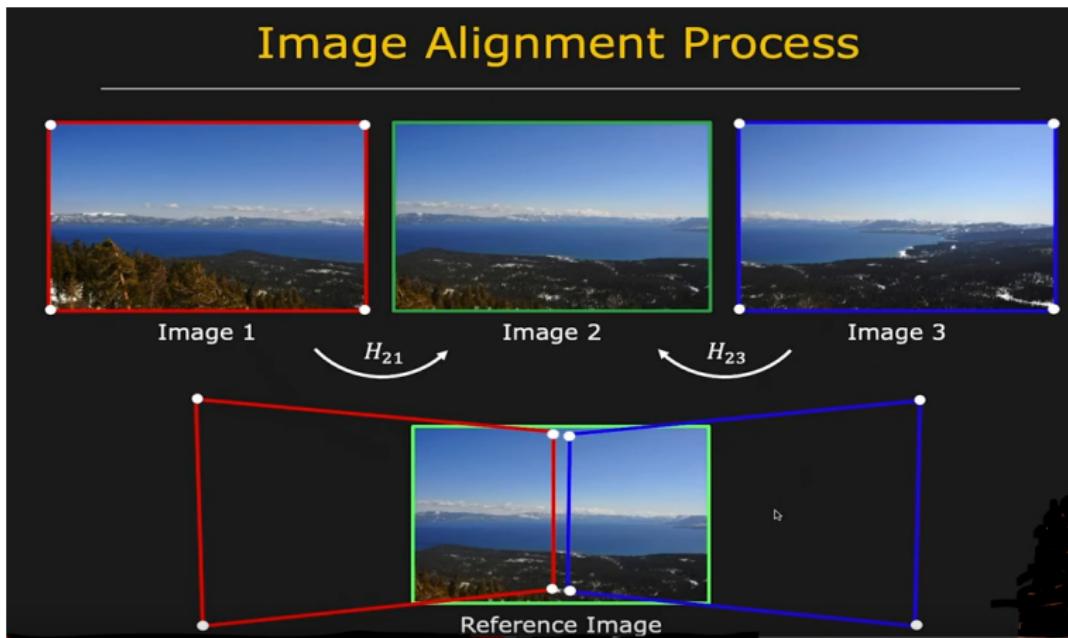


Image Alignment Process

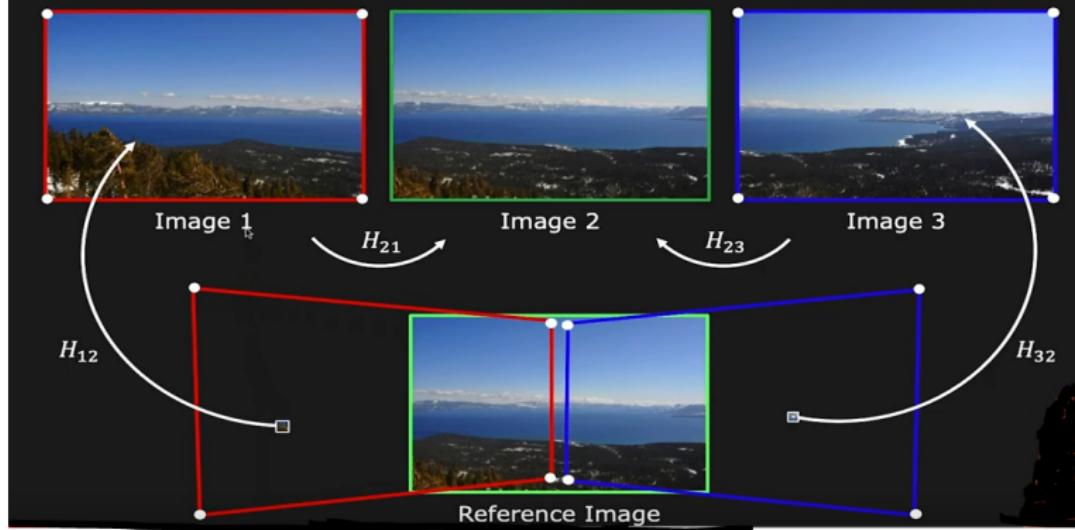
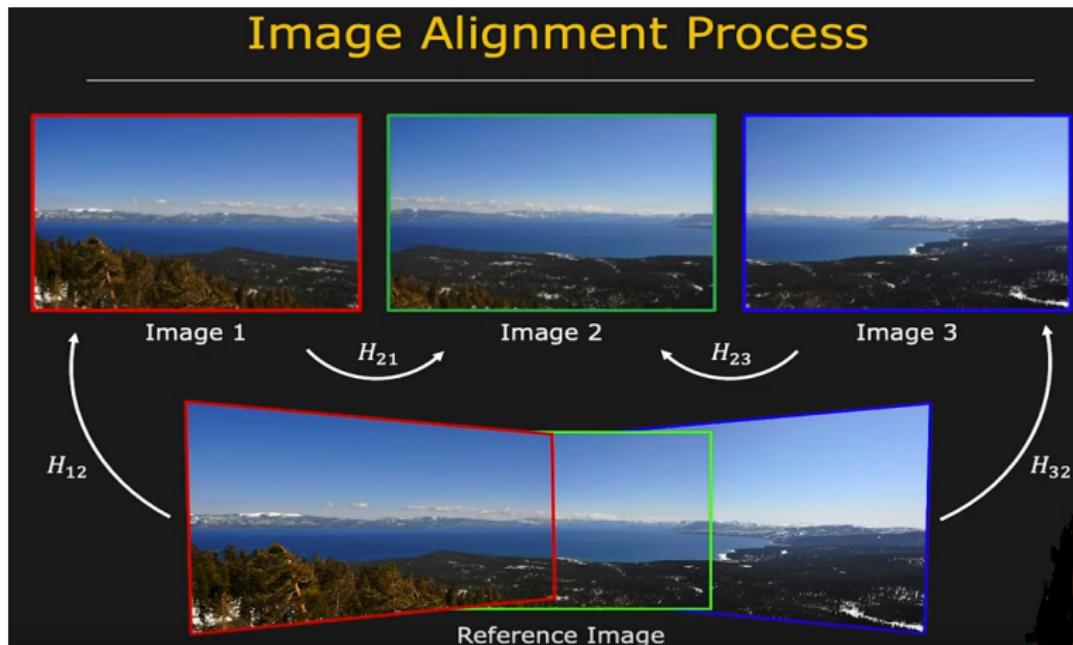
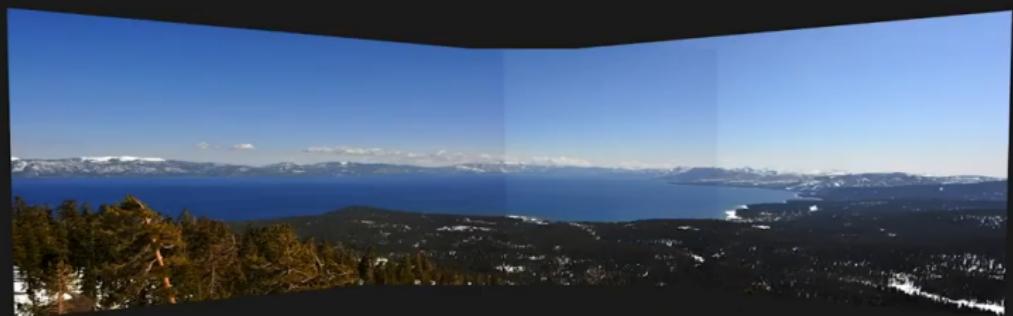


Image Alignment Process



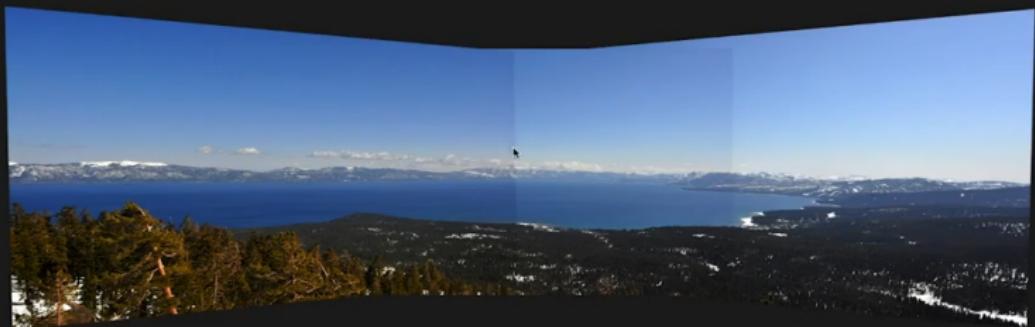
Blending Images



Overlaid Aligned Images

Hard seams due to vignetting, exposure differences, etc.

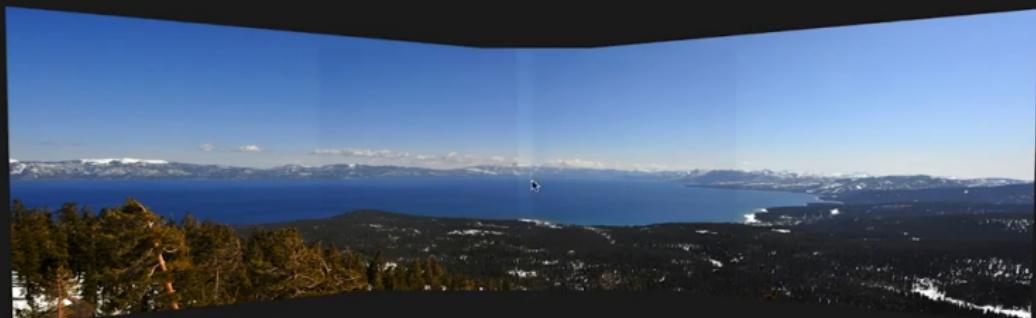
Blending Images



Overlaid Aligned Images

Hard seams due to vignetting, exposure differences, etc.

Blending Images: Averaging



Averaged Images

Seams still visible

Blending Images

Say we want to blend images I_1 and I_2 at the center



Image I_1



Image I_2

Blending Images

Say we want to blend images I_1 and I_2 at the center



Blending Images

Say we want to blend images I_1 and I_2 at the center



Image I_1

+



Image I_2



Weight w_1



Weight w_2

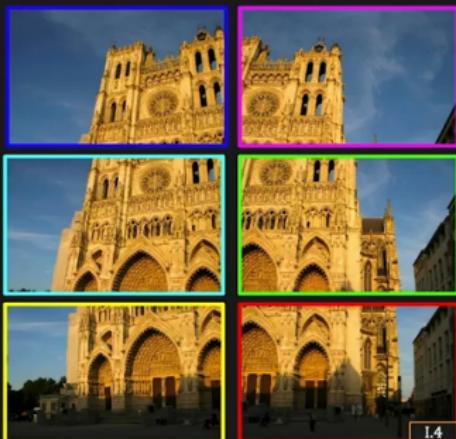
$$I_{blend} = \frac{w_1 I_1 + w_2 I_2}{w_1 + w_2}$$

Blending Images

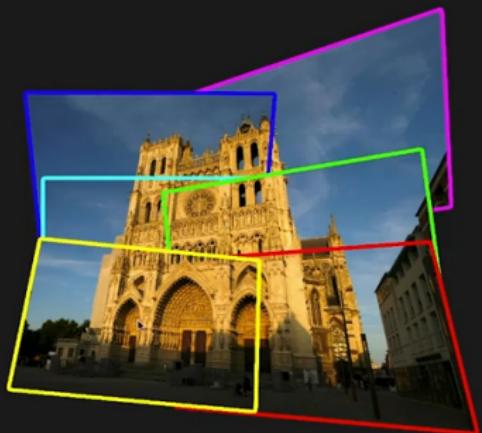
Say we want to blend images I_1 and I_2 at the center



Image Stitching Example



Source Images



Aligned Images

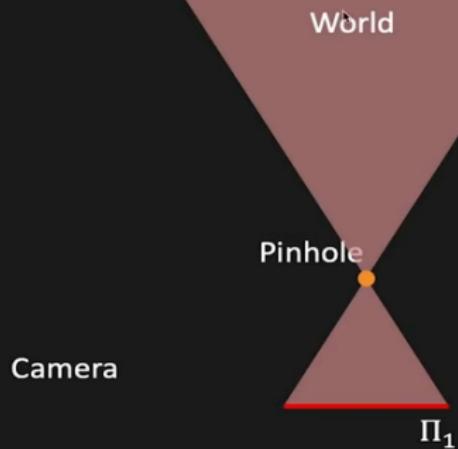
Computer Vision-IT416

Dinesh Naik

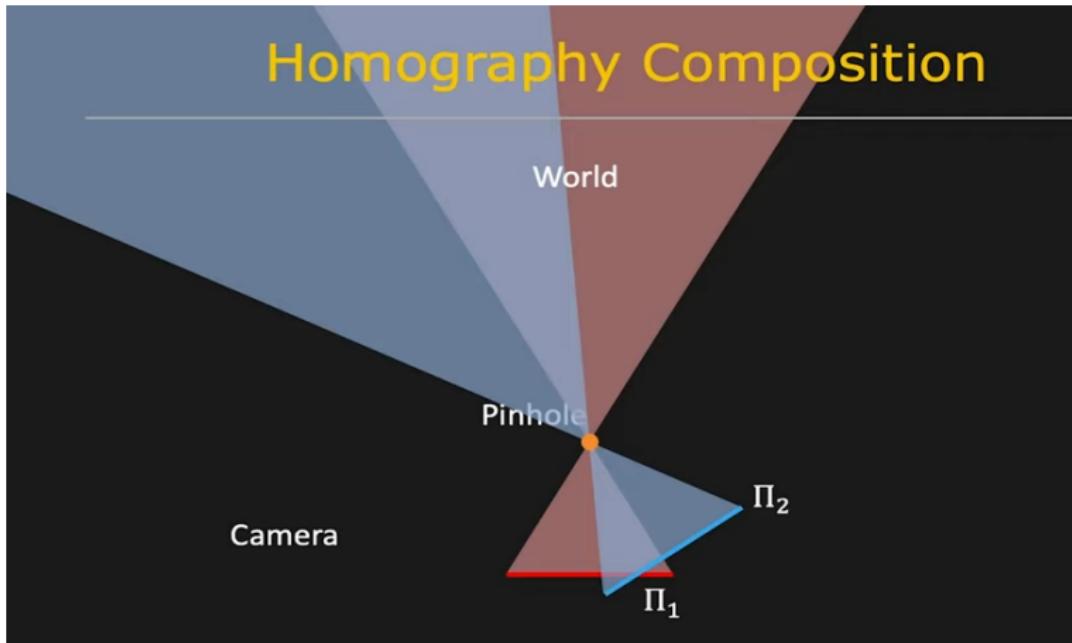
Department of Information Technology,
National Institute of Technology Karnataka, India

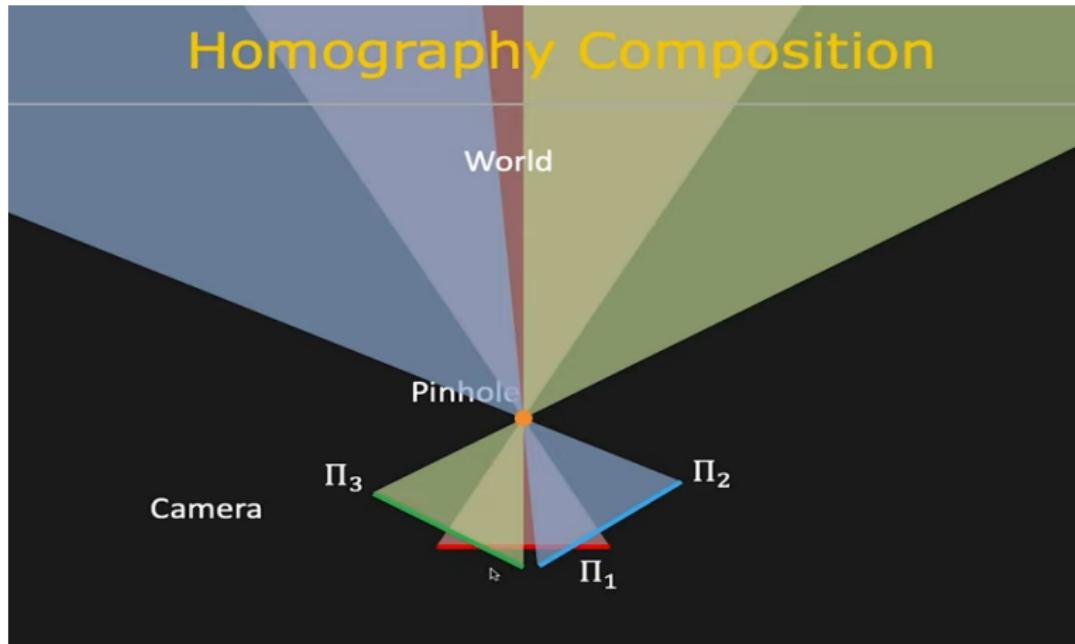
April 5, 2022

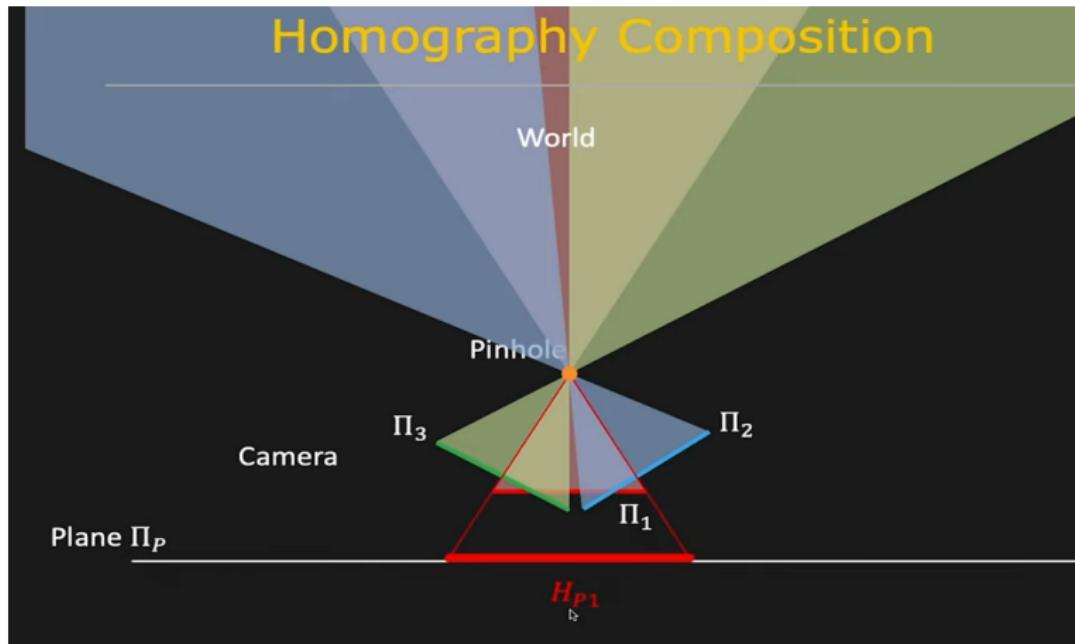
Homography Composition

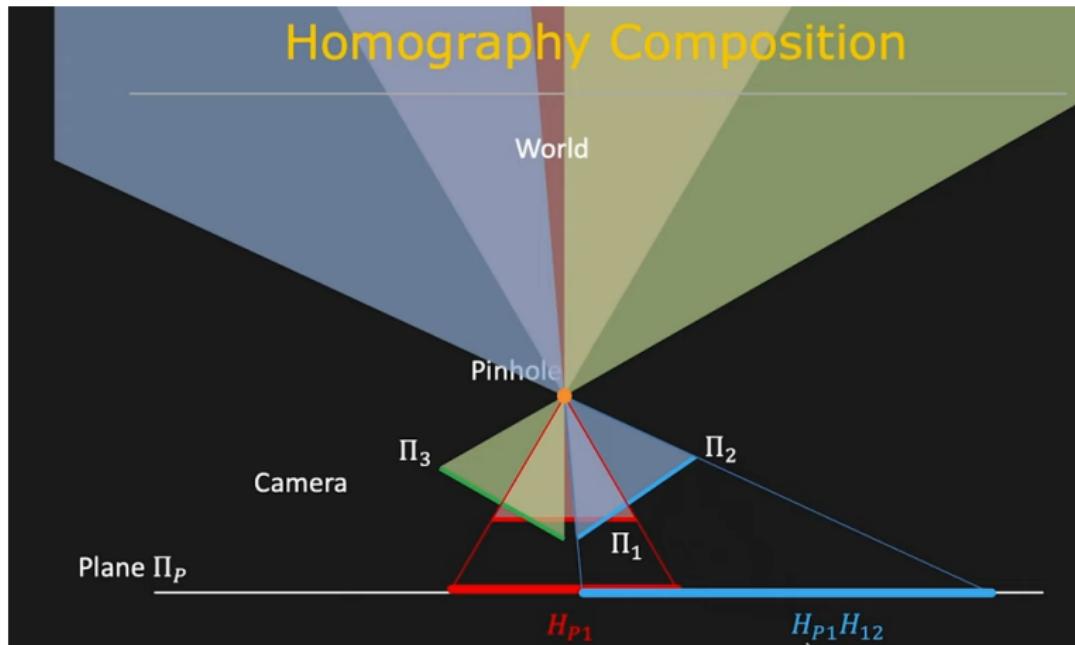


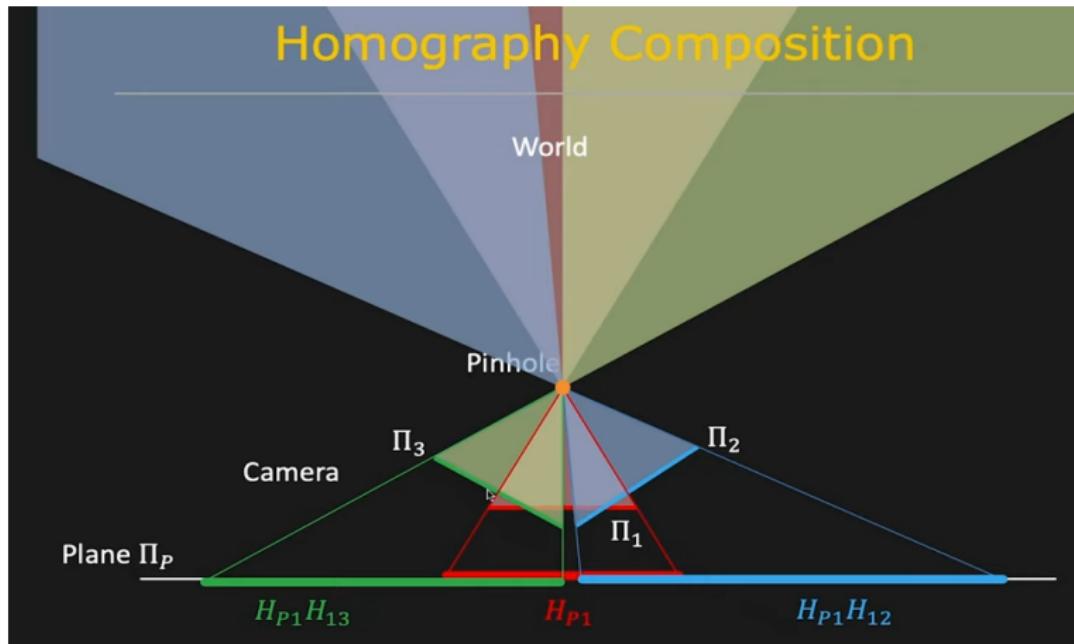
Homography Composition

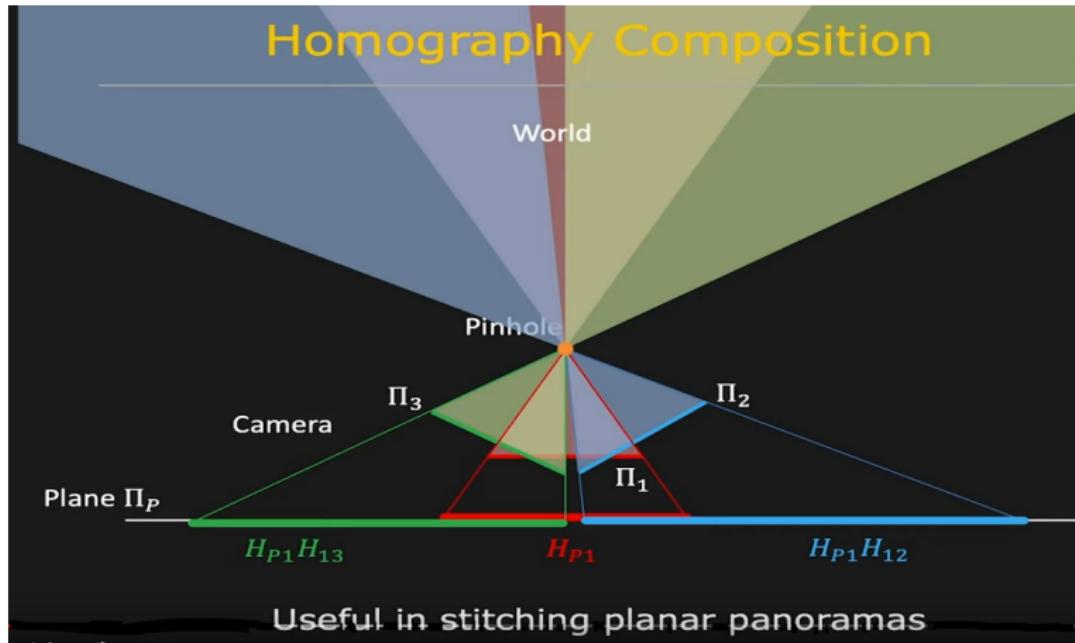




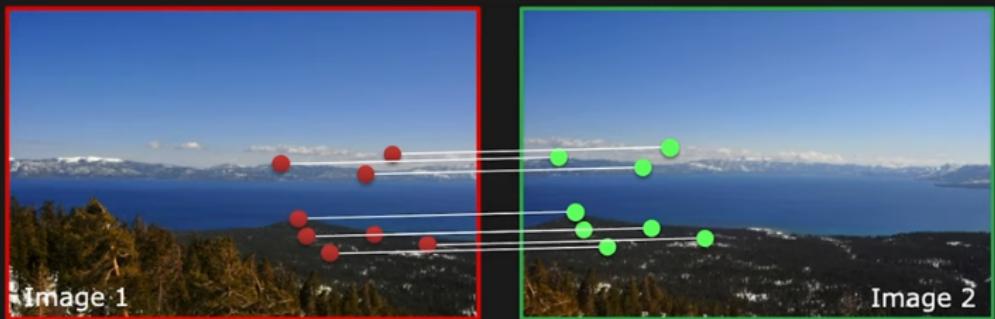








Computing Homography



Given a set of matching features/points between images 1 and 2, find the **homography H** that best “agrees” with the matches.

Computing Homography



Source Image



Destination Image

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \equiv \begin{bmatrix} \tilde{x}_d \\ \tilde{y}_d \\ \tilde{z}_d \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

How many unknowns?

Computing Homography

For a given pair i of corresponding points:

$$x_d^{(i)} = \frac{\tilde{x}_d^{(i)}}{\tilde{z}_d^{(i)}} = \frac{h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}}$$



$$y_d^{(i)} = \frac{\tilde{y}_d^{(i)}}{\tilde{z}_d^{(i)}} = \frac{h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}}{h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}}$$

Rearranging the terms:

$$x_d^{(i)} (h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}$$



$$y_d^{(i)} (h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}$$

Computing Homography

$$x_d^{(i)} (h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{11}x_s^{(i)} + h_{12}y_s^{(i)} + h_{13}$$

$$y_d^{(i)} (h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33}) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}$$

Rearranging the terms and writing as linear equation:

$$\begin{bmatrix} x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Computing Homography

$$x_d^{(i)} \left(h_{31} x_s^{(i)} + h_{32} y_s^{(i)} + h_{33} \right) = h_{11} x_s^{(i)} + h_{12} y_s^{(i)} + h_{13}$$

$$y_d^{(i)} \left(h_{31}x_s^{(i)} + h_{32}y_s^{(i)} + h_{33} \right) = h_{21}x_s^{(i)} + h_{22}y_s^{(i)} + h_{23}$$

Rearranging the terms and writing as linear equation:

$$\begin{bmatrix} x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

(Known)

(Unknown)

Computing Homography

Combining the equations for all corresponding points:

$$\left[\begin{array}{ccccccccc} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\ & & & & & \vdots & & & \\ x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \\ & & & & & \vdots & & & \\ x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\ 0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -y_d^{(n)}x_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)} \end{array} \right] = \left[\begin{array}{c} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{array} \right] = \left[\begin{array}{c} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{array} \right]$$

Computing Homography

Combining the equations for all corresponding points:

$$\begin{bmatrix} x_s^{(1)} & y_s^{(1)} & 1 & 0 & 0 & 0 & -x_d^{(1)}x_s^{(1)} & -x_d^{(1)}y_s^{(1)} & -x_d^{(1)} \\ 0 & 0 & 0 & x_s^{(1)} & y_s^{(1)} & 1 & -y_d^{(1)}x_s^{(1)} & -y_d^{(1)}y_s^{(1)} & -y_d^{(1)} \\ & & & & & \vdots & & & \\ x_s^{(i)} & y_s^{(i)} & 1 & 0 & 0 & 0 & -x_d^{(i)}x_s^{(i)} & -x_d^{(i)}y_s^{(i)} & -x_d^{(i)} \\ 0 & 0 & 0 & x_s^{(i)} & y_s^{(i)} & 1 & -y_d^{(i)}x_s^{(i)} & -y_d^{(i)}y_s^{(i)} & -y_d^{(i)} \\ & & & & & \vdots & & & \\ x_s^{(n)} & y_s^{(n)} & 1 & 0 & 0 & 0 & -x_d^{(n)}x_s^{(n)} & -x_d^{(n)}y_s^{(n)} & -x_d^{(n)} \\ 0 & 0 & 0 & x_s^{(n)} & y_s^{(n)} & 1 & -y_d^{(n)}x_s^{(n)} & -y_d^{(n)}y_s^{(n)} & -y_d^{(n)} \end{bmatrix} = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}$$

Solve for \mathbf{h} : $A \mathbf{h} = \mathbf{0}$ such that $\|\mathbf{h}\|^2 = 1$

Constrained Least Squares

Solve for \mathbf{h} : $A \mathbf{h} = \mathbf{0}$ such that $\|\mathbf{h}\|^2 = 1$

Define least squares problem:

$$\min_{\mathbf{h}} \|A\mathbf{h}\|^2 \text{ such that } \|\mathbf{h}\|^2 = 1$$

We know that:

$$\|A\mathbf{h}\|^2 = (A\mathbf{h})^T(A\mathbf{h}) = \mathbf{h}^T A^T A \mathbf{h} \quad \text{and} \quad \|\mathbf{h}\|^2 = \mathbf{h}^T \mathbf{h} = 1$$

▷

$$\min_{\mathbf{h}} (\mathbf{h}^T A^T A \mathbf{h}) \text{ such that } \mathbf{h}^T \mathbf{h} = 1$$

Constrained Least Squares

$$\min_{\mathbf{h}} (\mathbf{h}^T A^T A \mathbf{h}) \text{ such that } \mathbf{h}^T \mathbf{h} = 1$$

Define Loss function $L(\mathbf{h}, \lambda)$:

$$L(\mathbf{h}, \lambda) = \mathbf{h}^T A^T A \mathbf{h} - \lambda(\mathbf{h}^T \mathbf{h} - 1)$$

Taking derivatives of $L(\mathbf{h}, \lambda)$ w.r.t \mathbf{h} : $2A^T A \mathbf{h} - 2\lambda \mathbf{h} = \mathbf{0}$

$$A^T A \mathbf{h} = \lambda \mathbf{h} \quad \text{Eigenvalue Problem}$$

Eigenvector \mathbf{h} with smallest eigenvalue λ of matrix $A^T A$ minimizes the loss function $L(\mathbf{h})$.

Computer Vision-IT416

Dinesh Naik

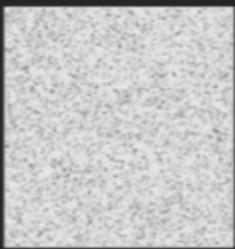
Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

Harris Corner Detection

Corners

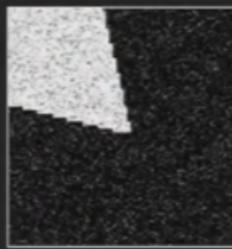
Corner: Point where Two Edges Meet. i.e., Rapid Changes of Image Intensity in Two Directions within a Small Region



"Flat" Region



"Edge" Region



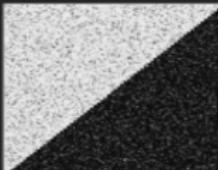
"Corner" Region

Image Gradients

Flat Region



Edge Region

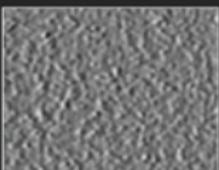


Corner Region

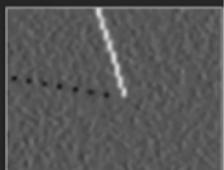
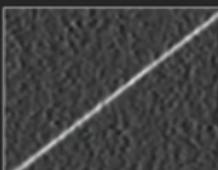
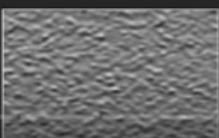


I

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



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



Distribution of Image Gradients

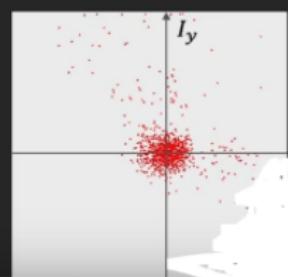
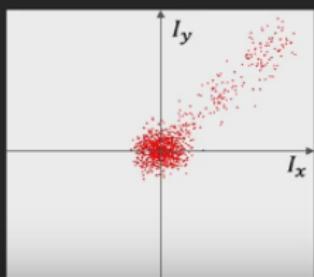
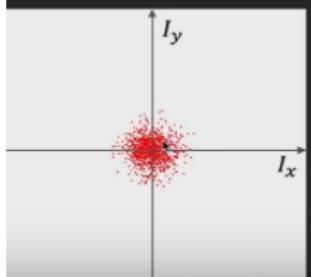
Flat Region



Edge Region



Corner Region

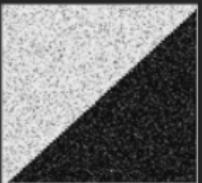


Fitting Elliptical Disk to Distribution

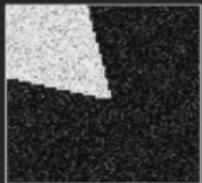
Flat Region



Edge Region



Corner Region



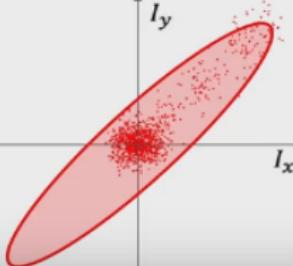
I_y

I_x



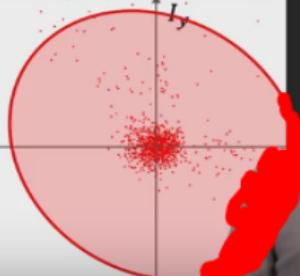
I_y

I_x



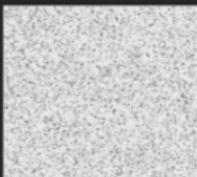
I_y

I_x



Fitting Elliptical Disk to Distribution

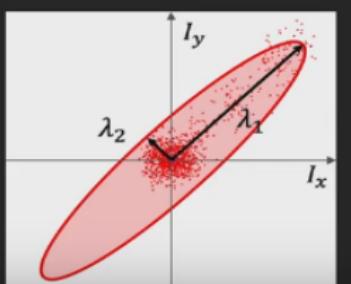
Flat Region



Edge Region

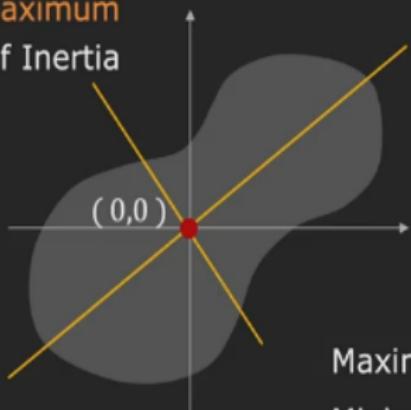


Corner Region



Fitting an Elliptical Disk

Axis of Maximum
Moment of Inertia



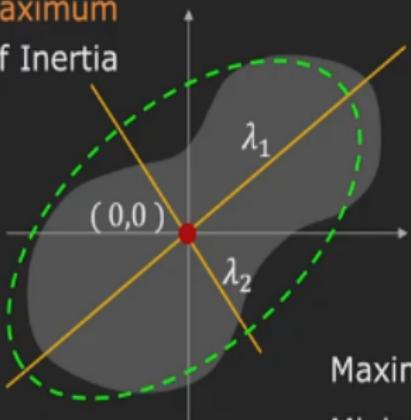
Axis of Minimum
Moment of Inertia

Maximum Moment of Inertia = E_{max}
Minimum Moment of Inertia = E_{min}

Fitting an Elliptical Disk

Axis of Maximum
Moment of Inertia

Axis of Minimum
Moment of Inertia



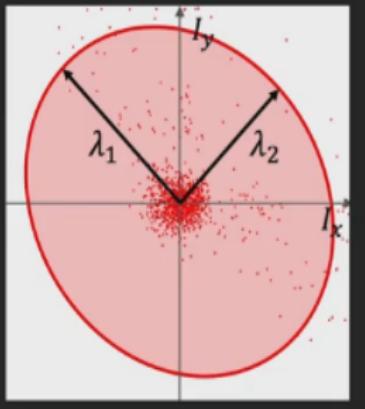
$$\begin{aligned}\text{Maximum Moment of Inertia} &= E_{max} \\ \text{Minimum Moment of Inertia} &= E_{min}\end{aligned}$$

Fitting an Elliptical Disk

Second Moments for a Region:

$$a = \sum_{i \in W} (I_{x_i})^2 \quad b = 2 \sum_{i \in W} (I_{x_i} I_{y_i})$$

$$c = \sum_{i \in W} (I_{y_i})^2 \quad W: \text{Window centered at pixel}$$



Second Moments for a Region:

$$a = \sum_{i \in W} (I_{x_i})^2 \quad b = 2 \sum_{i \in W} (I_{x_i} I_{y_i})$$

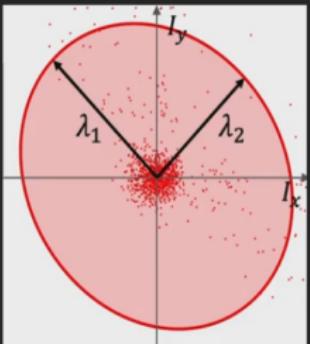
$$c = \sum_{i \in W} (I_{y_i})^2 \quad W: \text{Window centered at pixel}$$

(See lecture on Binary Images)

Ellipse Axes Lengths:

$$\lambda_1 = E_{max} = \frac{1}{2} [a + c + \sqrt{b^2 + (a - c)^2}]$$

$$\lambda_2 = E_{min} = \frac{1}{2} [a + c - \sqrt{b^2 + (a - c)^2}]$$



Interpretation of λ_1 and λ_2

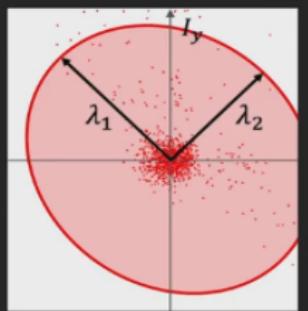
Flat Region



Edge Region



Corner Region

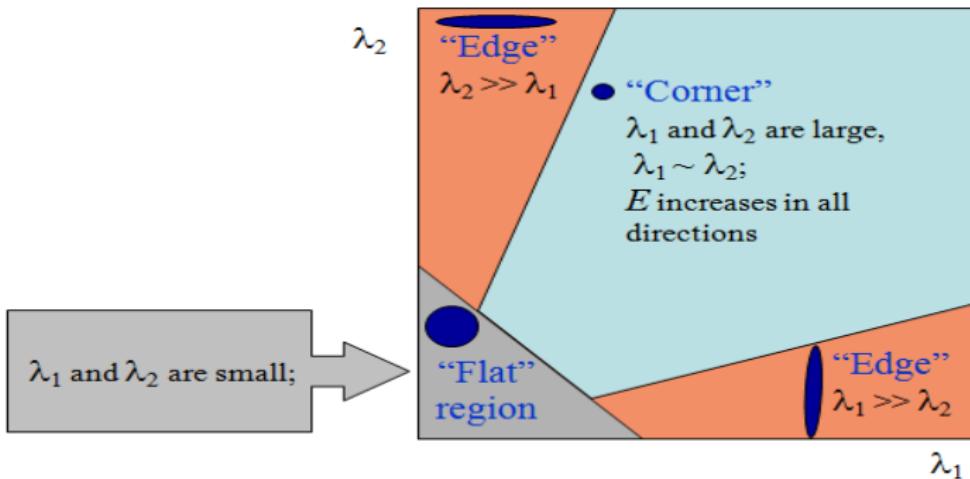


$\lambda_1 \sim \lambda_2$
Both are Small

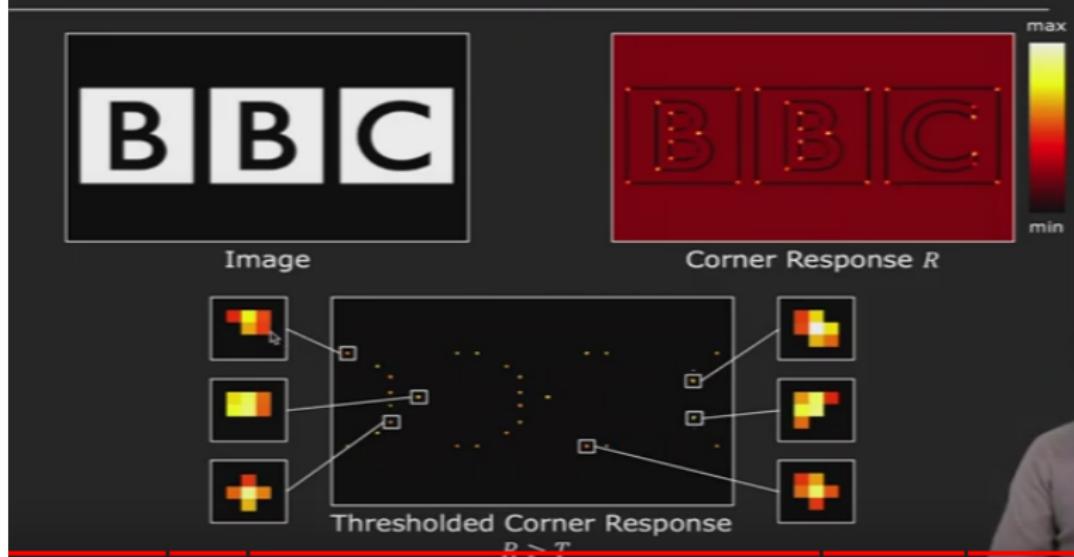
$\lambda_1 \gg \lambda_2$
 λ_1 is Large
 λ_2 is Small

$\lambda_1 \sim \lambda_2$
Both are Large

Harris Detector: Mathematics

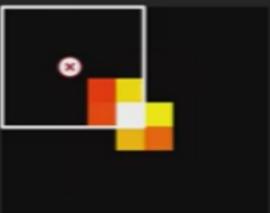


Harris Corner Detection Example

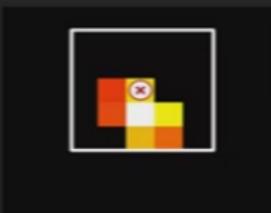


Non-Maximal Suppression

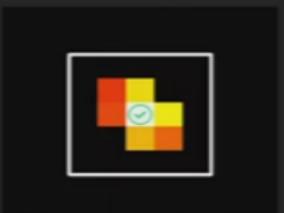
1. Slide a window of size k over the image.
2. At each position, if the pixel at the center is the maximum value within the window, label it as positive (retain it). Else label it as negative (suppress it).



Suppress



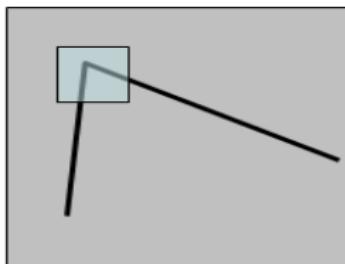
Suppress



Retain

The Basic Idea

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



Harris Detector: Mathematics

Measure of corner response:

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

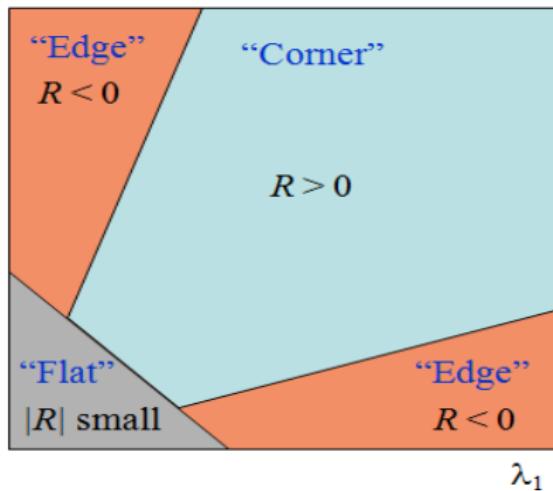
$$\det M = \lambda_1 \lambda_2$$

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

(k – empirical constant, $k = 0.04\text{-}0.06$)

Harris Detector: Mathematics

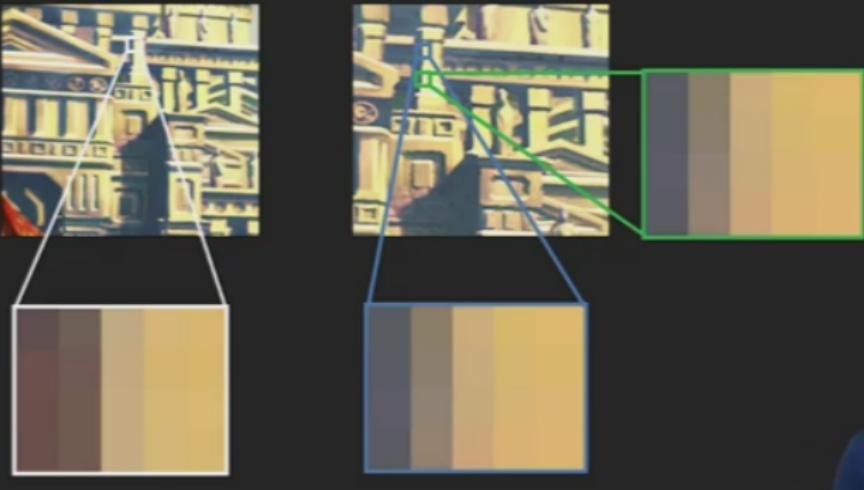
- R depends only on eigenvalues of M
- R is large for a corner
- R is negative with large magnitude for an edge
- $|R|$ is small for a flat region



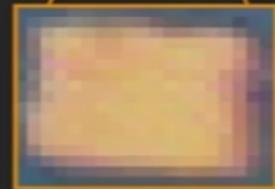
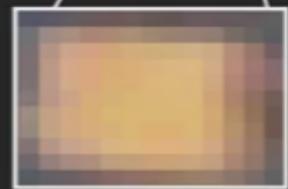
What is an Interesting Point/Feature?

- Has rich image content (brightness variation, color variation, etc.) within the local window
- Has well-defined representation (signature) for matching/comparing with other points
- Has a well-defined position in the image
- Should be invariant to image rotation and scaling
- Should be insensitive to lighting changes

Are Lines/Edges Interesting?



Are Blobs Interesting?



Blobs as Interest Points

For a **Blob-like Feature** to be useful, we need to:

- Locate the blob
- Determine its **size**
- Determine its **orientation**
- Formulate a **description** or
signature that is independent of
size and orientation

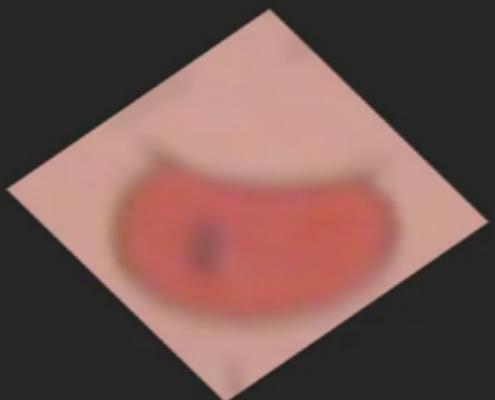
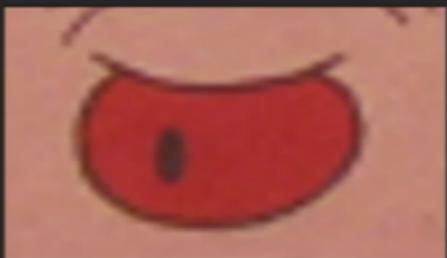


Raw Images are Hard to Match



Different size, orientation, lighting, brightness, etc.

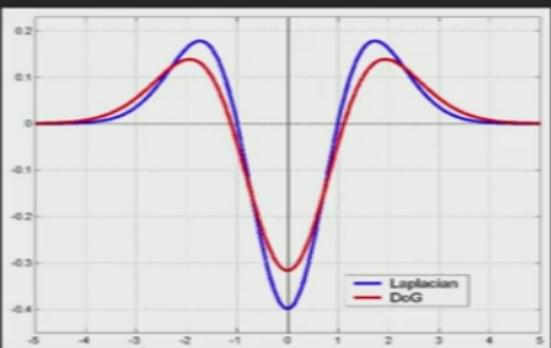
Removing Sources of Variation



Matching becomes easier if we can remove variations like size and orientation.

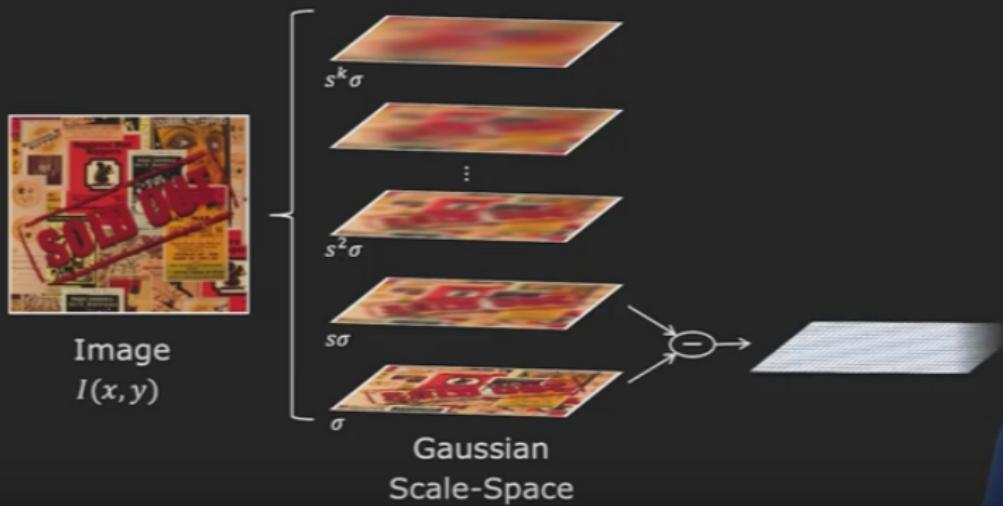
Fast NLoG Approximation: DoG

$$\text{Difference of Gaussian (DoG)} = (n_{s\sigma} - n_\sigma) \approx (s - 1)\sigma^2 \underbrace{\nabla^2 n_\sigma}_{\text{NLoG}}$$

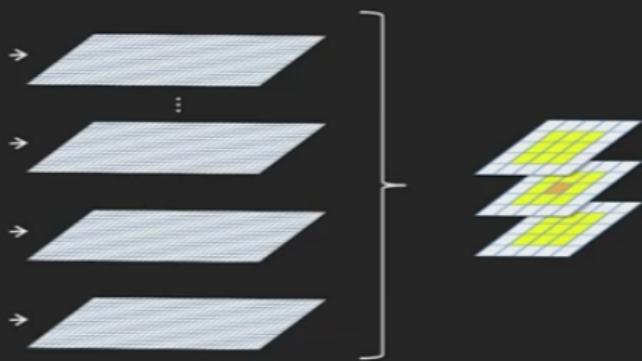


$$\text{DoG} \approx (s - 1) \text{ NLoG}$$

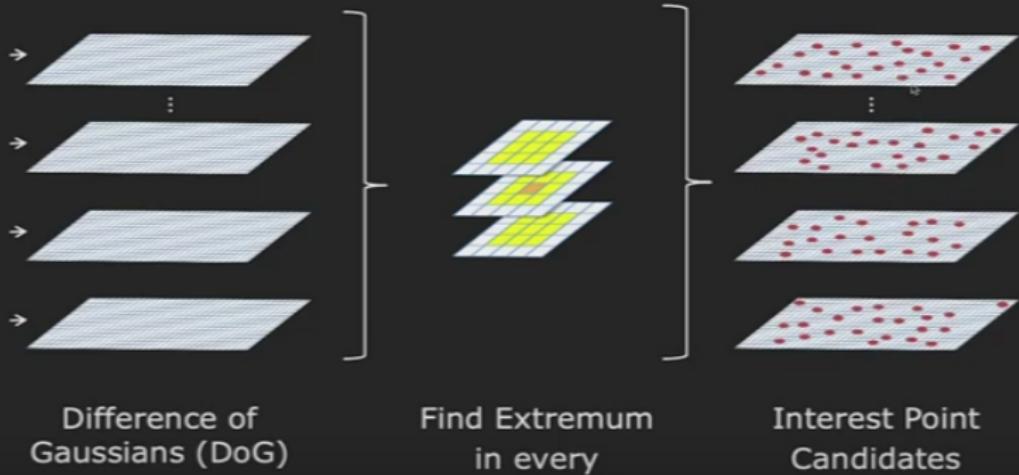
Extracting SIFT Interest Points



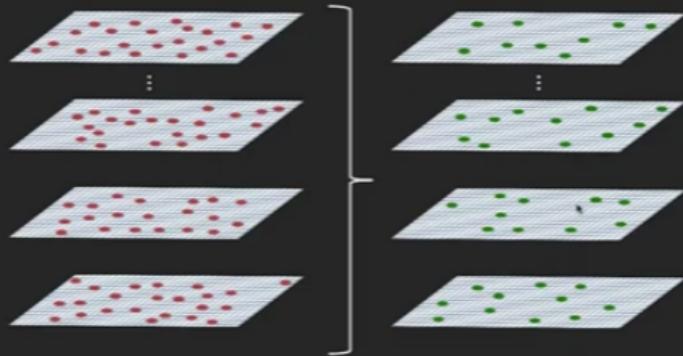
Extracting SIFT Interest Points



Extracting SIFT Interest Points



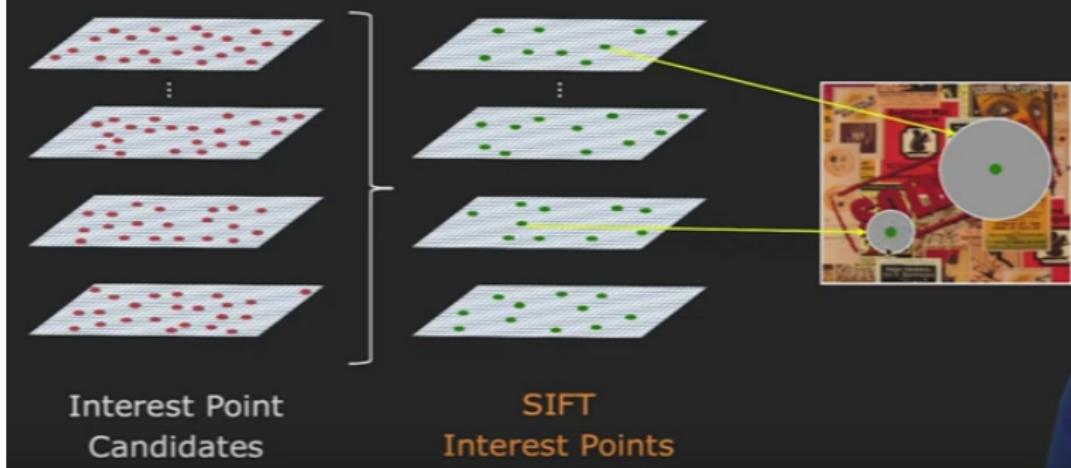
Extracting SIFT Interest Points



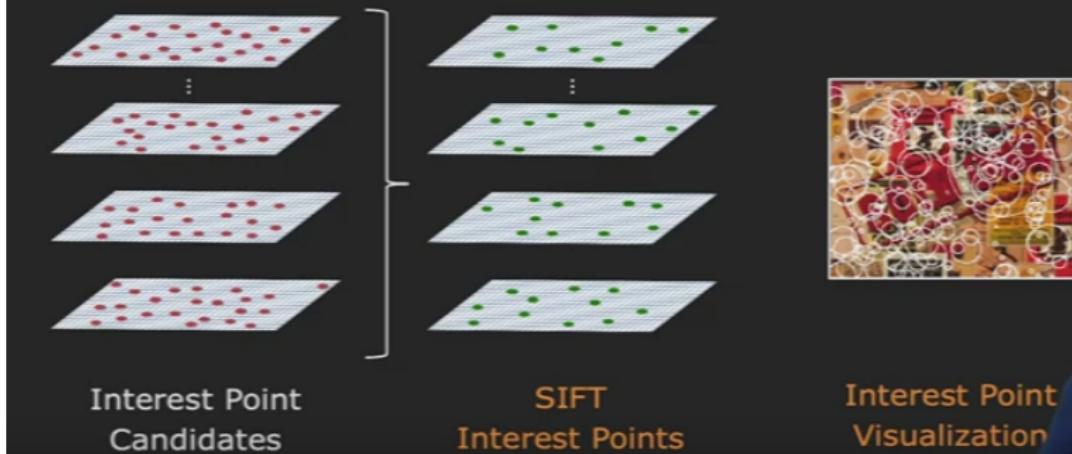
Interest Point
Candidates

SIFT
Interest Points
(final points)

Extracting SIFT Interest Points

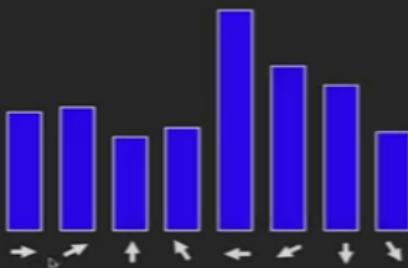
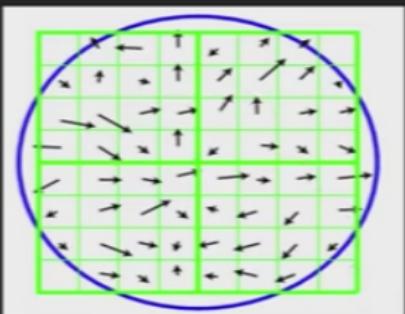


Extracting SIFT Interest Points



Computing the Principal Orientation

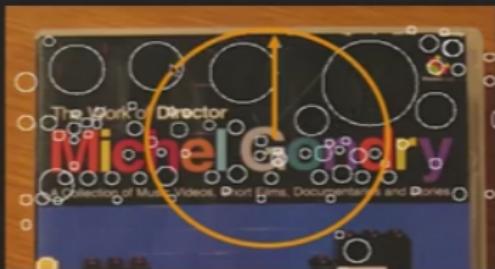
Use the histogram of gradient directions



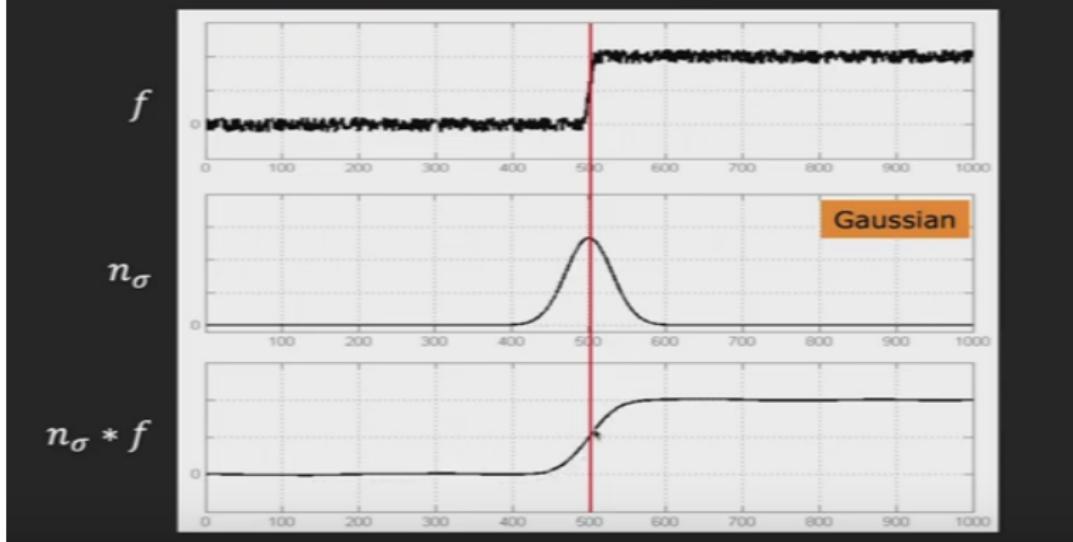
$$\theta = \tan^{-1} \left(\frac{\partial I}{\partial y} / \frac{\partial I}{\partial x} \right)$$

SIFT Rotation Invariance

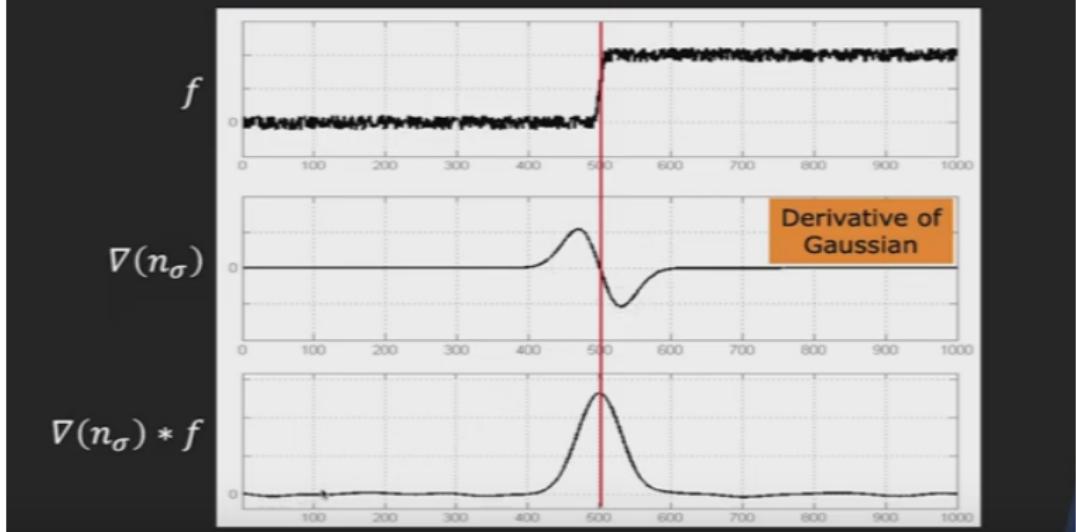
Use the principal orientation to undo rotation



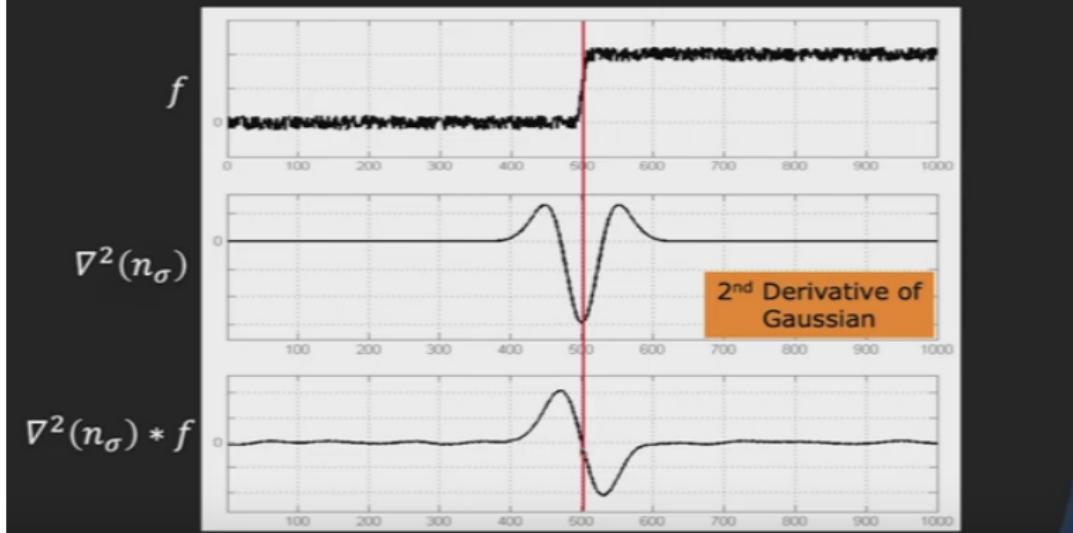
Review: Gaussian Filter



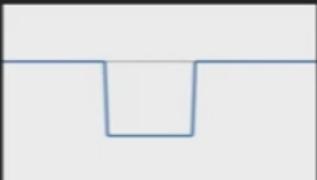
Review: Derivative of Gaussian



Review: 2nd Derivative of Gaussian

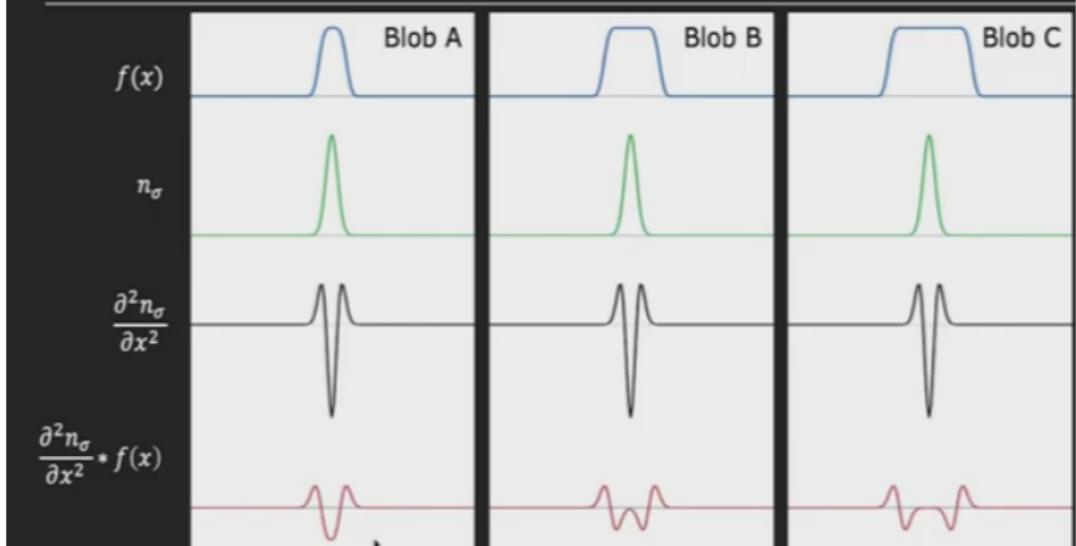


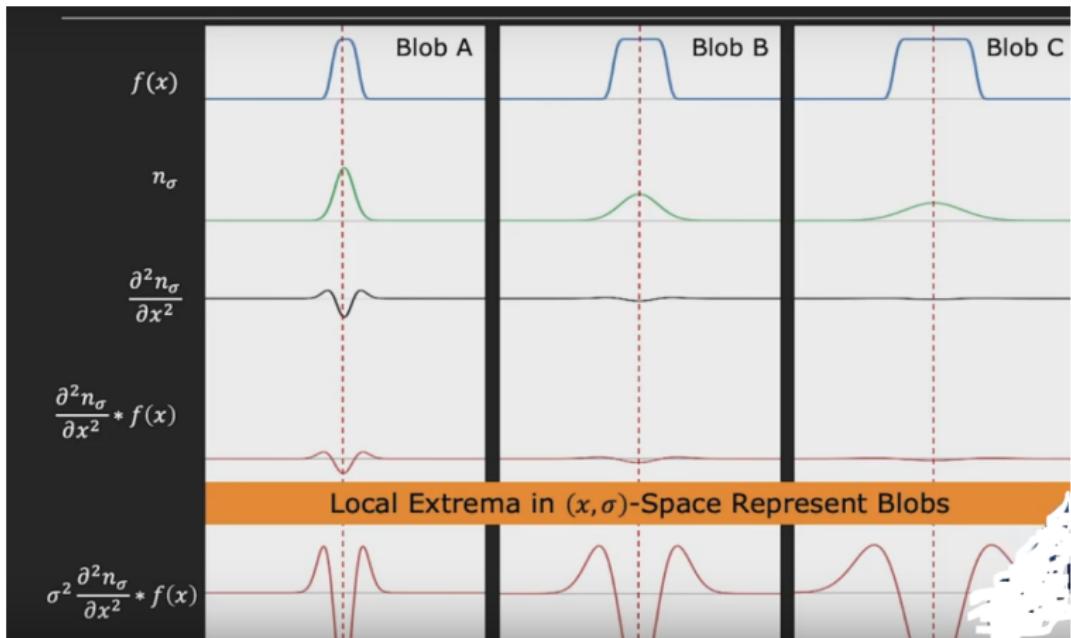
1D Blobs



Examples of 1D Blob-like structures

1D Blob and 2nd Derivative of Gaussian





1D Blob Detection Summary

Given: 1D signal $f(x)$

Compute: $\sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x)$ at many scales $(\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_k)$.

Find:
$$(x^*, \sigma^*) = \arg \max_{(x, \sigma)} \left| \sigma^2 \frac{\partial^2 n_\sigma}{\partial x^2} * f(x) \right|$$

x^* : Blob Position

σ^* : Characteristic Scale (Blob Size)

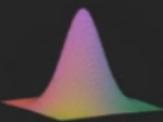
2D Blob Detector

Normalized Laplacian of Gaussian (NLoG) is used as the 2D equivalent for Blob Detection.

Laplacian

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

Gaussian



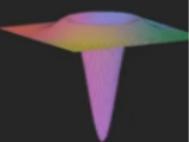
$$n_{\sigma}$$

LoG



$$\nabla^2 n_{\sigma}$$

NLoG



$$\sigma^2 \nabla^2 n_{\sigma}$$

Location of Blobs given by Local Extrema after applying Normalized Laplacian of Gaussian at many scales.

2D Blob Detection Summary

Given an image $I(x, y)$

Convolve the image using NLoG at many scales σ

Find:

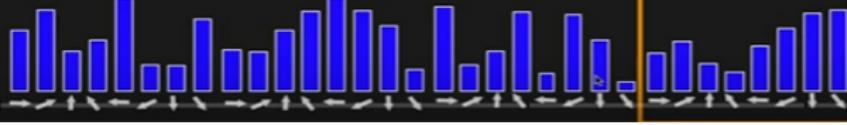
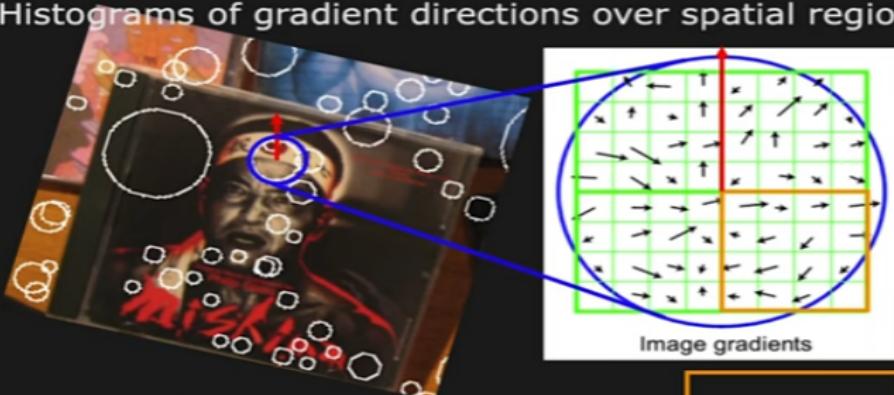
$$(x^*, y^*, \sigma^*) = \arg \max_{(x, y, \sigma)} |\sigma^2 \nabla^2 n_\sigma * I(x, y)|$$

(x^*, y^*) : Position of the blob

σ^* : Size of the blob

SIFT Descriptor

Histograms of gradient directions over spatial regions



Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N .

L2 Distance:

$$d(H_1, H_2) = \sqrt{\sum_k (H_1(k) - H_2(k))^2}$$

Smaller the distance metric, better the match.

Perfect match when $d(H_1, H_2) = 0$

Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N .

Normalized Correlation:

$$d(H_1, H_2) = \frac{\sum_k [(H_1(k) - \bar{H}_1)(H_2(k) - \bar{H}_2)]}{\sqrt{\sum_k (H_1(k) - \bar{H}_1)^2} \sqrt{\sum_k (H_2(k) - \bar{H}_2)^2}}$$

where: $\bar{H}_i = \frac{1}{N} \sum_{k=1}^N H_i(k)$

Larger the distance metric, better the match.

Perfect match when $d(H_1, H_2) = 1$

Comparing SIFT Descriptors

Essentially comparing two arrays of data.

Let $H_1(k)$ and $H_2(k)$ be two arrays of data of length N .

Intersection:

$$d(H_1, H_2) = \sum_k \min(H_1(k), H_2(k))$$

Larger the distance metric, better the match.

Speeded Up Robust Features (SURF)

- In computer vision, Speeded Up Robust Features (SURF) is a patented local feature detector and descriptor.
- It can be used for tasks such as object recognition, image registration, classification, or 3D reconstruction.
- It is partly inspired by the scale-invariant feature transform (SIFT) descriptor.
- The SURF method (Speeded Up Robust Features) is a fast and robust algorithm for local, similarity invariant representation and comparison of images.

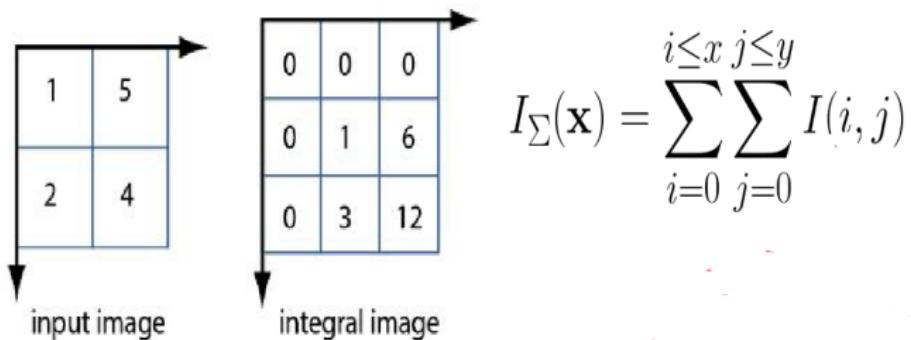
What is SURF?

SURF (**S**peeded-**U**p **R**obust **F**eatures) is a feature detection framework introduced by Herbert Bay and his colleagues at ETH Zurich. SURF interest points are in-plane rotation-invariant, robust to noise, and overall, extremely fast to calculate. This procedure can be divided into three steps:

1. Interest Point Detection
2. Interest Point Description
3. Interest Point Matching

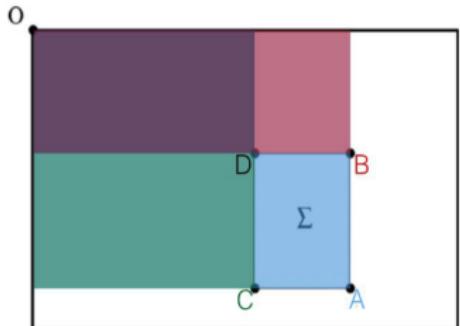
Detection: Integral Images

Integral images are an image transform such that any entry of an integral image $I_{\Sigma}(\mathbf{x})$ at a location $\mathbf{x} = (x, y)^T$ represents the sum of all pixels in the input image I within a rectangular region formed by the origin and \mathbf{x} .



Detection: Integral Images

Integral images are incredibly efficient. It is possible to characterize a region of the image using four memory accesses and three operations. This makes it very cheap to detect blobs.



$$\Sigma = A - B - C + D$$

Detection: Hessian-Based Interest Points

The detector detects blob-like structures at locations where the determinant of the Hessian is maximum.

The Hessian is defined as such:

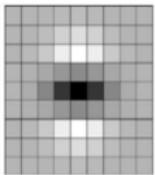
$$H(\mathbf{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix}$$

where $L_{xx}(\mathbf{x}, \sigma)$ is the convolution of the Gaussian second order derivative $\frac{\partial^2}{\partial x^2}g(\sigma)$ with the image I in point x .

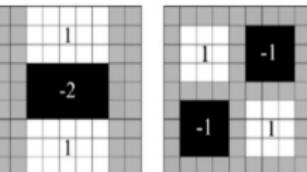
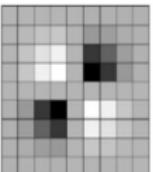
Detection: Hessian Approximation

The actual computation of the Hessian matrix is expensive and slow. Instead, the Hessian can be approximated using box filters!

$$\det(H_{approx}) = D_{xx}D_{yy} - (wD_{xy})^2 \quad \text{where } D_{xx} \text{ is the approximation of the Gaussian second order partial derivative in the x-direction and } w = 0.9.$$



The Gaussian second order partial derivative in y- and xy-direction.



Box filter approximations of the Gaussian second order partial derivatives.

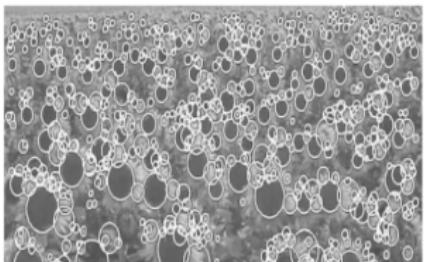
Detection: Scale Space Representation

To match interest points across different scales, a pyramidal scale space is built. Rather than serial downsampling, each successive level of the pyramid is built by upscaling the image in parallel. Each scale is defined as the response of the image convolved with a box filter of a certain dimension (9x9, 15x15, 27x27 etc.). The scale space is further divided into octaves (sets of filter responses).



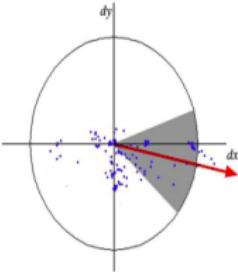
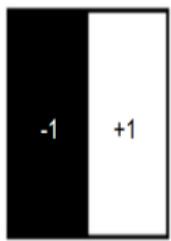
Detection: Interest Point Localization

To localize interest points in the image and over scales, a non-maximum suppression (non-maximum pixels are set to 0) in a $3 \times 3 \times 3$ neighborhood is applied. The maxima of the determinant of the Hessian matrix are then interpolated in scale and image space.



Descriptor: Orientation Assignment

The Haar wavelet responses in x- and y-direction within a circular neighborhood with radius $6s$ is calculated. Responses are weighted with a Gaussian ($\sigma = 2s$) centered at the interest point and then the directional strengths are plotted. These plots are then divided into sliding orientation windows and local orientation vectors are computed as the sum of the x and y responses within each window. The dominant orientation is the largest of all such vectors across all windows.



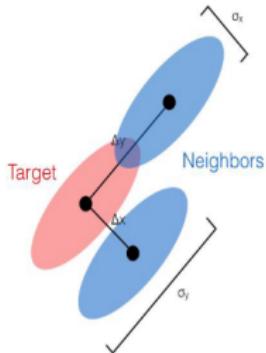
Descriptor: Feature Vector

To extract features, an axis-orientated square window of size 20s and centered around the interest point is defined. This window is subdivided into a 4 x 4 grid. The “horizontal” and “vertical” Haar wavelet response is calculated over each subdivision and four metrics are extracted from each subdivision using 5 x 5 equally spaced points. These metrics are then summed to produce the local feature vector. These local feature vectors are concatenated to form a 64-element feature vector describing the interest point and surrounding neighborhood.

$$\mathbf{v} = \begin{bmatrix} \Sigma d_x \\ \Sigma d_y \\ \Sigma |d_x| \\ \Sigma |d_y| \end{bmatrix} \quad \begin{array}{l} \text{where } d_x \text{ is the “horizontal” Haar wavelet response} \\ \text{and } d_y \text{ is the “vertical” Haar wavelet response} \end{array}$$

Matching: Nearest Neighbors

Features are matched across frames as the nearest neighbor within a distinct feature threshold. Either Euclidean or Mahalanobis distance may be used to determine “nearest”. In this implementation, uniform precision was assumed and, therefore, Euclidean distance was sufficient.



$$D_E = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

$$D_M = \sqrt{\frac{(x_2 - x_1)^2}{\sigma_x^2} + \frac{(y_2 - y_1)^2}{\sigma_y^2}}$$

Integral Image

A table that holds the sum of all pixel values to the left and top of a given pixel, **inclusive**.

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

Integral Image

A table that holds the sum of all pixel values to the left and top of a given pixel, inclusive.

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	966	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

Integral Image

A table that holds the sum of all pixel values to the left and top of a given pixel, inclusive.

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image II

Summation Within a Rectangle

Fast summations of arbitrary rectangles using integral images

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image *I*

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image *II*

Summation Within a Rectangle

Fast summations of arbitrary rectangles using integral images

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image *I*

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

Integral Image *II*

$$\begin{aligned} \text{Sum} &= II_P + \dots \\ &= 3490 + \dots \end{aligned}$$

Summation Within a Rectangle

Fast summations of arbitrary rectangles using integral image

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4204
680	1449	2433	3253	4118	5052

Integral Image II

$$\begin{aligned} \text{Sum} &= II_P - II_Q + \dots \\ &= 3490 - 1137 + \dots \end{aligned}$$

Summation Within a Rectangle

Fast summations of arbitrary rectangles using integral image.

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image I

98	208	329	454	576	705
197	417	658	899	1137	1395
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

S P Q Integral Image II

$$\begin{aligned} \text{Sum} &= II_P - II_Q - II_S + \dots \\ &= 3490 - 1137 - 1249 + \dots \end{aligned}$$

Summation Within a Rectangle

Fast summations of arbitrary rectangles using integral images

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

Image *I*

98	208	329	454	576	705
294	623	988	1340	1701	2093
392	833	1330	1790	2274	2799
489	1043	1687	2255	2864	3531
584	1249	2061	2751	3490	4294
680	1449	2433	3253	4118	5052

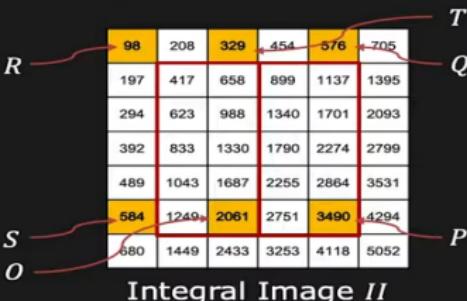
Integral Image *II*

$$\begin{aligned} \text{Sum} &= II_P - II_Q - II_S + II_R \\ &= 3490 - 1137 - 1249 + 417 = 1521 \end{aligned}$$

Haar Response using Integral Image

98	110	121	125	122	129
99	110	120	116	116	129
97	109	124	111	123	134
98	112	132	108	123	133
97	113	147	108	125	142
95	111	168	122	130	137
96	104	172	130	126	130

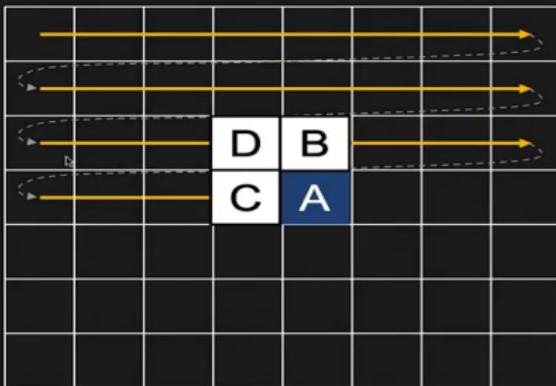
Image I



$$\begin{aligned}
 V_A &= \sum(\text{pixel intensities in white}) - \sum(\text{pixel intensities in black}) \\
 &= (II_O - II_T + II_R - II_S) - (II_P - II_Q + II_T - II_O) \\
 &= (2061 - 329 + 98 - 584) - (3490 - 576 + 329 - 2061) = 64
 \end{aligned}$$

Computational Cost: Only 7 additions

Computing Integral Image



Raster
Scanning

Let I_A and II_A be the values of Image and Integral Image, respectively, at pixel A.

$$II_A = II_B + II_C - II_D + I_A$$

Haar Features Using Integral Images

Integral image needs to be computed once per test image.
Allows fast computations of Haar features.



$$\otimes \begin{bmatrix} H_A \\ H_B \\ H_C \\ H_D \\ \vdots \end{bmatrix} = \begin{bmatrix} V_A[i, j] \\ V_B[i, j] \\ V_C[i, j] \\ V_D[i, j] \\ \vdots \end{bmatrix}$$

Input Image

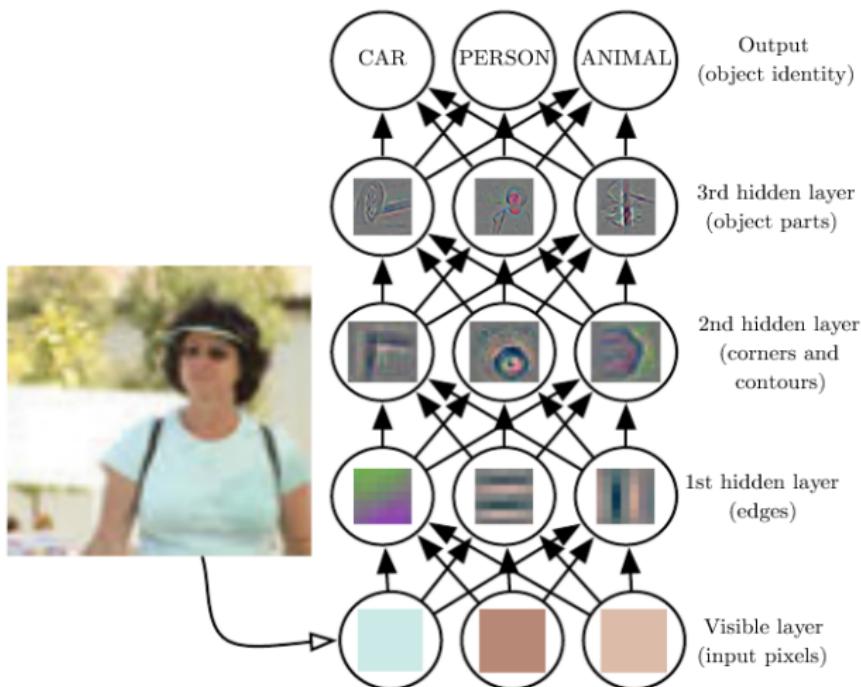
Deep Learning Basics -IT416

Dinesh Naik

Department of Information Technology,
National Institute of Technology Karnataka, India

April 5, 2022

Illustration of a Deep Learning model.



Flowcharts showing how the different parts of an AI system relate to each other within different AI disciplines. Shaded boxes indicate components that are able to learn from data.

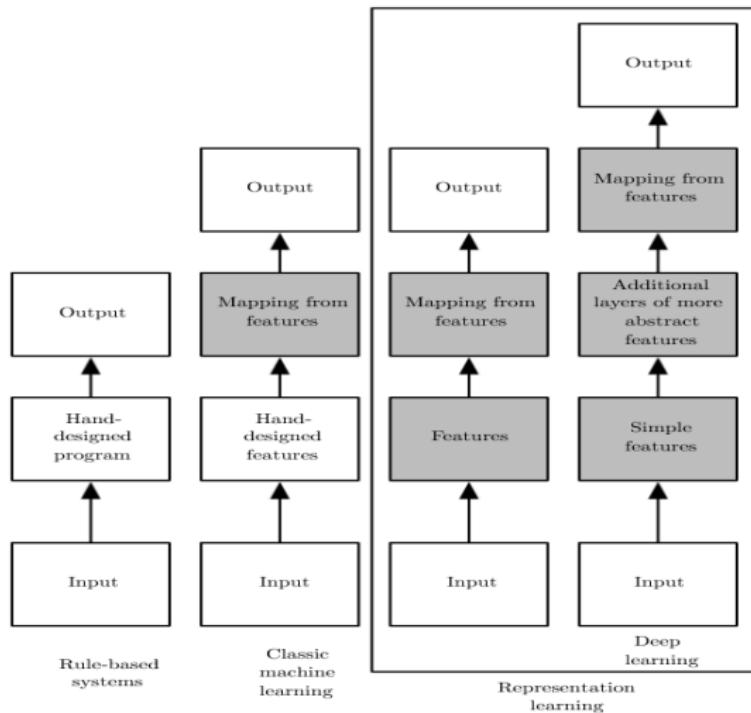


Image and Video Captioning

- Image captioning is a popular research area of artificial intelligence (AI) that deals with image understanding and a language description for that image.
- Image understanding entails detecting and recognizing objects, as well as understanding scene type or location, object properties, and their interactions.
- Generating well-formed sentences requires both syntactic and semantic understanding of the language
- Understanding an image largely depends on obtaining image features. The techniques used for this purpose can be broadly divided into two categories:
 - (1) traditional machine-learning-based techniques and
 - (2) deep machine-learning-based techniques.

Traditional machine-learning-based

- In traditional machine learning, handcrafted features such as local binary patterns (LBPs), scale-invariant feature transform (SIFT), the histogram of oriented gradients (HOG), and a combination of such features are widely used.
- In these techniques, features are extracted from input data. They are then passed to a classifier such as support vector machines (SVMs) in order to classify an object.
- Since handcrafted features are task specific, extracting features from a large and diverse set of data is not feasible.
- Moreover, real-world data such as images and video are complex and have different semantic interpretations.

Deep machine-learning-based techniques

- Deep machine-learning-based techniques, features are learned automatically from training data and they can handle a large and diverse set of images and videos.
- For example, convolutional neural networks (CNNs) are widely used for feature learning, and a classifier such as Softmax is used for classification.
- CNN is generally followed by recurrent neural networks (RNNs) in order to generate captions.

Image/Video captioning articles into three main categories

- (1) template-based image captioning,
- (2) retrieval-based image captioning, and
- (3) novel image caption generation.
- Most deep-learning-based image captioning methods fall into the category of novel caption generation

Deep-Learning-based image captioning method

- (1) visual space based,
- (2) multimodal space based,
- (3) supervised learning,
- (4) other deep learning,
- (5) dense captioning,
- (6) whole scene based,
- (7) encoder-decoder architecture based,
- (8) compositional architecture based,
- (9) LSTM (Long Short-Term Memory) language model based,
- (10) other language model based,
- (11) attention based,
- (12) semantic concept based,
- (13) stylized captions, and
- (14) novel-object-based.