

# ACDS Lecture Series

---

Lecture - 15

CSIR

## **Image Processing and Computer Vision**

**G. N. Sastry and Team**

ADVANCED COMPUTATION AND DATA SCIENCES (ACDS) DIVISION

CSIR-North East Institute of Science and Technology, Jorhat, Assam, India

## 13.1 Introduction

### 13.1.1 Motivation

### 13.1.2 Image Processing vs. Computer Vision

### 13.1.3 Requirements and Set up

### 13.1.4 Datasets and Resources

## 13.2 Preprocessing

### 13.2.1 Reading an Image

### 13.2.2 Image Color Conversions

### 13.2.3 Image Transformation

### 13.2.4 Image Blending

### 13.2.5 Thresholding

### 13.2.6 Blurring and Smoothing

### 13.2.7 Morphological Operators

### 13.2.8 Image Filtering

### 13.2.9 Image Gradients

### 13.2.10 Histograms

### 13.2.11 Exercises

## 13.3 Feature Extraction

**13.3.1 Features and Its Importance**

**13.3.2 Harris Corner Detection**

**13.3.3 FAST Feature Detection**

**13.3.4 ORB Feature Detection**

**13.3.5 Black Box Feature**

**13.3.6 SIFT**

**13.3.7 Exercises**

## 13.4 Deep Learning in Computer Vision

**13.4.1 Deep Learning Models**

**13.4.2 CNN Topology**

**13.4.3 Optimization**

**13.4.4 Choosing Hyperparameters**

**13.4.5 Regularization**

**13.4.6 Data Augmentation**

**13.4.7 Transfer learning**

**13.4.8 Exercises**

## 13.5 Object Detection

**13.5.1 Mechanism**

**13.5.2 Corner Detection**

**13.5.3 Edge Detection**

**13.5.4 Grid Detection**

**13.5.5 Contour Detection**

**13.5.6 Template Matching**

**13.5.7 Feature Matching**

**13.5.8 Yolo**

**13.5.9 Face Detection**

**13.5.10 Exercises**

## 13.6 Segmentation

**13.6.1 Introduction**

**13.6.2 Graph Based Segmentation**

**13.6.3 Watershed Algorithm**

**13.6.4 Deep Learning for Segmentation**

**13.6.5 Exercises**

## 13.7 Object Tracking

**13.7.1 Introduction**

**13.7.2 Challenges**

**13.7.3 MOSSE tracker**

**13.7.4 Deep SORT**

**13.7.5 Optical Flow**

**13.7.6 Meanshift and Camshift**

**13.7.7 Tracking APIs**

**13.7.8 Exercises**

## 13.8 References

**13.8.1 Books**

**13.8.2 Video Lectures**

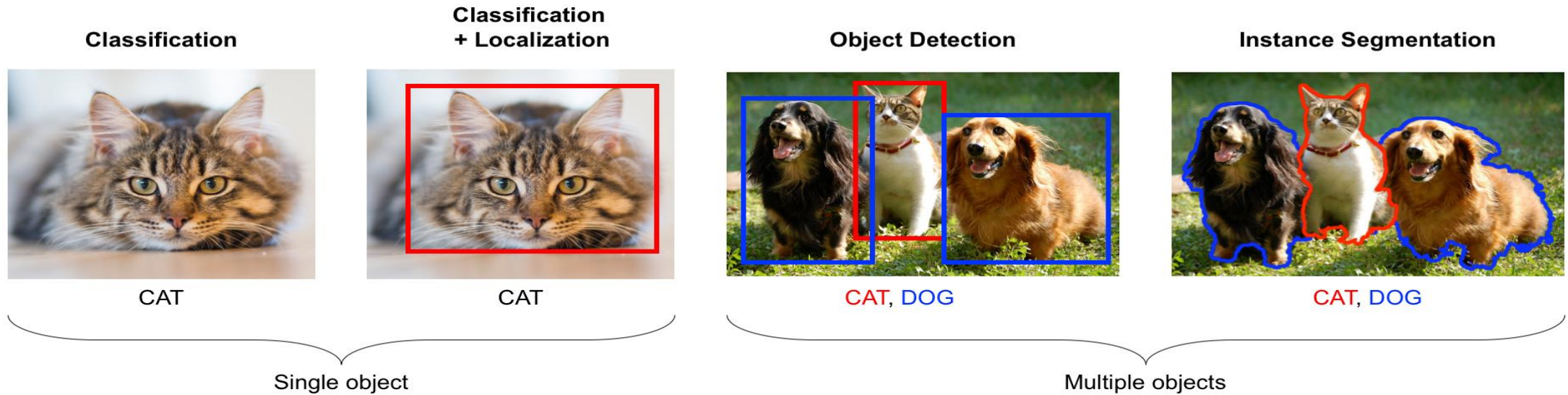
## **The Topics Covered In This Section**

13.1.1 Motivation

13.1.2 Image Processing vs. Computer Vision

13.1.3 Requirements and Setup

13.1.4 Datasets and Resources



- **Image Classification:** Label an image with a predefined set of categories based on objects present.
- **Segmentation:** Break up an image into cluster regions according to object occupancy and property.
- **Object Detection:** Detecting and categorizing the object.
- **Object Tracking:** Define the location and keeps track of objects across a sequence of images.
- **Image Reconstruction:** Extracting the information about geometry in the image.
- **Image Geometry:** Computing the depth of objects from the camera.
- **Image Generation:** Merging different image styles or generating completely new ones.

## 13.1.2 Image Processing vs. Computer Vision

Image Processing	Computer Vision
Mainly focused on processing the raw input images to enhance them or preparing them by tuning many parameter and features of the images to do other tasks.	Focused on extracting information from input images or videos to have a proper high-level understanding of them to predict visual input for automating tasks like human visual system.
Uses methods like transformation, Different Filtering, Hidden Markov models, Independent component analysis, etc.	Image processing is one of the methods that is used for computer vision along with other Machine learning techniques, CNN etc.
Examples of applications are - Rescaling image (Digital Zoom), Correcting illumination etc.	Examples of applications are - Object detection, Face detection, Hand writing recognition etc.
A subset of Computer Vision.	A superset of Image Processing.



### Development Platform

- Python and Anaconda IDE

### Tools and Libraries

- OpenCV: consists of most common algorithms of computer vision.
- TensorFlow and Keras: For more recent techniques of deep learning.
- Panda, NumPy, Matplotlib, seaborn, SciPy, scikit-learn: For Machine learning task

### Note:

- Hardware Requirements: For deep learning, a GPU is highly recommended.
- To run our programs using GPUs, install both CUDA and cuDNN binaries provided by Nvidia.

### OpenCV (Open Source Computer Vision)

- A library of programming functions mainly aimed at real-time computer vision.
- Created by Intel in 1999, it is written in C++
- There are bindings in Python, Java and MATLAB/OCTAVE.
- It contains many popular algorithms for computer vision built-in.
- It supports models from deep learning frameworks like TensorFlow, Torch, PyTorch and Caffe.
- Official Link: <https://opencv.org/>

### **TensorFlow for deep learning**

- Popular deep learning libraries available and has APIs for Python, C++, Java, and so on.
- separate version for GPU is available.

### **Keras for deep learning**

- Keras is a Python based API that uses TensorFlow, CNTK, or Theano as backend for deep learning.
- Due to its high level API and simplified abstraction, it is quite popular in deep learning community.
- To install this, first install TensorFlow as described in previous section, and use the following:
- There is no separate version for GPU available.

### MNIST

- Go to dataset for starting machine learning or deep learning.
- A dataset for handwritten digits (0-9) with 60,000 training images and 10,000 test images.
- The size of each image is 28 x 28 and each has 2 color channel (gray).
- It is provided in most of the frameworks and there is no need to download it separately.

### CIFAR-10

- A complex dataset consists of 10 categories of images with 60,000 training images and 10,000 test images, uniformly from each category.
- The labels in order: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.
- The size of each image is 32 x 32 and each has 3 color channels (*i.e. color dataset*).
- It is provided in most of the frameworks and there is no need to download it separately.

### Pascal VOC

- Gained in popularity as one of the major datasets for object recognition and segmentation.
- The dataset is also usually referred to by year; for example, VOC2012 refers to 2012.
- The available categories in this dataset are aeroplane, bicycle, boat, bottle, bus, car, cat, chair, cow, dining table, dog, horse, motorbike, person, potted plant, sheep, train, and TV.
- Official link: <http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html>

### MSCOCO

- Dataset for object recognition, with 80 categories and 330K images.
- After Pascal VOC'12, this became a popular benchmark for training and evaluating the system.
- Official link: <http://cocodataset.org/#download>

### ImageNet

- One of the largest annotated datasets for computer vision arranged for non-commercial use.
- Used to create image classification models due to availability of large number of varied images.
- There are 1,000 classes with 1.4 million images overall.
- Official link: <http://image-net.org/download>

### TUM RGB-D

- Used to understand the geometry of a scene.
- RGB-D refers to a dataset with both RGB (color) images and Depth images.
- The depth here refers to distance of pixel from camera and are taken using a depth camera.
- Because of depth information, this dataset is used to evaluate depth based SLAM algorithms and three-dimensional reconstructions.
- Official link: <https://vision.in.tum.de/data/datasets/rgbd-dataset/download>

### **Conferences (to look for latest research and applications)**

- Conference on Computer Vision and Pattern Recognition (CVPR)
- International Conference on Computer Vision (ICCV)
- Asian Conference on Computer Vision (ACCV)
- European Conference on Computer Vision (ECCV)
- International Conference on Machine Learning (ICML)

### **The Topics Covered In This Section**

- 13.2.1 Reading an Image
- 13.2.2 Image Color Conversions
- 13.2.3 Image Transformation
- 13.2.4 Image Blending
- 13.2.5 Thresholding
- 13.2.6 Blurring and Smoothing
- 13.2.7 Morphological Operators
- 13.2.8 Image Filtering
- 13.2.9 Image Gradients
- 13.2.10 Histograms
- 13.2.11 Exercises



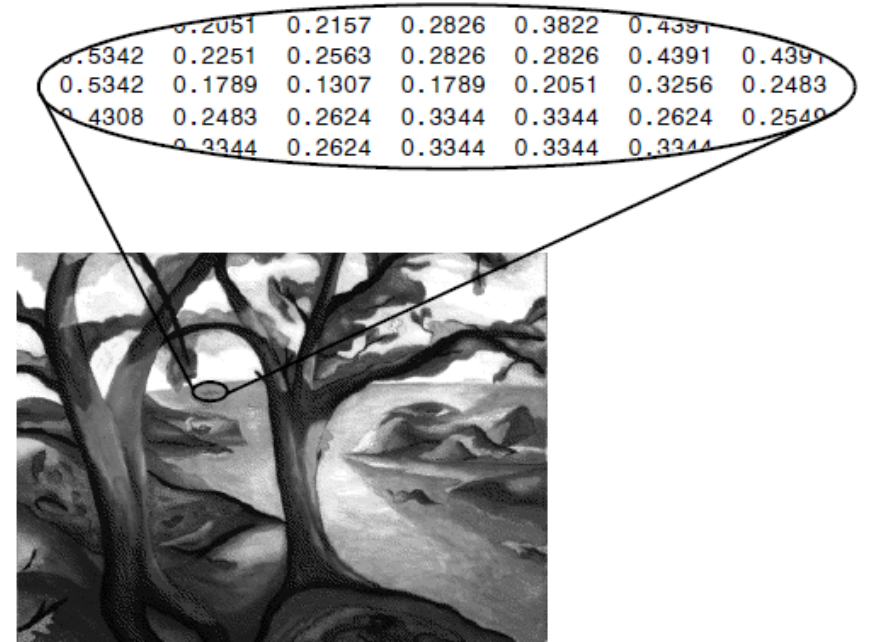
## 13.21 Reading an Image

- An image can be turned into a vector or a matrix of numbers.

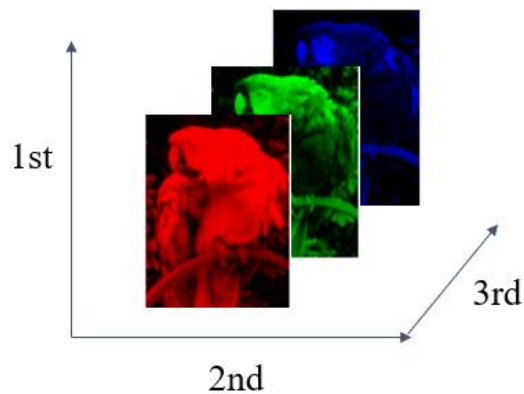
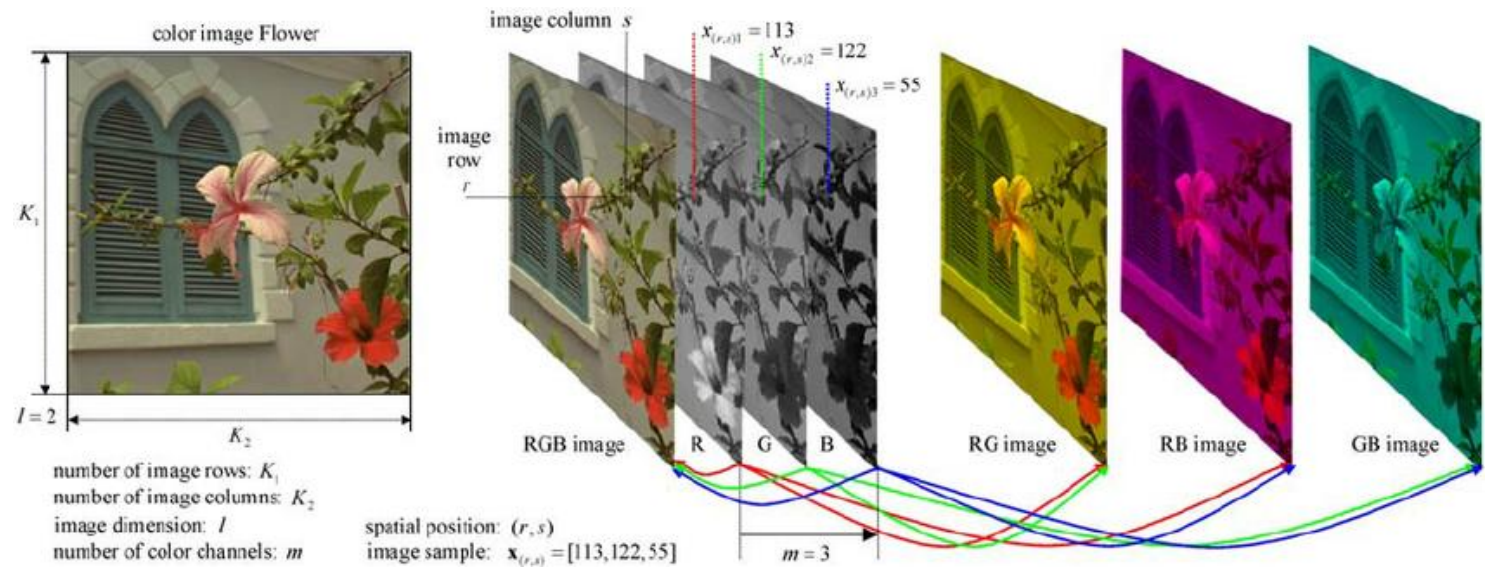
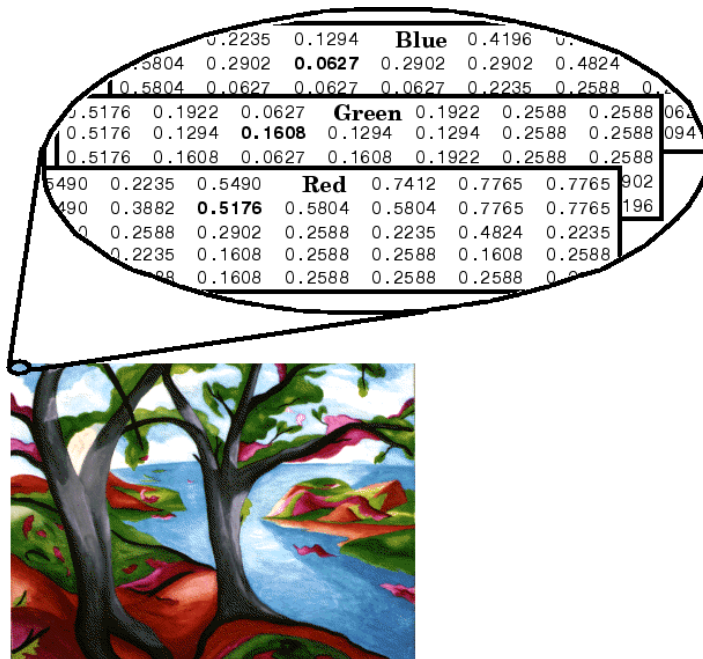


12

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



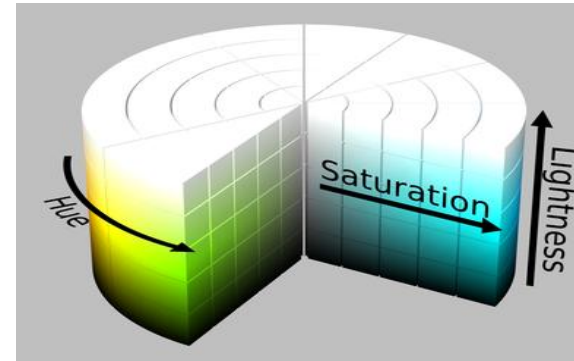
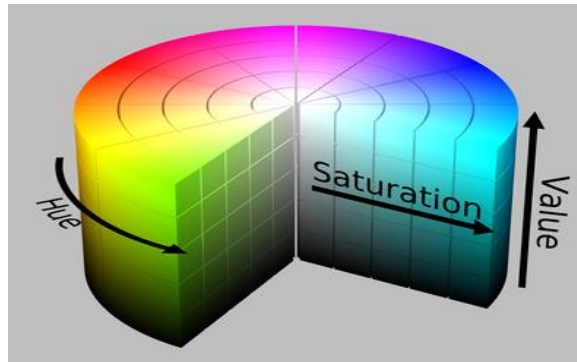
# 13.21 Reading an Image



**Note:** The computer won't "know" a channel is Red, it just knows that there are 3 intensity channels.

Different color types and their conversion

- **Grayscale (binary):** Simple 1 channel image with pixels intensity values ranging from 0 to 255.
- **RGB:** 3 channel image (RGB) with pixels intensity values ranging from 0 to 255 for each channel.
- **HSV and HLS:** Another representation where H is hue, S is saturation, V is value and L is lightness.



- **LAB color space:** Another representation where  $L$  is lightness,  $A$  for green-red colors, and  $B$  for blue-yellow colors.

- Transformation operations on image are usually referred to as geometric transformations.
- These consist of shifting (translation), rotating along axis or projecting it onto different planes etc.
- At the core of transformation is a matrix multiplication of our image.

### Image Translation

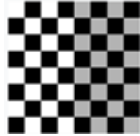

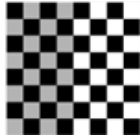
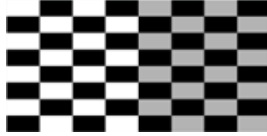


- Displacement of an image in any direction done by applying a transformation matrix to the image.
- The transformation matrix for translation only is given as:  $T = \begin{bmatrix} 0 & 1 & t_x \\ 1 & 0 & t_y \end{bmatrix}$

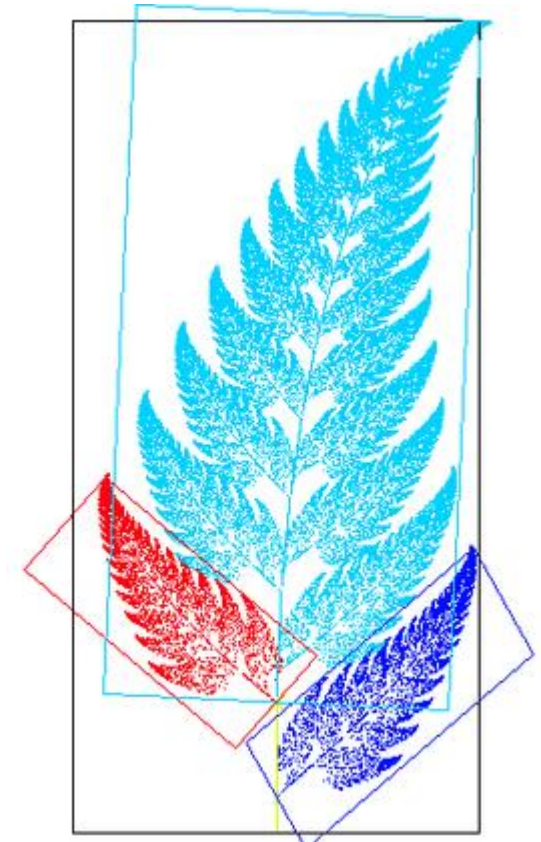
where  $t_x$  is translation in  $x$  direction and  $t_y$  in  $y$  direction in image reference.

### Image Rotation

- Similar to translation, rotating an image is also possible by creating a transformation matrix.

$$\begin{bmatrix} \alpha & \beta & (1 - \alpha).center_x - \beta.center_y \\ -\beta & \alpha & \beta.center_x - (1 - \alpha).center_y \end{bmatrix} \quad \begin{aligned} \alpha &= scale.cos\theta \\ \beta &= scale.sin\theta \end{aligned}$$

Transformation Type	Affine Matrix	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Translation	$\begin{bmatrix} 1 & 0 & v_x > 0 \\ 0 & 1 & v_y = 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Reflection	$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Scale	$\begin{bmatrix} c_x = 2 & 0 & 0 \\ 0 & c_y = 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Rotate	$\begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$	
Shear	$\begin{bmatrix} 1 & c_x = .5 & 0 \\ c_y = 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	



**Note:** various transformations applied in cascaded fashion to create combined transformed results.



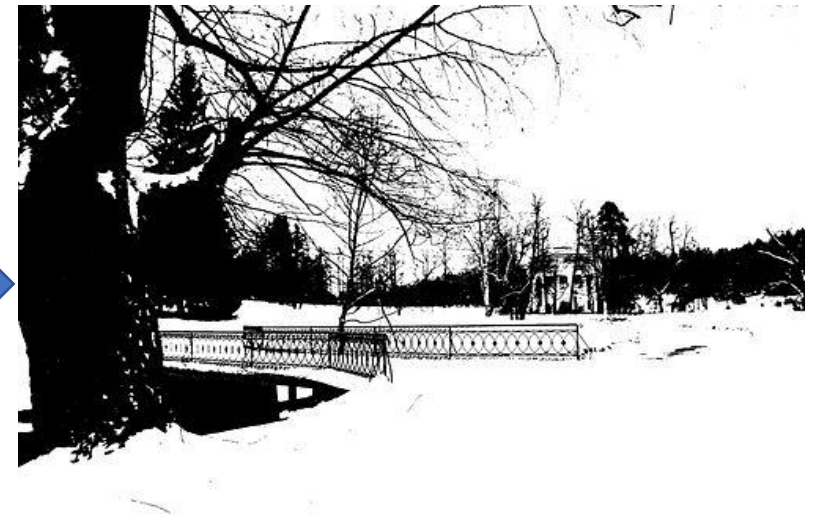
- Blending multiple images together and pasting images on top of each other.
- Blending images done through **addWeighted** function that uses both images and combines them.
- To blend images we use a simple formula:  $\text{new\_pixel} = \alpha \times \text{pixel\_1} + \beta \times \text{pixel\_2} + \gamma$



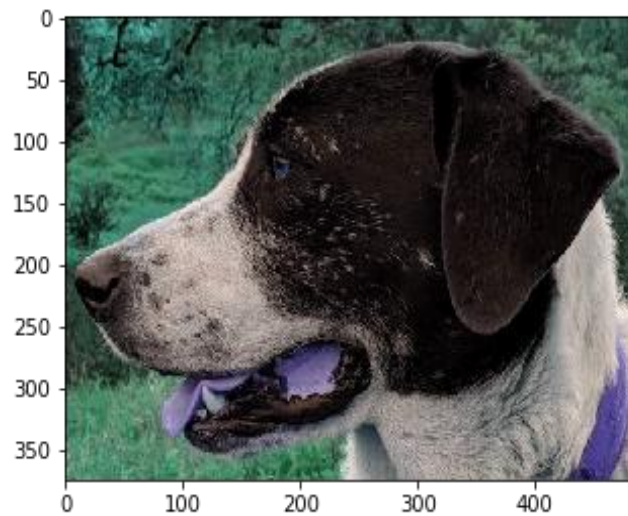
- Thresholding is a method of segmenting an image into different parts showing general shapes.
- If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black)
- Convert color images to grayscale (binary), since only edges and shapes end up being important.



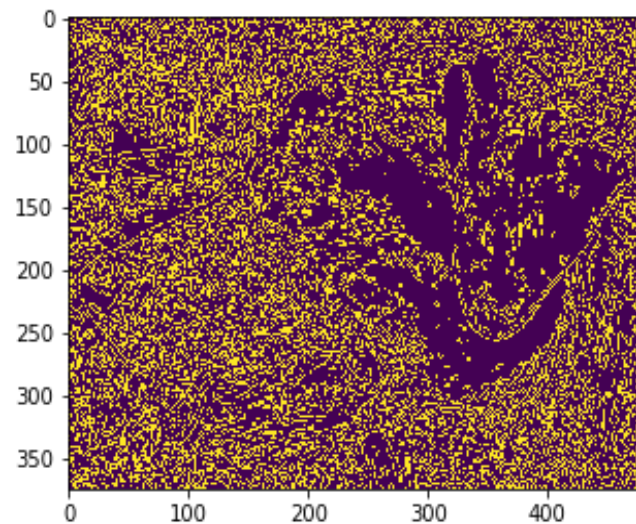
Converting a color image  
to binary



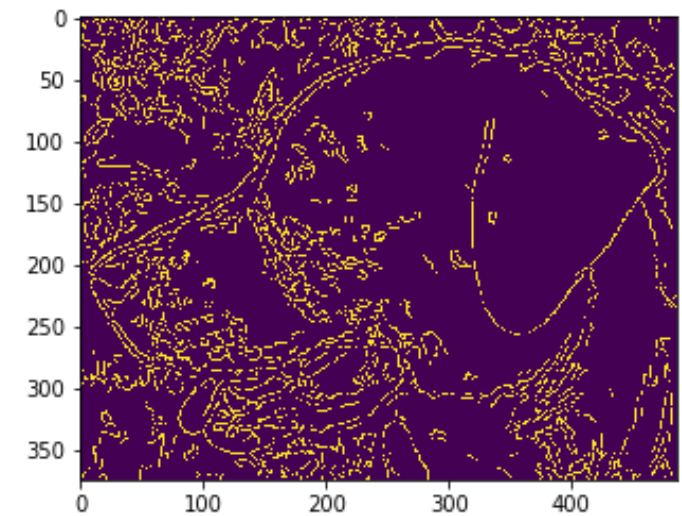
- Smoothing or blurring an image can help get rid of noise.
- Often blurring or smoothing is combined with edge detection.
- Edge detection algorithms detect too many edges on a high resolution image without any blurring.



**Original**



**Detected Edges (no blur)**



**Detected Edges (with blur)**



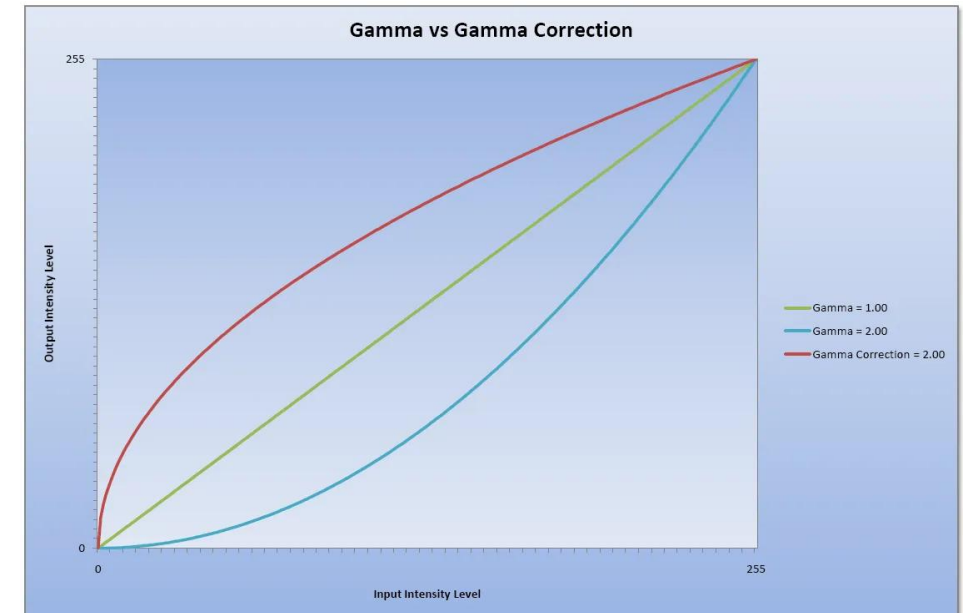
There are different methods for smoothing and blurring

- **Gamma Correction:** Applied to an image to make it appear brighter or darker depending on the Gamma value chosen.
- Note that with a gamma ( $\gamma$ ) of 1 the input equals the output producing a straight line.
- For calculating gamma correction the input value is raised to the power of the inverse of gamma.

$$I' = 255 \times \left( \frac{I}{255} \right)^{1/\gamma}$$

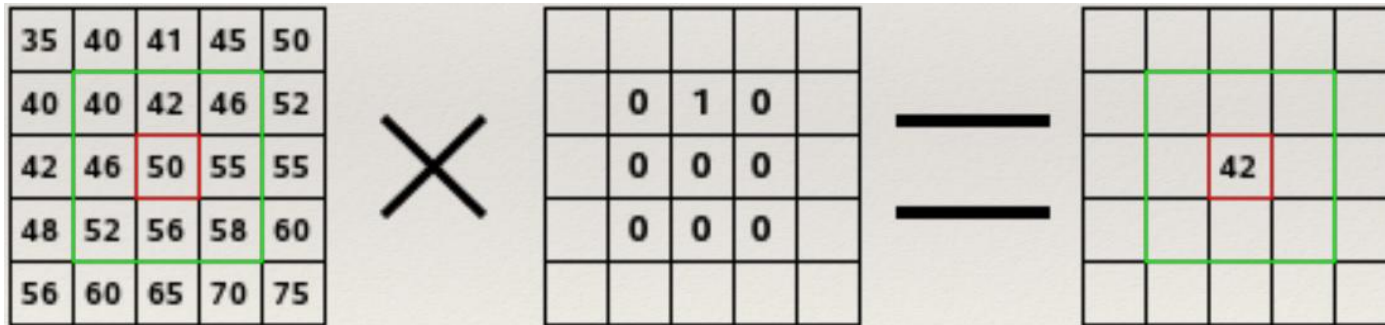


'Lena' image with gamma of 0.25 (left) and 2.00 (right)  
(click images to enlarge)



## 13.26 Blurring and Smoothing

- **Kernel Based Filters:** Kernels applied over an image to produce a variety of effects.
- Also called convolution matrix or mask
- Matrix used to convolve kernel values with image values
- Square and small (3x3, 5x5 etc.)
- The larger the matrix, the more local information is lost
- Allows for “area” effects such as blur, sharpening and edge-detection

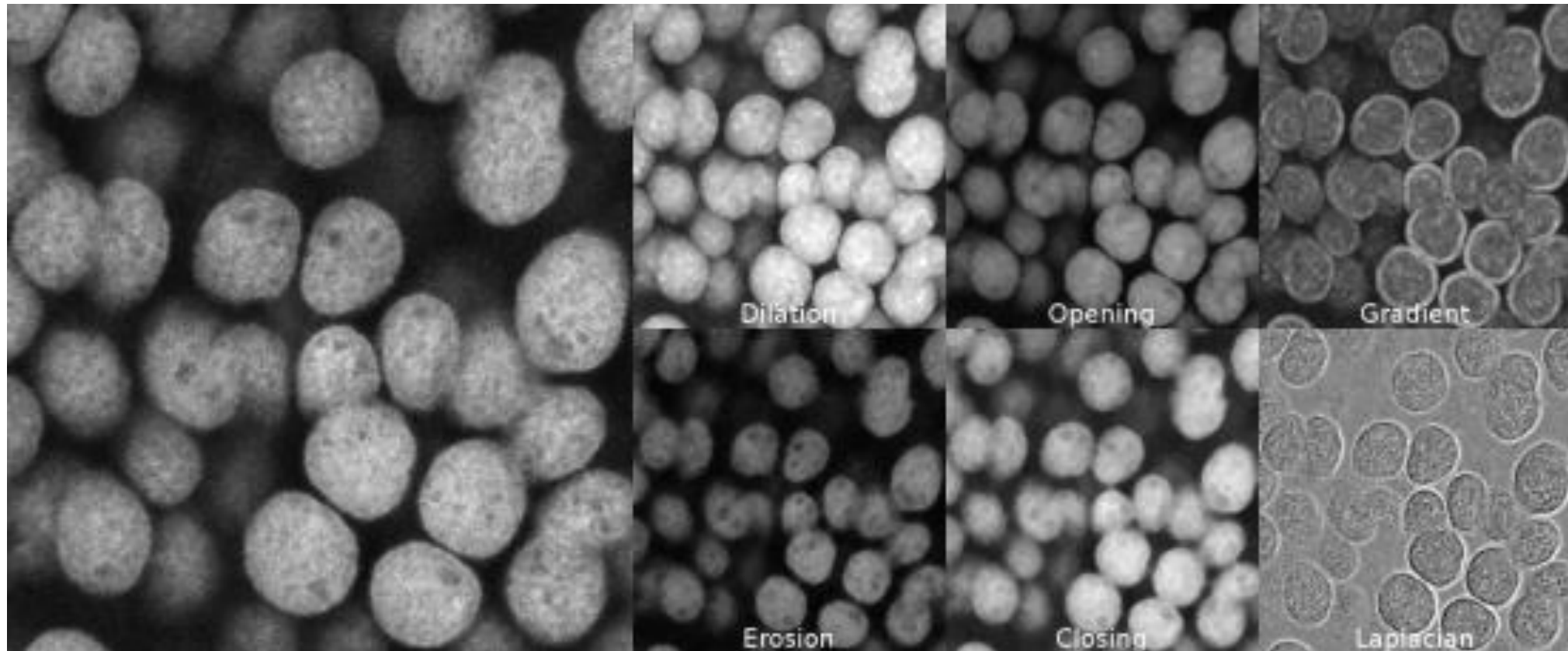


Blurred



## 13.27 Morphological Operators

- Morphological Operators are sets of kernels that achieve variety of effects, such as reducing noise.
- Certain operators are very good at reducing black points on a white background (and vice versa).
- This effect is most easily seen on text data (white text on a black background).



- Filters are operations on image to modify them for computer vision tasks.
- These perform various functions such as removing noise, extracting edges, blurring etc.

### **What is noise?**

- Taking a picture in improper environment like night or sunlight, induces lots of bright and dark areas in the image which considered as noise.
- Unwanted objects or colors in an image are also considered as noise.

### **Importance of noise**

- In several applications, noise plays an important role especially in Deep Learning based models.
- As an example, model designed for applications like image classification need to work with noisy images as well to know how robust the application is against noise becomes very important.
- Hence noise is deliberately added in the images to test the application precision.

### Linear Filters

Properties of linear filters

$$a * b = b * a$$

$$(a * b) * c = a * (b * c)$$

- **1D Linear filters (point based scalar filter):**

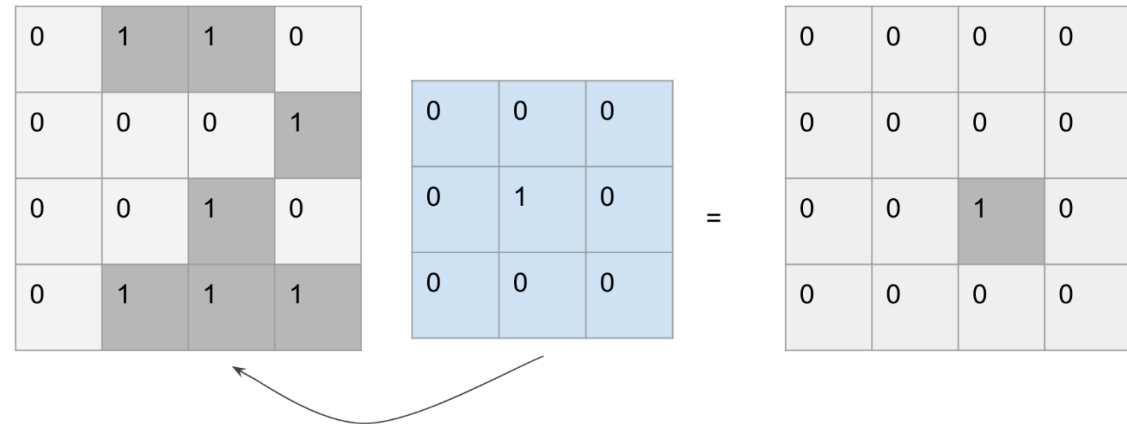
$$g(i, j) = K \times f(i, j)$$

$$g(i, j) = K \times f(i, j) + L$$

- **2D linear filters (neighbouring filter):**

- **Box filters:**

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



- Applying this filter results in blurring the image.
- In frequency domain analysis of the image, this filter is a low pass filter.
- The frequency domain analysis is done using Fourier transformation of the image.

### Non-linear filters

- **Gaussian filter** 
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right)$$

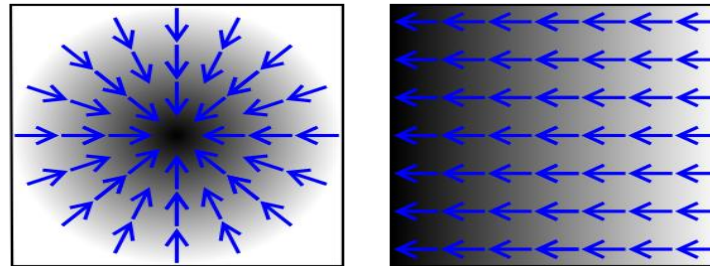
- Enhances the effect of center pixel and gradually reduces the effects further from center.
- Removes high-frequency components which results in removing strong edges and hence a blurred photo (performs better blurring than a box filter)

- **Median filter**

- Uses same technique of neighborhood filtering but use of a median value of the region.
- This results in the removal of random peak values in the region, which can be due to noise.
- Application is in smartphone application which filters input image and adds additional artifacts to add artistic effects.



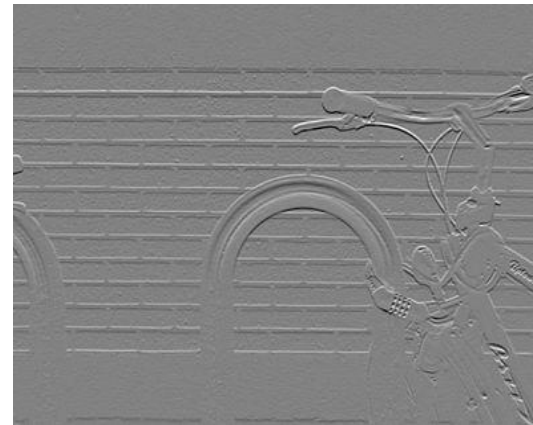
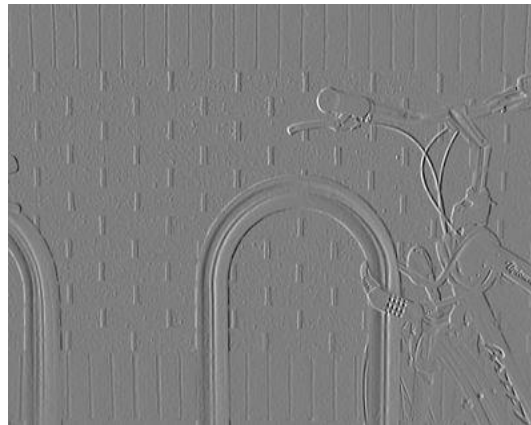
- An image gradient is a directional change in the intensity or color in an image.
- These are more edge detectors or sharp changes in a photograph.
- Image gradients widely used in object detection and segmentation tasks.
- This is also termed as an image derivative with respect to given direction (here X or Y).



- Sobel-Feldman Operators is a popular gradient operator often used for general edge detection.
- The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical.

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

- Gradients can be calculated in a specific direction.
- Normalized  $x$ -gradient from Sobel-Feldman Operator.
- Normalized  $y$ -gradient from Sobel-Feldman Operator.
- Normalized gradient magnitude for general edge detection from Sobel–Feldman operator.



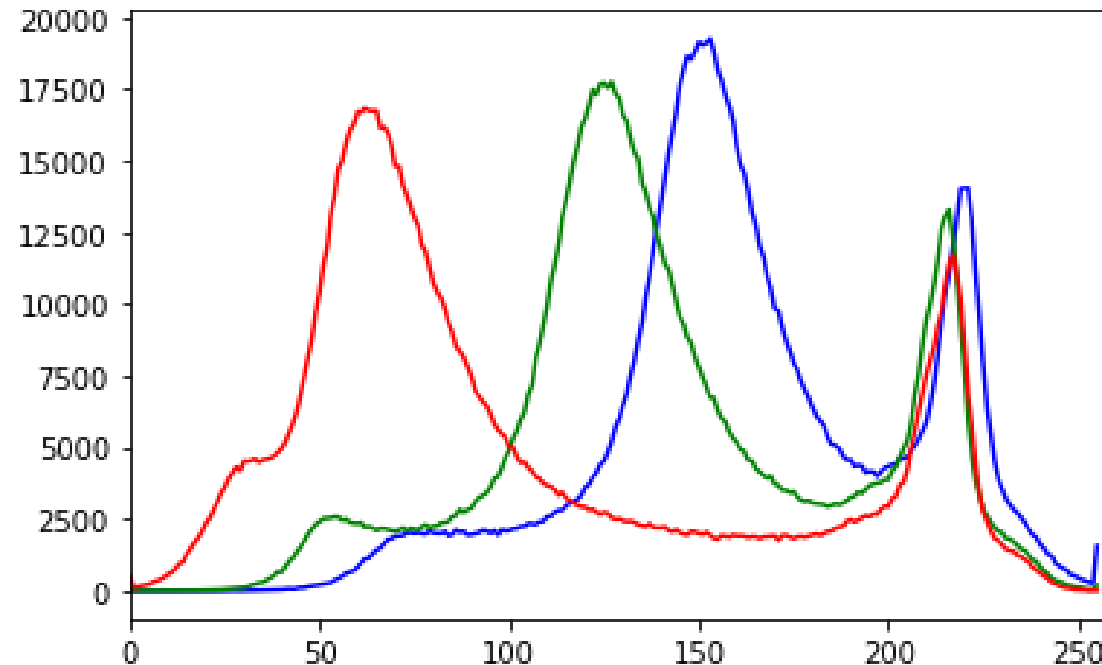
$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$



- A histogram is a visual representation of the distribution of a continuous feature.

### Example:

- For images, the frequency of values for each of the three RGB channels between 0-255.
- We can plot these as 3 histograms on top of each other to see how much of each channel there is.

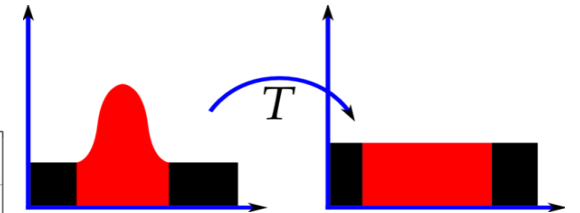
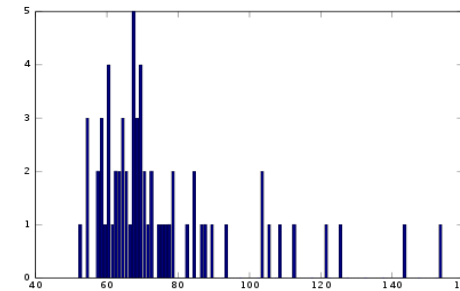
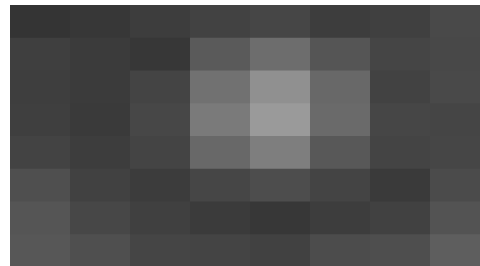


## Histogram Equalization

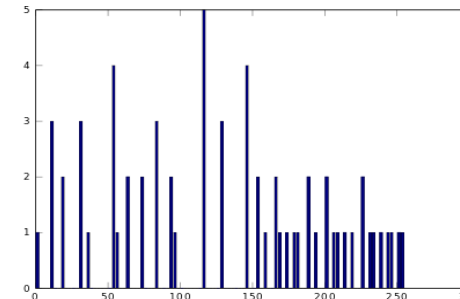
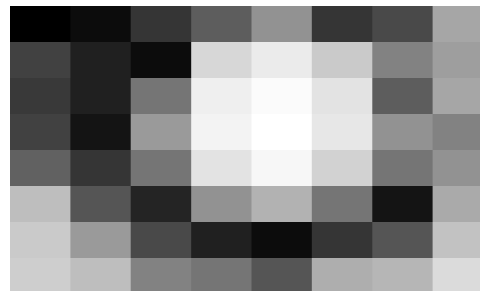
- A method of contrast adjustment (change brightness and contrast) based on image's histogram.
- Intuitively, this method tries to set the brightest pixels to white and the darker pixels to black.

### Example

52	55	61	59	70	61	76	61
62	59	55	104	94	85	59	71
63	65	66	113	144	104	63	72
64	70	70	126	154	109	71	69
67	73	68	106	122	88	68	68
68	79	60	79	77	66	58	75
69	85	64	58	55	61	65	83
70	87	69	68	65	73	78	90



52	55	61	59	70	61	76	61
62	59	55	104	94	85	59	71
63	65	66	113	144	104	63	72
64	70	70	126	154	109	71	69
67	73	68	106	122	88	68	68
68	79	60	79	77	66	58	75
69	85	64	58	55	61	65	83
70	87	69	68	65	73	78	90

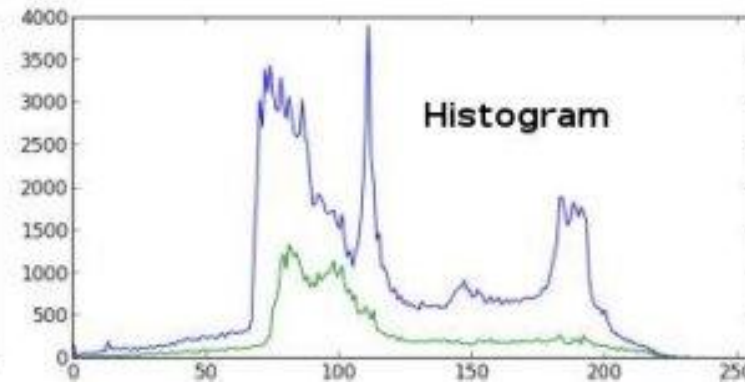
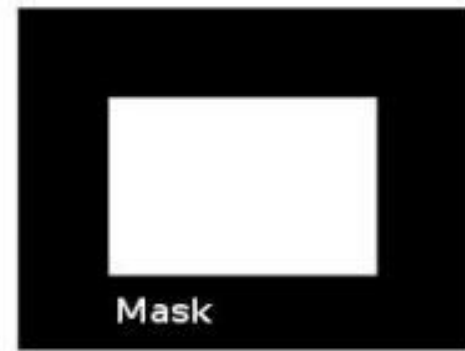


0	12	53	32	146	53	174	53
57	32	12	227	219	202	32	154
65	85	93	239	251	227	65	158
73	146	146	247	255	235	154	130
97	166	117	231	243	210	117	117
117	190	36	190	178	93	20	170
130	202	73	20	12	53	85	194
146	206	130	117	85	166	182	215

- Let's take a look at the original min and max values and apply histogram equalization
- Notice how min and max values equalized to be 0 and 255 and less shades of gray.

### Histograms on a masked portion of the image

- Select an ROI and only calculate the color histogram of that masked section.



## **The Topics Covered In This Section**

13.3.1 Features and Its Importance

13.3.2 Harris Corner Detection

13.3.3 FAST Feature Detection

13.3.4 ORB Feature Detection

13.3.5 Black Box Feature

13.3.6 SIFT

**Features:** Properties like corners, edges, texture, color, surface, shape and so on that are used to find similarities between images.

### **Importance**

- Image matching
- Object detection
- Object tracking etc.

### **Different types of features extractors**

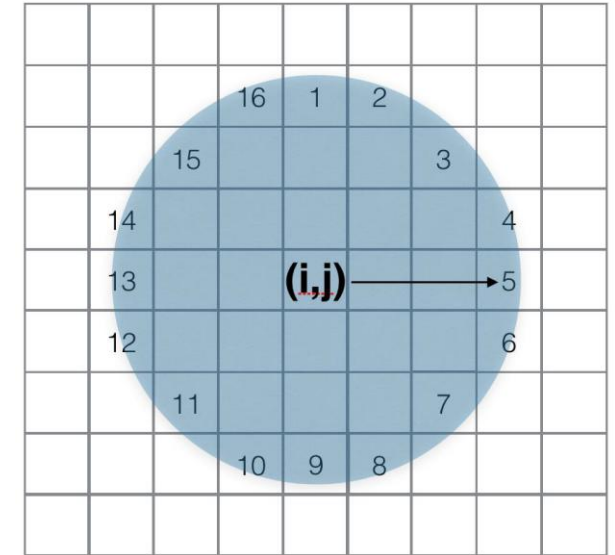
- Harris corner detector
- FAST keypoint detectors
- *SIFT*
- *SURF*
- *BRIEF*
- ORB features detectors
- Black box features

- Overlay chosen window on input image and observe only the overlayed region from input image.
- This window is later shifted over the image and the new overlayed region is observed.
- In this process, there arise three different cases:
  - i. If there is a flat surface, there will not be any change in the window region irrespective of the direction of movement of the window.
  - ii. If there is edge in the image and the window is overlayed on it and shifted along the direction of the edge, there will not be any changes in the window. Otherwise (shifted in any other direction), there will be changes in the window region.
  - iii. If there a corner in the image and the window is overlayed on it and shifted then there will be changes in the window region.
- Harris Corner Detection uses this property in terms of a score function
$$E[u, v] = \sum_{x,y} w(x, y)[I(x + u)(y + v) + I(x, y)]$$
- Where  $w$  is a window,  $u$  and  $v$  are the shift and  $I$  is image pixel value. The output  $E$  is the objective function and maximizing this with respect to  $u$  and  $v$  results in corner pixels in the image  $I$ .
- The score value will show whether there is an edge, corner, or flat surface.

- Features from Accelerated Segment Test introduced by Edward Rosten & Tom Drummond [ 2006].

- A candidate pixel  $(i,j)$  is selected with an intensity  $I(i,j)$ :

- In a circle of 16 pixels, given a threshold  $t$ , estimate  $n$  adjoining points which are brighter or darker than pixel  $(i,j)$  intensity by a threshold  $t$ .
- That means these  $n$  pixels are either  $< (I(i,j) + t)$  or  $> (I(i,j) - t)$
- In a high-speed test, only four pixels are looked at.
- The intensity value of at least three pixels of these decides whether the center pixel  $p$  is a corner.
- If these values are either  $> (I(i,j) + t)$  or  $< (I(i,j) - t)$  then the center pixel is considered a corner.

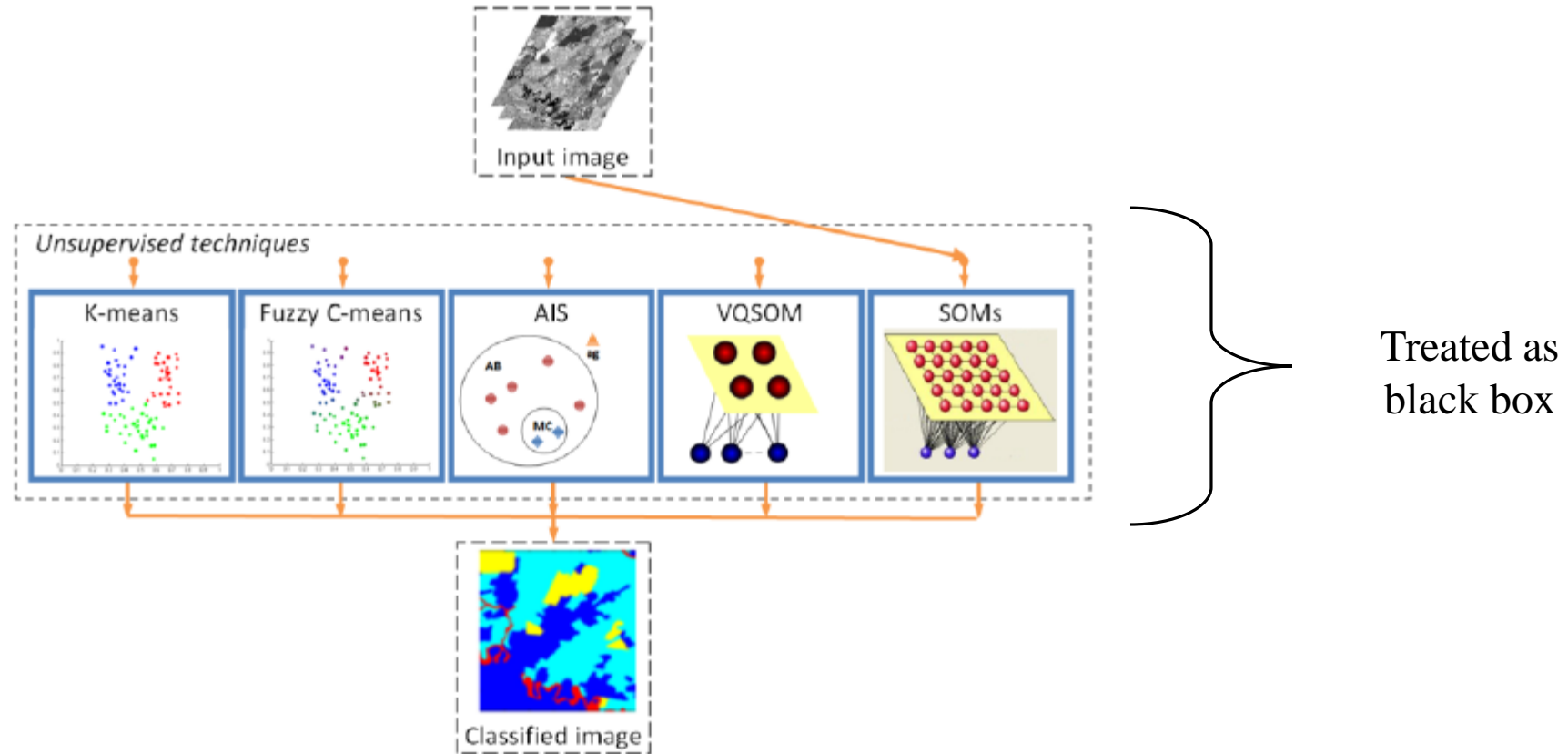


- Many features detectors are not useful for real-time applications such as a robot with a camera is moving on the streets.
- Any delay caused may decrease the functionality of the robot or complete system failure.
- Features detection is not the only part of the robot system but if this effects the runtime, it can cause significant overhead on other tasks to make it work real time.

- Oriented FAST and Rotated BRIEF introduced by Ethan Rublee et al. [2011]
- This combines two algorithms:
  - FAST feature detector with an orientation component
  - BRIEF Descriptors
- Harris corner detectors are fast to compute, however in matching two images, it is difficult to select which two image corners are matched for corresponding pixels.
- The advantage is the speed of detections while maintaining robust detections.
- This makes useful for real-time applications like robotics system, smartphone apps, and so on.



- Black box feature detection is a generally used for classification of images
- Maximum likelihood approach is used as black box feature detection and

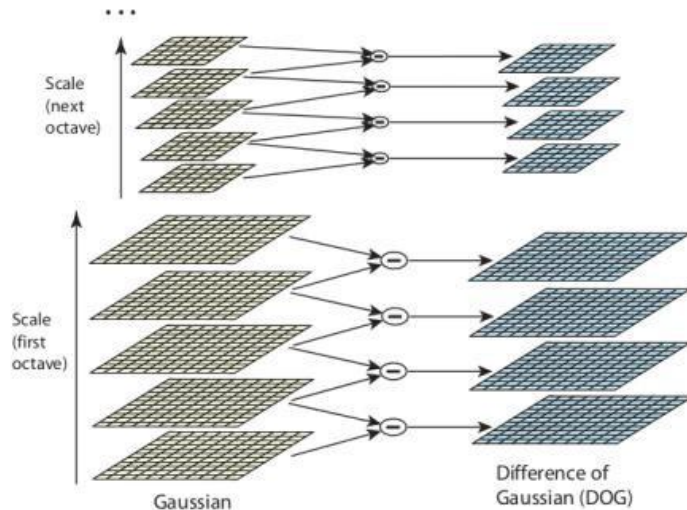


- Scale-Invariant Feature Transform (SIFT) algorithm is used to detect and describe logical features in an image
- The algorithm locates certain key points and maps them with quantitative information (descriptors) which can be used for object detection
- These descriptors are invariant to the different transformations of image depicting the same objects every time.
- The SIFT algorithm is implemented in four steps:
  - **Scale-space peak selection**
  - **Keypoint Localization**
  - **Orientation Assignment**
  - **Keypoint descriptor**
  - **Keypoint Matching**
- ***Scale-space peak selection:*** The scale space is a function  $f(x, y, \sigma)$  that is produced from convolution of a Gaussian kernel

$$L(x, y, \sigma) = G(x, y, \sigma) \times I(x, y); \quad G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

where  $G$  is the Gaussian Blur and  $I$  is the image and  $x, y$  are the coordinates in  $I$  which needs to be scaled with a factor  $\sigma$

- All the blurred images are then used to for difference of Gaussian (DOG) kernel to identify the key points



- Key Point localization:** Using DOG method a large number of key points are identified of which all may not be important
- Key point localization helps to get rid of unwanted points using Taylor series expansion to identify the maxima's and remove all the points which are beyond a threshold

- Reject flats:

$$\square \quad |D(\hat{\mathbf{x}})| < 0.03$$

- Reject edges:

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \quad \text{Let } \alpha \text{ be the eigenvalue with larger magnitude and } \beta \text{ the smaller.}$$

$$\text{Tr}(\mathbf{H}) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\text{Det}(\mathbf{H}) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta.$$

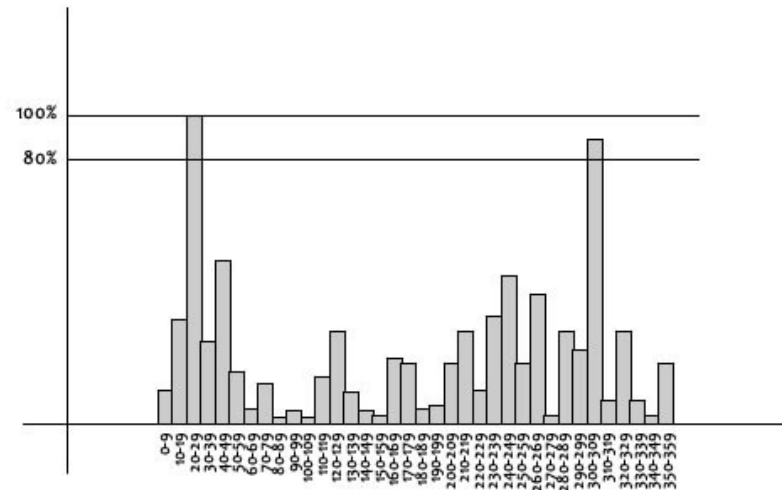
$$\text{Let } r = \alpha/\beta. \\ \text{So } \alpha = r\beta$$

$$\frac{\text{Tr}(\mathbf{H})^2}{\text{Det}(\mathbf{H})} = \frac{(\alpha + \beta)^2}{\alpha\beta} = \frac{(r\beta + \beta)^2}{r\beta^2} = \frac{(r + 1)^2}{r},$$

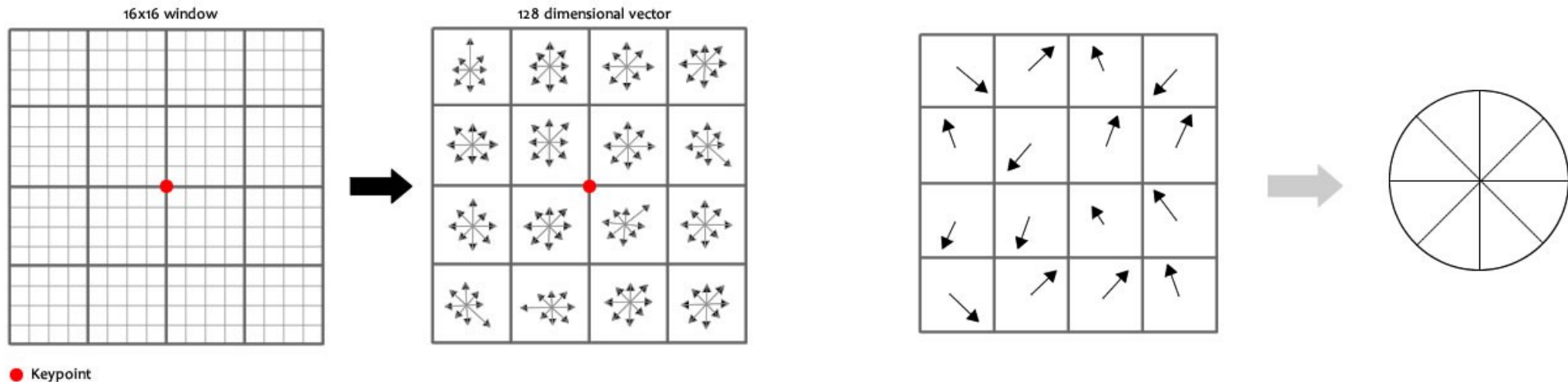
$(r+1)^2/r$  is at a min when the 2 eigenvalues are equal.

$$\square \quad r < 10$$

- **Orientation Assignment:** In order to make the rotation invariance, an orientation is assigned to each key point
- A neighbourhood is taken and 360 degree histogram with 36 bin is created
- If a gradient direction is 15 degree then it go to the 10-19 degree bin and the amount added to the bin is the magnitude of the gradient.
- Once the process is done for all the pixels in the neighbourhood of the key point there will be some peak in the histogram
- The highest peak in the histogram is taken and any peak above 80% of it is also considered to calculate the orientation.
- It creates key points with same location and scale, but different directions contributing to the stability of matching.



- **Key point descriptor:** As each key point has location, scale and orientation a highly indistinctive descriptor for each key point needs to be computed
- A 16 x 16 window around the key point with all possible orientation is taken where every block is again divided to 4 x 4 size
- Now for each sub block, 8 orientation histogram is created resulting into 128 bin values.
- Clearly if the image is rotated, all gradients will also change so for rotation independence, key point's rotation is subtracted from each orientation

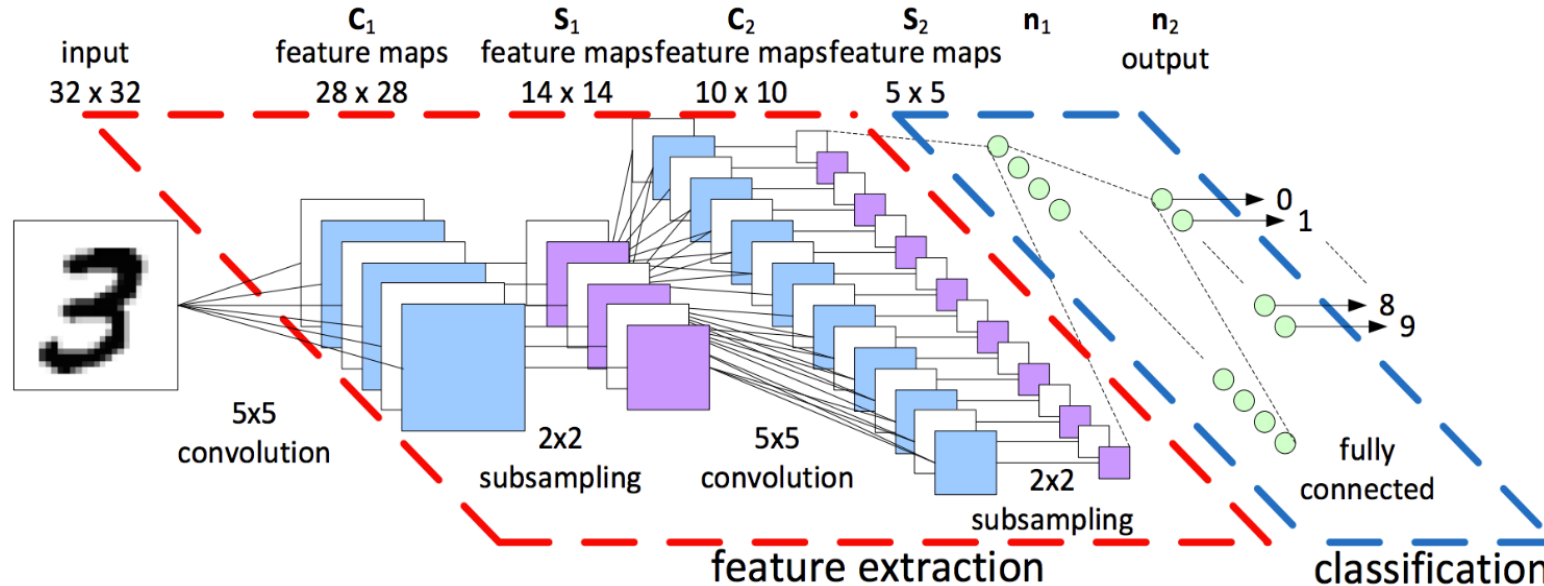


- **Key point matching:** Key points between two images are matched by identifying their nearest neighbors.

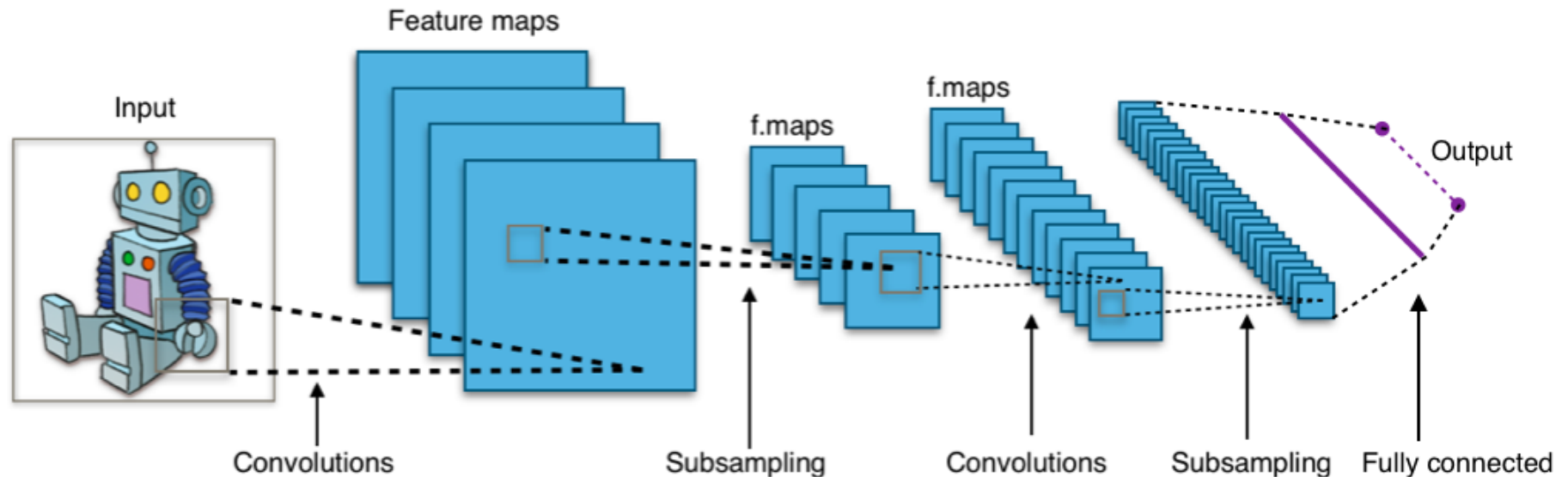
## **The Topics Covered in This Section**

- 13.4.1 Deep Learning Models
- 13.4.2 CNN Topology
- 13.4.3 Optimization
- 13.4.4 Choosing Hyperparameters
- 13.4.5 Regularization
- 13.4.6 Data Augmentation
- 13.4.7 Transfer learning
- 13.4.8 Exercises

- The better the features are, the more accurate the results are going to be.
- In recent periods, the features have become more precise and better accuracy has been achieved.
- This is due to a new kind of feature extractor called **Convolutional Neural Networks (CNNs)**.
- CNN shown remarkable accuracy in complex tasks such as object detection and classifying images with high accuracy.
- Applications ranging from smartphone photo enhancements to satellite image analysis.



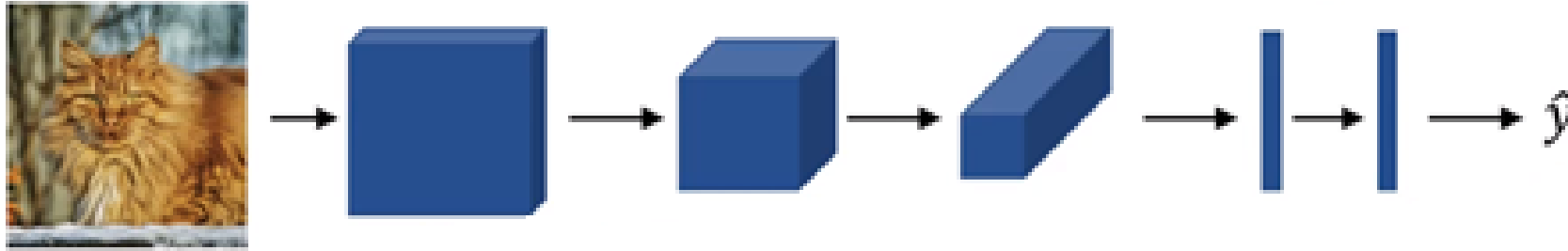
- Convolution or Feature Extraction Layer (+ ReLU )
- Pooling or Subsampling Layer (max or average Pooling)
- Fully Connected + ReLU
- Softmax



**Note:** CNN hidden layers are basically alternation of convolution and pooling layers.

**Note:** Though the layers are colloquially referred to as convolutions, this is only by convention. Mathematically, it is technically a *sliding dot product* or cross-correlation.





- Training set  $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$
- Cost Function,  $J = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$
- Use Gradient Descent to optimize parameters to reduce cost J

### Filter Size

- Common filter shapes found in literature vary greatly, and are usually chosen based on the dataset.
- The challenge is thus to find right filter shape, given a particular dataset without overfitting.

### Number of Filters

- Since feature map size decreases with depth, lower layers can have fewer filters while higher layers can have more.
- The number of feature maps depends on the number of available examples and task complexity.

### Pooling Size

- Typical values are  $2 \times 2$ .
- Very large input volumes may need  $4 \times 4$  pooling in the lower layers.
- However, choosing larger shapes will dramatically reduce the dimension of the signal and may result in excess information loss.

### **Empirical**

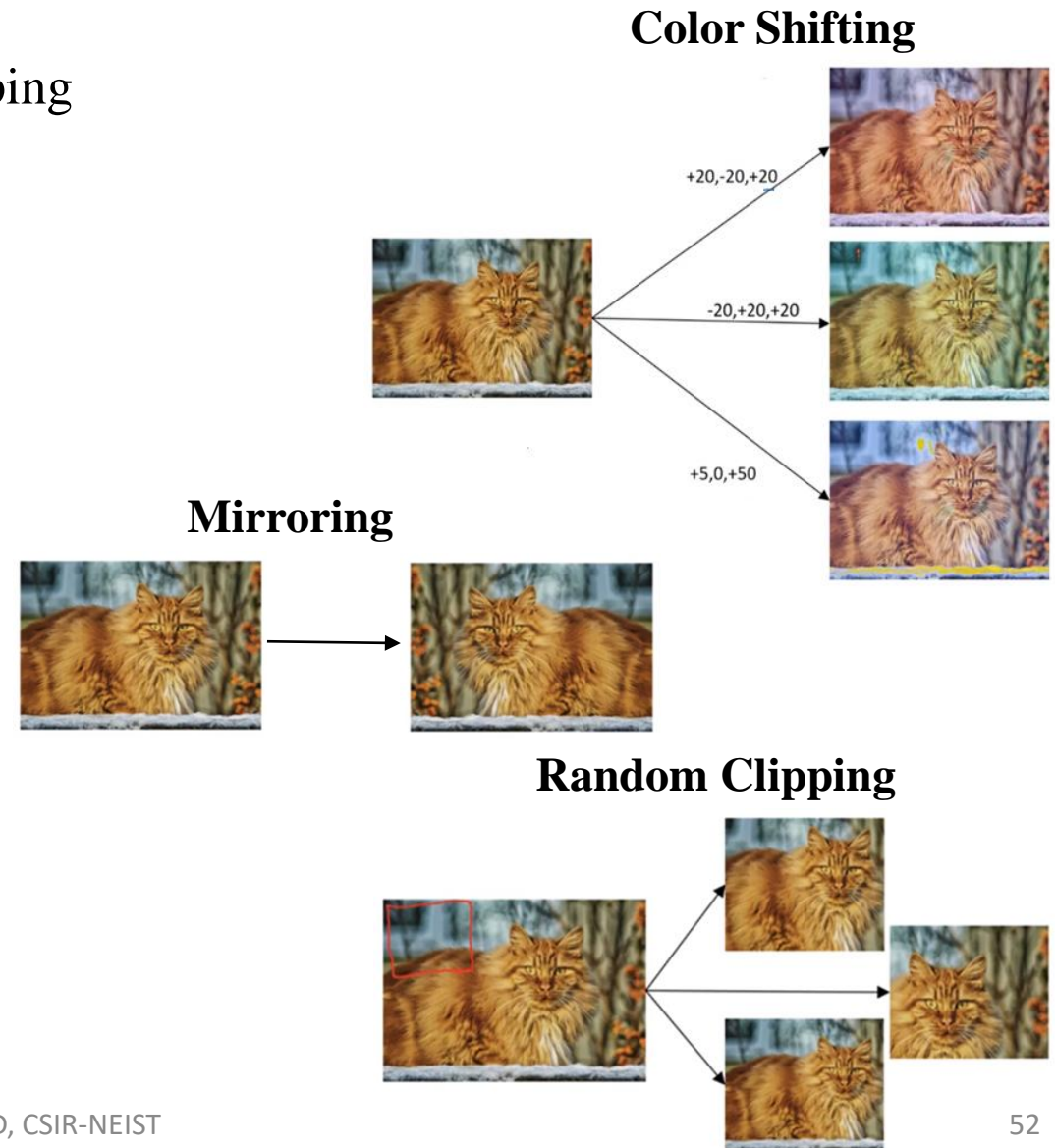
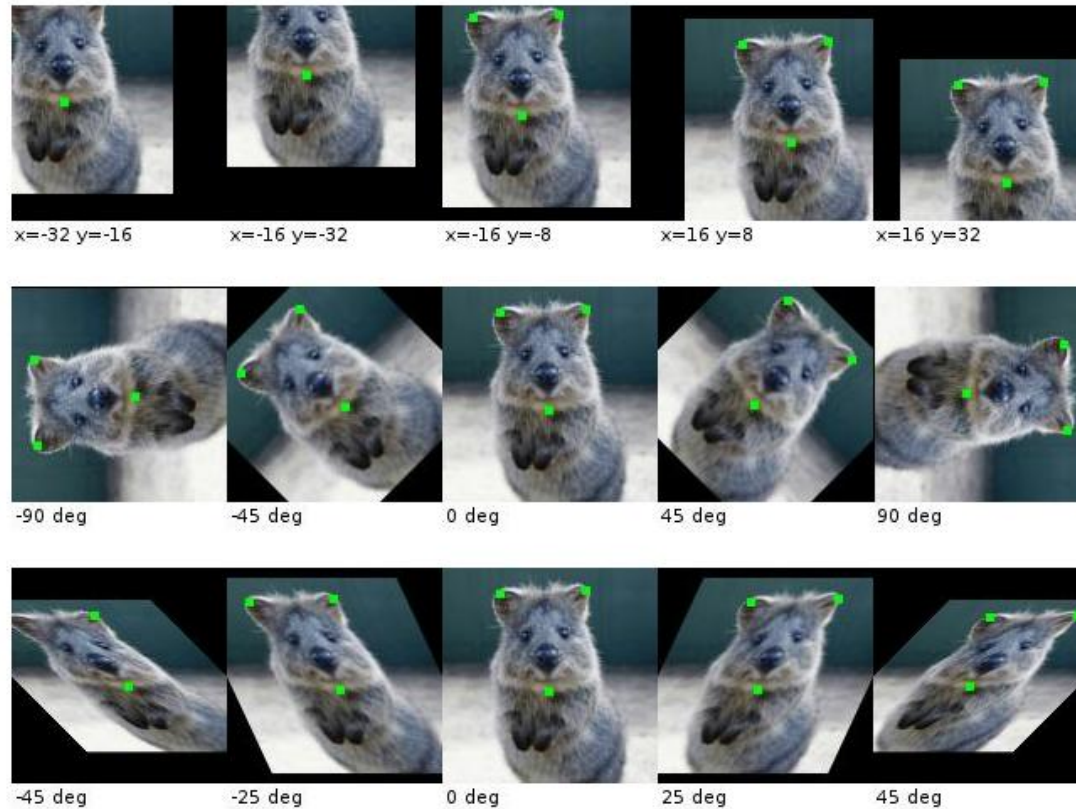
- Dropout
- DropConnect
- Hand Engineering

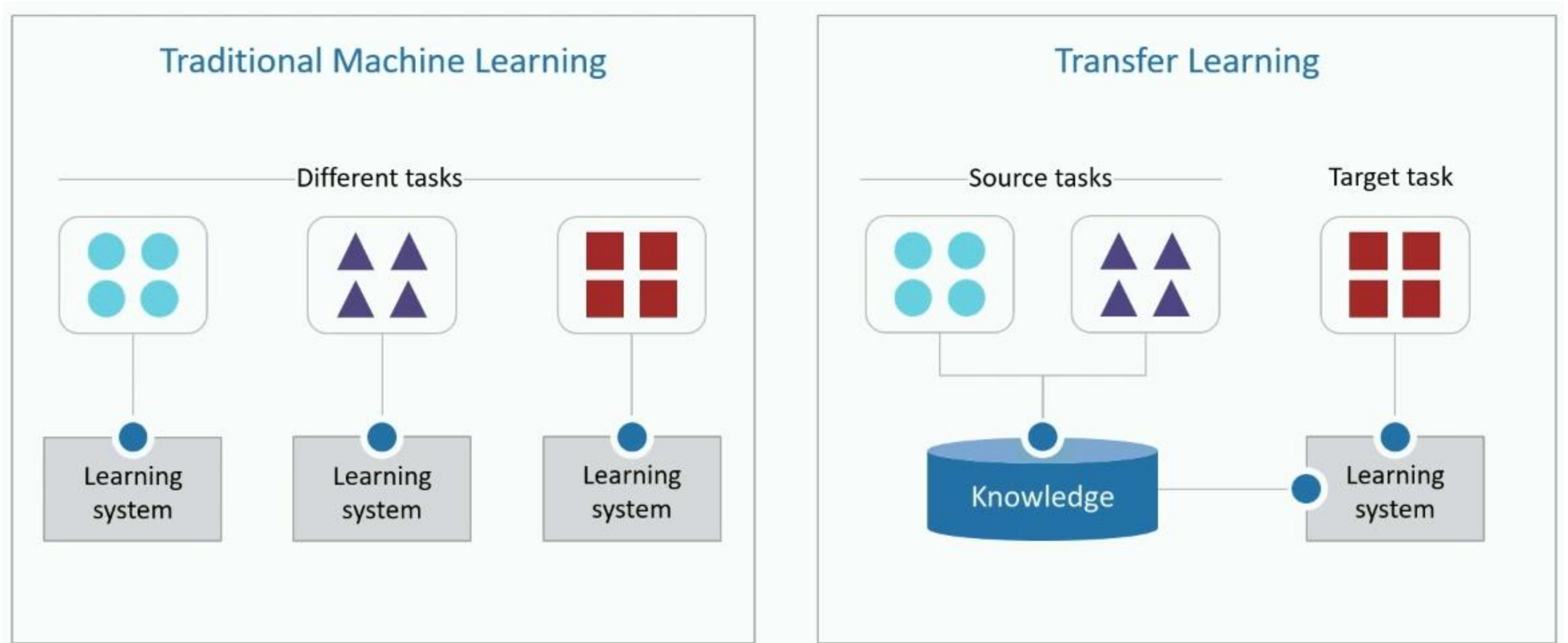
### **Explicit**

- Early Stopping
- Number of Parameters
- Weight Decay

## 13.4.6 Data Augmentation

- Affine Transformation
- Others - Color Shifting, Mirroring, Random Clipping





## **The Topics Covered in This Section**

- 13.5.1 Mechanism
- 13.5.2 Corner Detection
- 13.5.3 Edge Detection
- 13.5.4 Grid Detection
- 13.5.5 Contour Detection
- 13.5.6 Template Matching
- 13.5.7 Feature Matching
- 13.5.8 Yolo
- 13.5.9 Face Detection
- 13.5.10 Exercises

### **Meaning of Object Detection**

- Object detection is a problem of both localizing the position of an object in an image.
- Basically it tells “Where is that object”

### **Involves Detecting**

- Corner Detection
- Edge Detection
- Grid Detection
- Contour Detection
- Feature Matching
- Facial and Eye Detection



### **What is a corner?**

- A corner is a point whose local neighborhood stands in two dominant and different edge directions.
- A corner can be interpreted as the junction of two edges, where an edge is a sudden change in image brightness.

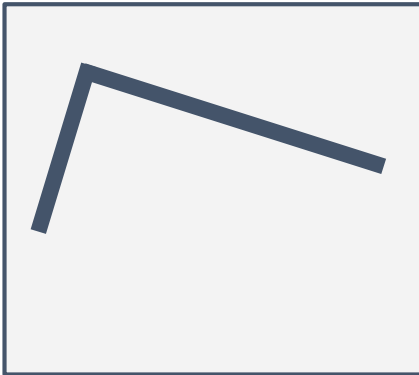
### **Some of the popular corner detection algorithms**

- Harris Corner Detection
- Shi-Tomasi Corner Detection

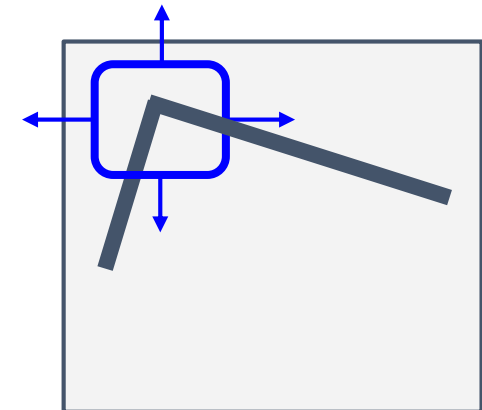
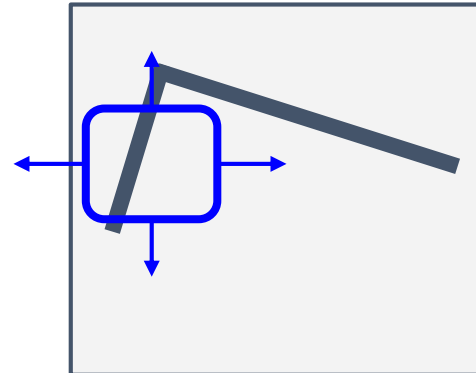
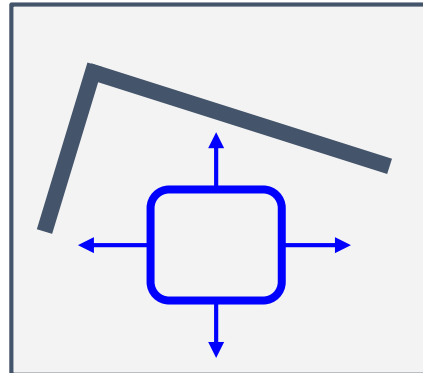
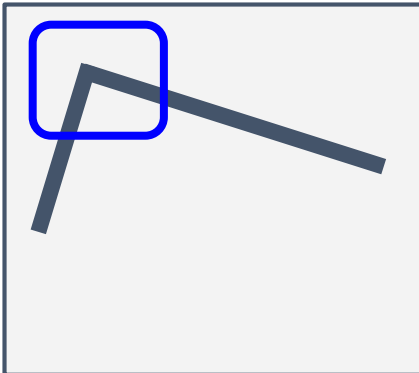
### Harris Corner Detection [Published by Chris Harris and Mike Stephens in 1998]

- The basic intuition is that corners can be detected by looking significant change in all directions.

#### Example



- Shifting this window in any direction would result in a large change in appearance.
- Flat regions will have no change in all directions.
- Edges won't have a major change along the direction of the edge.
- Edges will have a major change along the direction of the edge.



**Shi-Tomasi Corner Detection** [Published by J. Shi and C. Tomasi *Good Features to Track* in 1994]

- It made a small modification to the Harris Corner Detector which ended up with better results.
- It changes the scoring function selection criteria that Harris uses for corner detection.
  - Harris Corner Detector uses:  $\mathbf{R} = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)$
  - Shi-Tomasi uses:  $\mathbf{R} = \min(\lambda_1, \lambda_2)$

- The most popular edge detection algorithm is Canny edge detector.
- It was developed in 1986 by John Canny and is a multi-stage algorithm.

### **Canny Edge Detection**

- Apply Gaussian filter to smooth the image in order to remove the noise
- Find the intensity gradients of the image
- Apply non-maximum suppression to get rid of spurious response to edge detection
- Apply double threshold to determine potential edges
- Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

### **Notes**

- For high resolution images where you only want general edges, it is usually a good idea to apply a custom blur before applying the Canny Algorithm.
- The Canny Algorithm also requires a user to decide on low and high threshold values.

- Often cameras create a distortion in an image, such as radial distortion and tangential distortion.
- A good way to account for these distortions when performing operations like object tracking is to have a recognizable pattern attached to the object being tracked.
- Grid patterns are often use to calibrate cameras and track motion.

- Contours are defined as **a curve joining all the continuous points** (along the boundary), having same color or intensity.
- Contours are a useful tool for shape analysis and object detection and recognition.
- Allows us to detect foreground vs background images.
- Allows detection of external vs internal contours (e.g. grabbing eyes and smile from a smile face).

Note: OpenCV has a built in Counter finder function that can also helps us differentiate between internal and external contours.

- Template matching is the simplest form of object detection.
- Simply looking for an exact copy of an image in another image.
- It simply scans a larger image for a provided template by sliding the template target image across the larger image.



### What is Feature Matching?

- Detecting matching objects in another image, even if the target image is not shown exactly the same in the image we are searching.

### Feature matching vs. Template matching

- Template matching also finds object within a larger image, but it requires exact copy of the image.
- Often that isn't useful in a real world situation!

### How feature matching works

- Feature matching extracts defining key features from an input image.
- Then using a distance calculation, finds all the matches in a secondary image.
- This means we are no longer required to have an exact copy of the target image!
- Notice how the input image is not exactly what is shown in the secondary image!

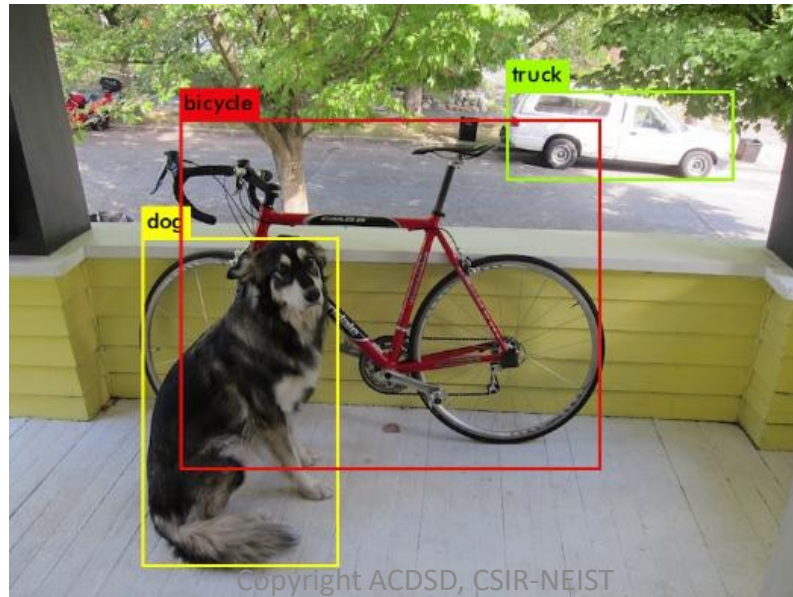


### Feature matching methods

- Brute-Force Matching with ORB Descriptors
- Brute-Force Matching with SIFT Descriptors and Ratio Test
- FLANN based Matcher

### YOLO (You Only Look Once) Object Detection Algorithm

- Prior detection systems apply the model to an image at multiple locations and scales.
- High scoring regions of the image are considered detections.
- YOLO applies a single neural network to the full image.
- It divides the image into regions and predicts bounding boxes and probabilities for each region.
- These bounding boxes are weighted by the predicted probabilities.

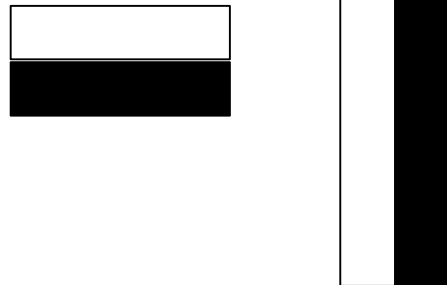


### Face detection from Viola-Jones object detection framework using Haar Cascades

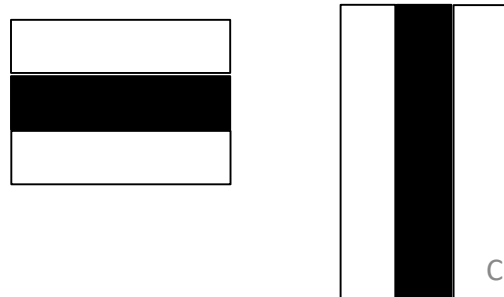
- In 2001 Paul Viola and Michael Jones published their method of face detection based on the simple concept of a few key features.
- We will be able to very quickly detect if a face is in an image and locate it.
- However we won't know who's face it belongs to.

### The main feature types that Viola and Jones proposed

- Edge Features

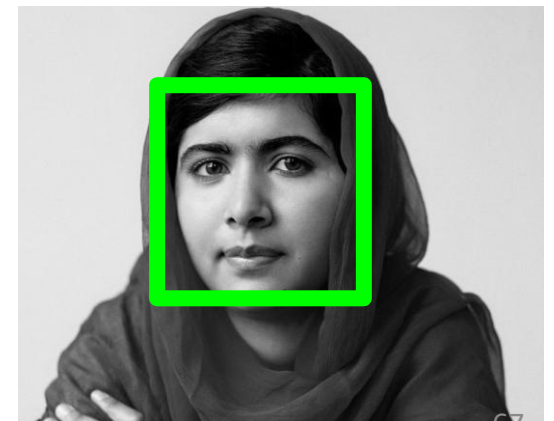


- Line Features



### Example

- Convert the front facing person's face image into grayscale.
- We will use Haar Cascades features to detect faces in images.
  - First features searched for is an edge feature indicating eyes and cheeks.
  - If it passed, then we search for the next feature, such as the bridge of the nose.
  - We continue through this cascade (which can be thousands of features).
  - Until the algorithm detects the face.
- If we fail for the search of this feature, we can say there is no face.



## **The Topics Covered in This Section**

13.6.1 Introduction

13.6.2 Graph Based Segmentation

13.6.3 Watershed Algorithm

13.6.4 Deep Learning for Segmentation

13.6.5 Exercises

- Object detection methods creates a bounding box around the target object.
- Some applications requires a precise boundary called object instance detection.

### What is segmentation?

- Segmentation performs object instance detection.
- Segmentation is often referred to as the clustering of pixels of a similar category.

### Challenges in segmentation

- **Noisy boundaries:** Grouping pixels that belong to a category may not be as accurate due to the fuzzy edges of an object. As a result, objects from different categories are clustered together.
- **Cluttered scene:** With several objects in the image frame, it becomes harder to classify pixels correctly. With more clutter, the chances of false positive classification also increase.



### **Methods of segmentation**

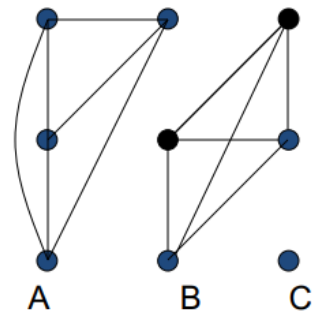
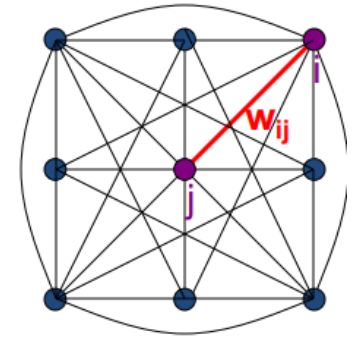
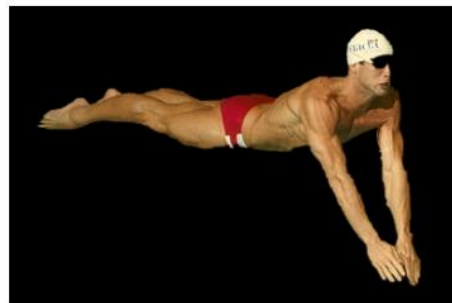
- Traditional methods
- Deep learning methods

### **Differences**

- In traditional segmentation techniques, the key property used is image intensity levels.
- Different smaller regions of similar intensity values are found, and later merged into larger regions.
- To get the best performance, an initial point is chosen by the user for algorithms.
- Recent deep learning approaches shown better performance without the need for initialization.

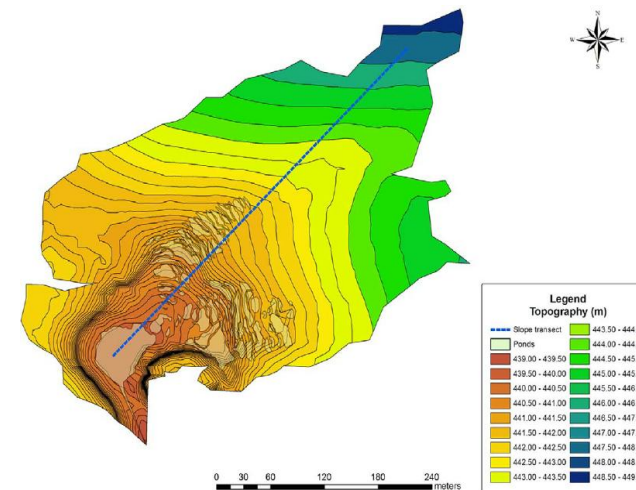
## 13.6.2 Graph Based Segmentation

- Images are considered as fully connected graphs where every node is a pixel and the edge is the link between every pair of points
- Segmentation by graph cuts
  - Delete links that cross between segments
  - Easiest to break links that have low cost (low similarity)
  - similar pixels should be in the same segments
  - dissimilar pixels should be in different segments
- Commonly used for segmentation



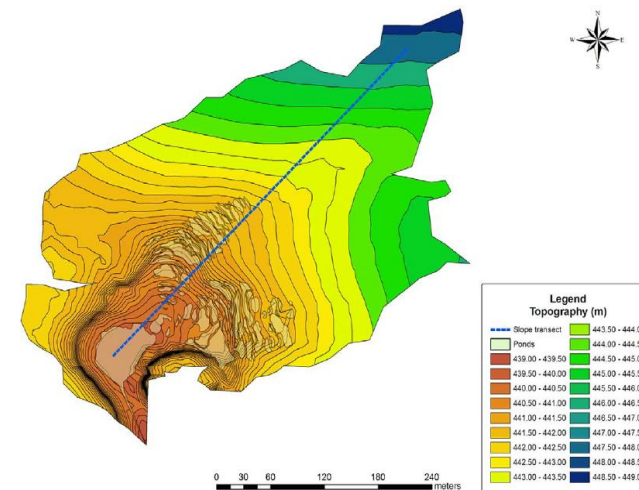
## 13.6.3 Watershed Algorithm

- Watershed algorithm allows to segment images into foreground and background.
- Also allows us to manually set seeds to choose segments of an image.
- In geography, a watershed is a land area that channels rainfall and snowmelt to creeks, streams, and rivers, and eventually to outflow points such as reservoirs, bays, and the ocean.
- These watersheds can then be segmented as topographical maps with boundaries.

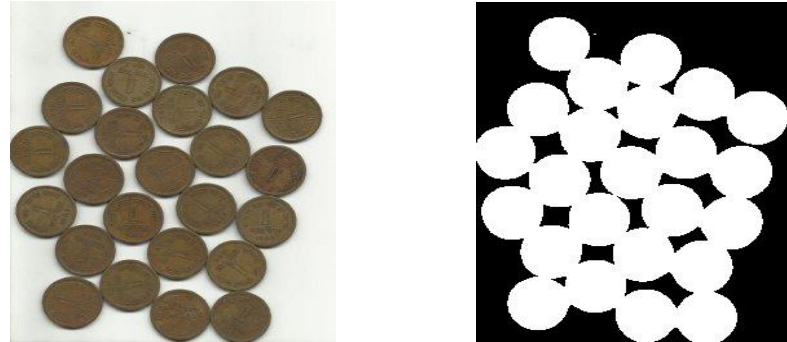


## 13.6.3 Watershed Algorithm

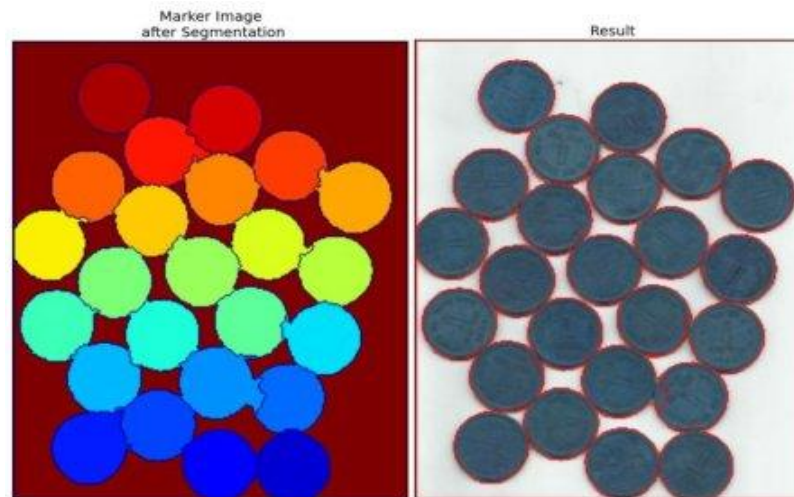
- Any grayscale image can be viewed as a topographic surface where **high intensity** denotes peaks and hills while **low intensity** denotes valleys.
- The algorithm fill every isolated valleys (local minima) with different colored water (labels).
- As the “water” rises, depending on the peaks (gradients) nearby, “water” from different valleys (different segments of the image), with different colors could start to merge.
- To avoid merging, it creates barriers (segment edge boundaries) in locations where “water” merges.
- Useful for segmenting images into background and foreground that is tough for other algorithms.



- Attempting to segment these coins can be difficult: It may be unclear to the algorithm if it should be treated as one large object or many small objects.



- The watershed algorithm can be very effective for these sort of problems.





- It also allows to provide our own custom “seeds” that allow us to manually start where the valleys of the watersheds go.
- We will draw on our own seeds to an image.



- And then calculate the image segments.

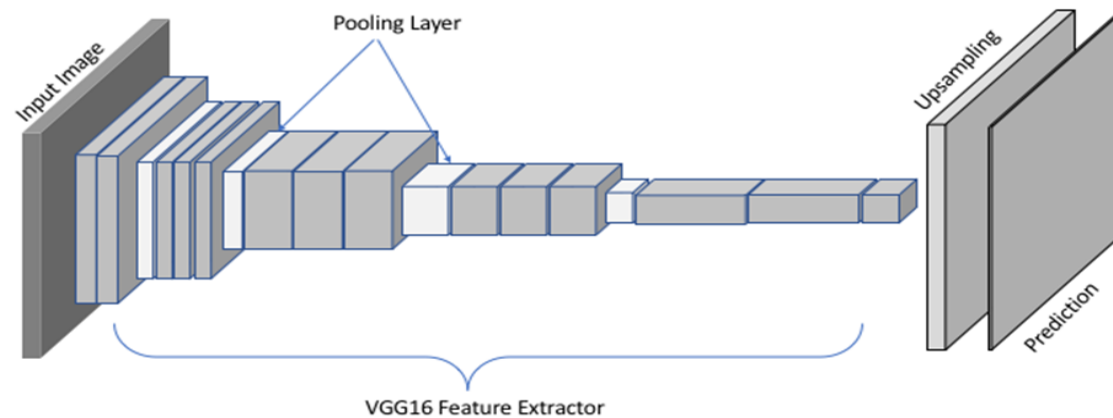


- DL based segmentation approaches recently grown, both in terms of accuracy and effectiveness.
- One of the popular models using CNN for segmentation is a **fully convolutional network (FCN)**.
- It has advantage of training end-to-end CNN to perform pixel-wise semantic segmentation.
- The output is an image with each pixel classified as either background or into one of the predefined categories of objects.

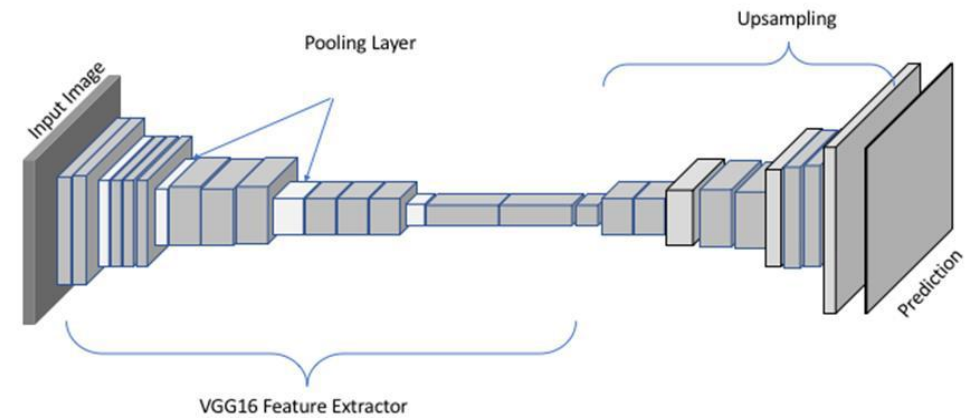
### How it works?

- As the layers are stacked hierarchically, the output from each layer gets downsampled.
- In the last layer the downsampled output is upsampled using a deconvolutional layer, resulting in the final output being the same size as that of the input.
- The deconvolutional layer is used to transform the input feature to the upsampled feature, however, the name is a bit misleading, as the operation is not exactly the inverse of convolution.
- This acts as transposed convolution, where the input is convolved after a transpose, as compared to a regular convolution operation.

- In the first model, the upsampling of the feature layer was done with a single layer. This can, however, be extended over to a hierarchical structure as shown in second layer.
- In the first model, the feature extractor is kept the same, while upsampling is updated with more deconvolutional layers where each of these layers upsamples features from the previous layer and generates an overall richer prediction.



Fully Convolutional Network for Segmentation



Network for Segmentation with Stage wise Upsampling



## **The Topics Covered in This Section**

- 13.7.1 Introduction
- 13.7.2 Challenges
- 13.7.3 MOSSE tracker
- 13.7.4 Deep SORT
- 13.7.5 Optical Flow
- 13.7.6 Meanshift and Camshift
- 13.7.7 Tracking APIs
- 13.7.8 Exercises

### **What is object tracking?**

- Tracking is the problem of estimating the position of an object over consecutive image sequences.
- Single object tracking and multiple object tracking require slightly different approaches.

### **Applications**

- Action recognition and Motion capture
- Self-driving cars
- Security and surveillance
- Augmented reality apps

### **Object tracking techniques**

- Optical Flow
- MeanShift and CamShift
- Tracking APIs with OpenCV

- **Object occlusion:** If the target object is hidden behind other objects in a sequence of images, then it becomes hard to detect the object and to update future images if it becomes visible again.
- **Fast movement:** Cameras, such as on smartphones, often suffers from jittery movement. This causes a blurring effect and, sometimes, the complete absence of an object from the frame. Therefore, sudden changes in the motion of cameras also lead to problems in tracking applications.
- **Change of shape:** If we are targeting non-rigid objects, changes in shape or the complete deformation of an object often lead to being unable to detect the object and also tracking failure.
- **False positives:** In a scene with multiple similar objects, it is hard to match which object is targeted in subsequent images. The tracker may lose the current object in terms of detection and start tracking a similar object.

- This is fast object tracking using correlation filter methods which comprises the following steps:
  1. For target object template  $T$  and input image  $I$ , take Fast Fourier Transform of both  $T$  and  $I$ .
  2. A convolution operation is performed between template  $T$  and image  $I$ .
  3. The result of step 2 is inverted to the spatial domain using Inverse Fast Fourier Transform.
- This correlation filter-based technique has limitations in the choice of  $T$ .
- As a single template image match may not observe all the variations of an object, such as rotation in the image sequence.
- Bolme and co-authors proposed a more robust tracker-based correlation filter termed as Minimum Output Sum of Squared Error (MOSSE) filter.
- In this method template  $T$  for matching is first learned by minimizing a sum of squared error as:

$$\min_{T^*} \sum_i |I_i \odot T^* - O_i|^2$$

Here,  $i$  is the training samples and the resulting learned template is  $T^*$ .

- Simple online and realtime tracking with a deep association metric (deep SORT) is an extension of, *simple online and realtime tracking* (SORT).

### ***Motion information***

- The state of each target at some point is modelled as:

$$(u, v, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$$

where  $(u, v)$  is bounding box center position,  $\gamma$  is aspect ratio,  $h$  is height, overdot means their respective velocities in image coordinates.

- A standard Kalman filter with constant velocity motion and linear observation model, is used to update the above target state.
- Link detections to existing tracks:

$$d^{(1)}(i, j) = (\mathbf{d}_j - \mathbf{y}_i)^T \mathbf{S}_i^{-1} (\mathbf{d}_j - \mathbf{y}_i) \quad (\text{cost})$$

where  $\mathbf{d}_j$  is the  $j$ -th bounding box detection,  $\mathbf{S}_i$  is the covariance matrix of the Kalman filter prediction,  $\mathbf{y}_i$  is the Kalman filter prediction bounding box.

The equation calculates the (Mahalanobis) distance of groundtruth detection and the Kalman filter prediction.

## *Appearance information*

- For each bounding box detection  $\mathbf{d}_j$ , we compute an appearance descriptor  $\mathbf{r}_j$  with  $\|\mathbf{r}_j\| = 1$  by L2 normalization, where  $\mathbf{r}_j$  comes from a convolutional neural network(wide residual).
- Keep a gallery  $\mathcal{R}_k = \{\mathbf{r}_k^{(i)}\}_{k=1}^{L_k}$  of the last  $L_k = 100$  associated appearance descriptors for each track  $k$ , i.e, only keep the last 100 descriptors.
- Distance between the  $i$ -th track and  $j$ -th detection in appearance space is the smallest cosine distance the  $i$ -th track and  $j$ -th detection that:

$$d^{(2)}(i, j) = \min\{1 - \mathbf{r}_j^T \mathbf{r}_k^{(i)} \mid \mathbf{r}_k^{(i)} \in \mathcal{R}_i\}$$

- Then, apply a threshold  $t^{(2)}$  to  $d^{(2)}(i, j)$ , check whether it's feasible to accept this link:

$$b_{i,j}^{(2)} = \mathbb{1}[d^{(2)}(i, j) \leq t^{(2)}]$$

## *Combine motion and appearance information*

- Combine both metrics using a weighted sum:

$$c_{i,j} = \lambda d^{(1)}(i, j) + (1 - \lambda) d^{(2)}(i, j)$$

This term is interpreted as the cost of associating the  $i$ -th track and  $j$ -th detection.

- check whether motion and appearance are both less than the threshold:

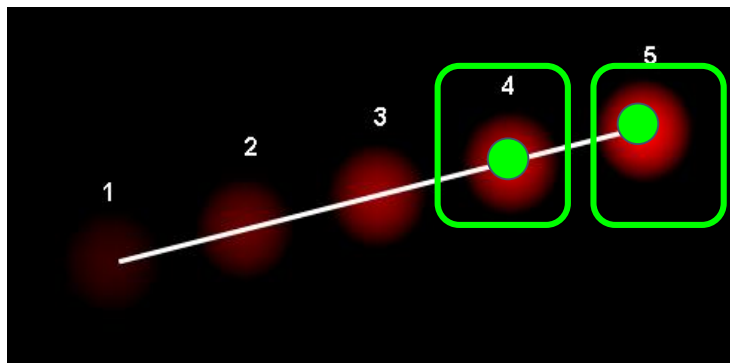
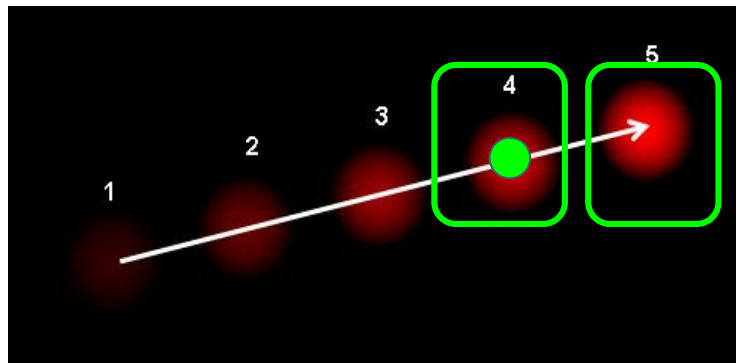
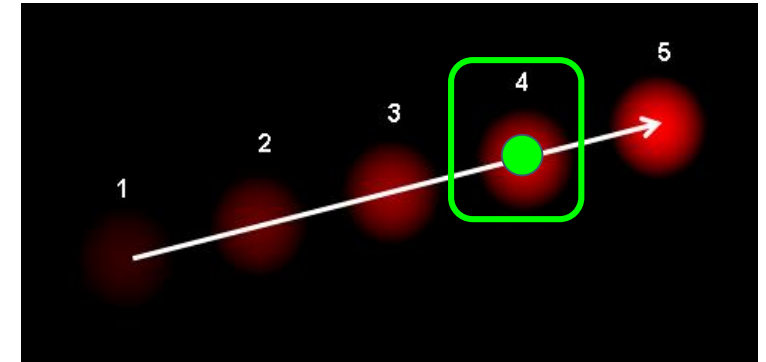
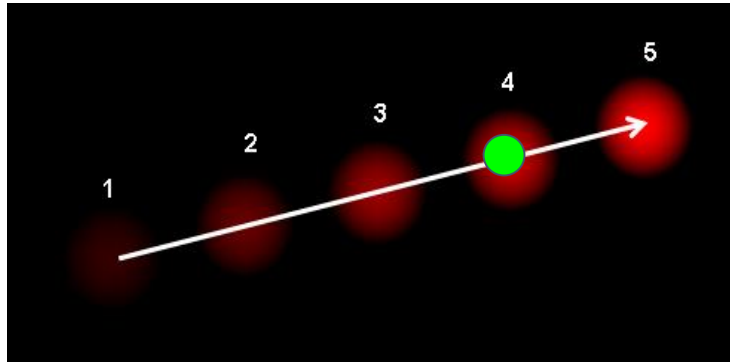
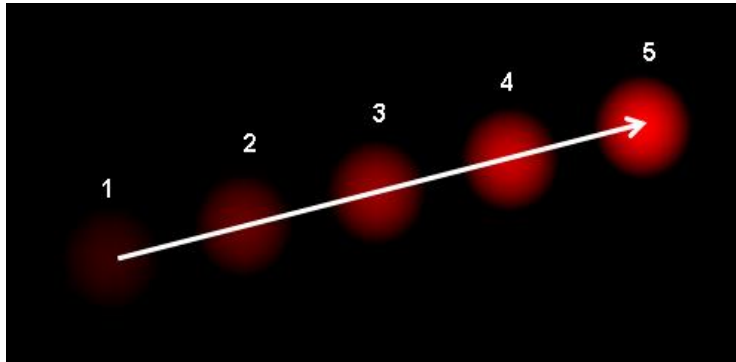
$$b_{i,j} = \prod_{m=1}^2 b_{i,j}^{(m)}$$

This term is interpreted as the cost of associating the  $i$ -th track and  $j$ -th detection.

- Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera.
- Optical Flow Analysis has a few assumptions:
  - The pixel intensities of an object do not change between consecutive frames.
  - Neighbouring pixels have similar motion.
- The optical flow methods in OpenCV will first take in a given set of points and a frame.
  - Then it will attempt to find those points in the next frame.
  - It is up to the user to supply the points to track.

**Example:** Here we display a five frame clip of a ball moving up and towards the right.

- Note that given just this clip, we can not determine if the ball is moving or if the camera moving.
- Pass the previous frame, previous points and the current frame to the Lucas-Kanade function.
- The function then attempts to locate the points in the current frame.





- Lucas-Kanade computes optical flow for a **sparse** feature set.
- Meaning only the points it was told to track.
- But what if we wanted to track all the points in a video?
  
- Gunner Farneback's algorithm (also built in to OpenCV) calculates **dense** optical flow.
- Dense optical flow will calculate flow for all points in an image.
- It will color them black if no flow (no movement) is detected.

- MeanShift track the target object by calculating the color histogram of the target area and then keep sliding the tracking window to the closest match (the cluster center).
- Just using MeanShift won't change the window size if target moves away or towards the camera.
- Extend MeanShift into CAMshift (Continuously Adaptive MeanShift) to update window size.

- There are many Object Tracking methods.
- Fortunately, many have been designed as simple API calls with OpenCV.
- Let's explore a few of these easy to use Object Tracking APIs with OpenCV!

### **BOOSTING TRACKER**

- Based off AdaBoost algorithm (the same underlying algorithm that the HAAR Cascade based Face Detector Used).
- Evaluation occurs across multiple frames.

### **Pros and Cons**

- Pros:
  - Very well known and studied algorithm.
- Cons:
  - Doesn't know when tracking has failed.
  - Much better techniques available!

### MIL TRACKER

- Multiple Instance Learning
- Similar to BOOSTING, but considers a neighborhood of points around the current location to create multiple instances.
- Check project page for more details.

### Pros and Cons

- Pros:
  - Good performance and doesn't drift as much as BOOSTING.
- Cons:
  - Failure to track an object may not be reported back.
  - Can't recover from full obstruction.

### KCF TRACKER

- Kernelized Correlation Filters
- Exploits some properties of the MIL Tracker and the fact that many data points will overlap, leading to more accurate and faster tracking.

### Pros and Cons

- Pros:
  - Better than MIL and BOOSTING
  - Great first choice!
- Cons:
  - Can not recover from full obstruction of object.

### TLD TRACKER

- Tracking, Learning, and Detection
- The tracker follows the object from frame to frame.
- The detector localizes all appearances that have been observed so far and corrects the tracker if necessary.
- Tracking, Learning, and Detection
- The learning estimates detector's errors and updates it to avoid these errors in the future.

### Pros and Cons

- Pros:
  - Good at tracking even with obstruction in frames.
  - Tracks well under large changes in scale.
- Cons:
  - Can provide many false positives.

### MedianFlow TRACKER

- Internally, this tracker tracks the object in both forward and backward directions in time and measures the discrepancies between these two trajectories.

### Pros and Cons

- Pros:
  - Very good at reporting failed tracking.
  - Works well with predictable motion.
- Cons:
  - Fails under large motion (fast moving objects)



### **The Topics Covered in This Section**

13.8.1 Books

13.8.2 Video Lectures

13.8.3 Research Articles

- Ronzalez, R.C. (2018). *Digital Image Processing*. Pearson Education.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- Ronzalez, R.C., Woods, R.E., and Eddins, S.L. (2017). *Digital Image Processing with MATLAB*. McGraw Hill Education.
- Bradski, G., Kaehler, A. (2008). *Learning OpenCV*. O'Reilly.
- Solem J.E.(2012). *Programming Computer Vision with Python*. O'Reilly.

- UCF Computer Vision Video Lectures 2014

[https://www.youtube.com/watch?v=715uLCHt4jE&list=PLd3hlSJxX\\_ImKP68wfKZJVIPTd8Ie5u-9](https://www.youtube.com/watch?v=715uLCHt4jE&list=PLd3hlSJxX_ImKP68wfKZJVIPTd8Ie5u-9)

- Introduction to Computer Vision | Udacity

[https://www.youtube.com/watch?v=2S4nn7S8Hk4&list=PLAwxTw4SYaPnbDacyrK\\_kB\\_RUkuxQB1Cm](https://www.youtube.com/watch?v=2S4nn7S8Hk4&list=PLAwxTw4SYaPnbDacyrK_kB_RUkuxQB1Cm)

- Stanford Computer Vision | Fei Fei Li

[https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PLf7L7Kg8\\_FNxHATtLwDceyh72QQL9pvpQ](https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PLf7L7Kg8_FNxHATtLwDceyh72QQL9pvpQ)

- Computer Vision Lectures | William Hoff

<https://www.youtube.com/watch?v=skaQfPQFSyY&list=PL7v9EfkjLswLfjcI-qia-Z-e3ntl9l6vp>

- Convolutional Neural Networks | Deeplearning.ai

[https://www.youtube.com/watch?v=ArPaAX\\_PhIs&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF](https://www.youtube.com/watch?v=ArPaAX_PhIs&list=PLkDaE6sCZn6Gl29AoE31iwdVwSG-KnDzF)

- Computer Vision Lectures | Scott Butters

<https://www.youtube.com/watch?v=ZF-3aORwEc0&list=PLtvdbXJRL1QGu9knxdIsF6Y5O5MgVJ5HR>

Thank You