

Lecture 11

Recap

- ▶ HTML Canvas
 - ▶ Comparison to SVG browser graphics
 - ▶ Drawing shapes
 - ▶ Basic Trigonometry
 - ▶ Basic collision detection
 - ▶ Animation and User Interaction
 - ▶ Examples
 - ▶ Advanced collision handling
 - ▶ Linear Algebra and Transformations
- ▶ SVG
- ▶ D3
- ▶ 3D - WebGL/Three.js
- ▶ Computer Vision

Edges - recap

- ▶ Similarities/Differences

Image filtering - moving average - 1d

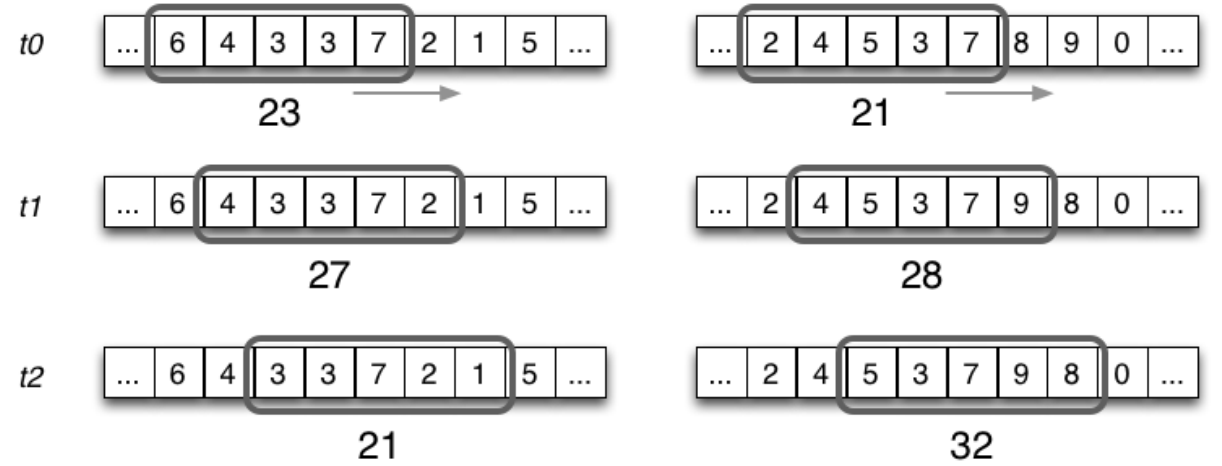
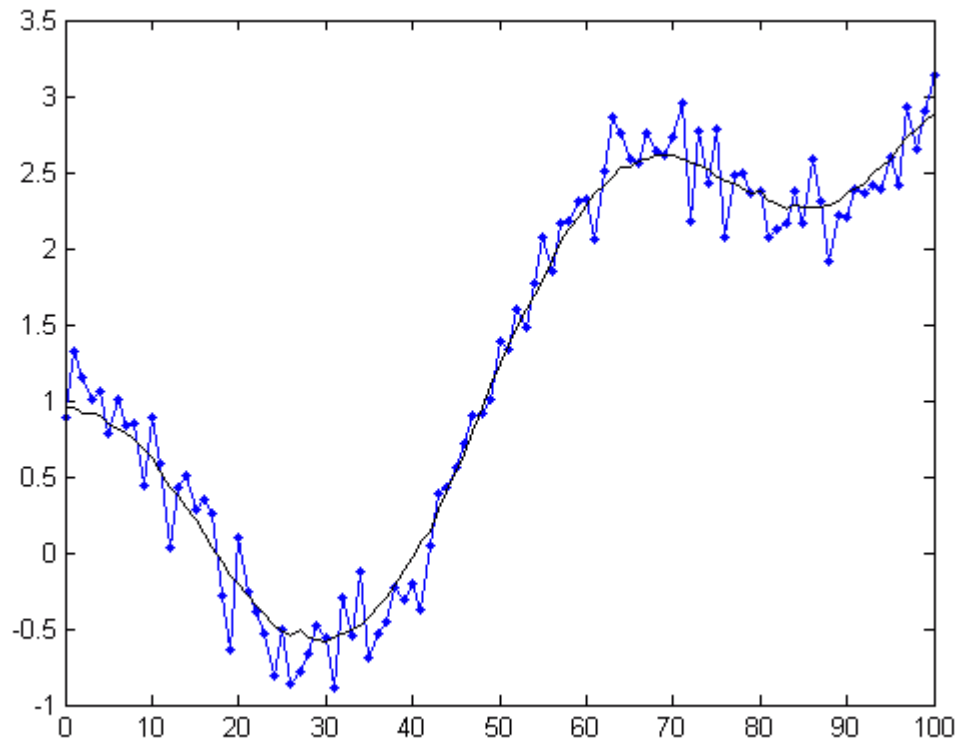
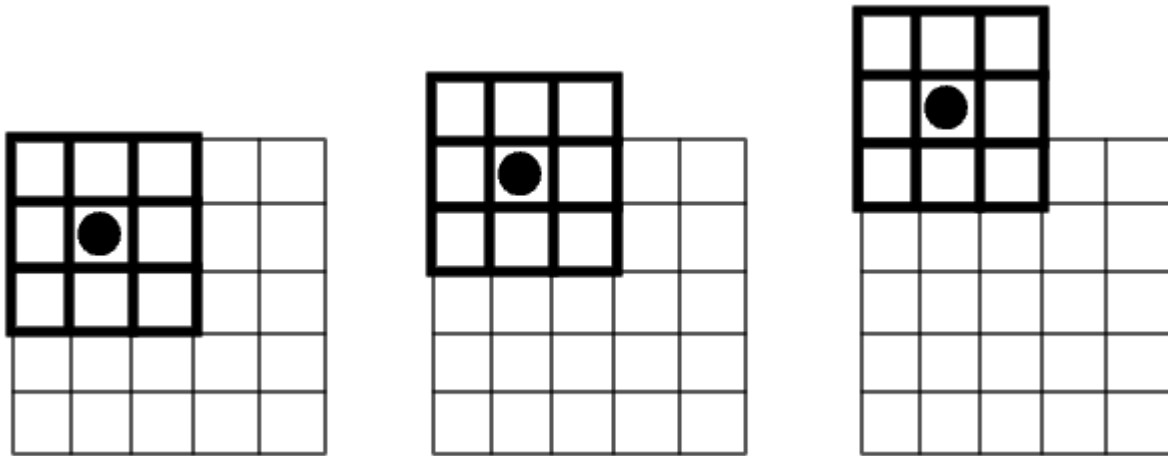


Image filtering - moving average - 2d

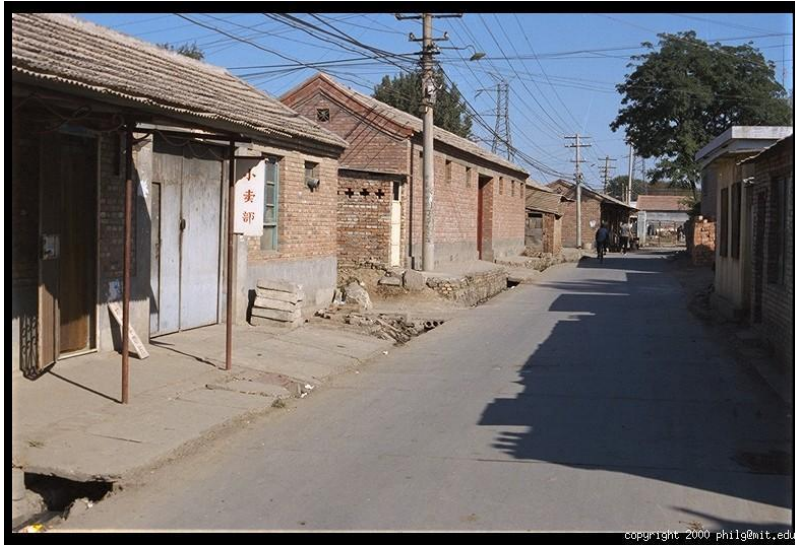


Images in OpenCV

- ▶ Grayscale - 2d array
- ▶ Colour - 3d array

```
9ca197af64736d708a82808c81948f8e9488717b6d9e98a2989bd1717b7a876d
9b99a4ad687a6e778290878a8380818885818b816895989f9a9bd5647d7d864e
9da4aea766756a6e79908e8c9681798784707b7e7587a3a19da38cda72845d35
a3b495a160656a74787a807d809ea3c580868d7c7785969a989697cf98502f35
ab969b9a5d626c7e807f8f74869ca9b4c9d37b7364819a9b958e978fd5393e38
9e619cb361616e727b7f6b86857aadadc7d2d860698e9a6f908d8ea33635365b
7556959a63686a77816a747781989faac6c6c3e5548c9c497992a12b4740479b
635f96ad5e5673729776828a898cbfc0d3d3c1c6e67b9c5527d3773c3f428fa6
626198a655686b77b075767f6eb1adbfbec7c2cecd7c9a3dardl1da3b2b35a3a1
697196b760615d6e9f6f79788aa3aeaca8bbb9c8c4c4d7d8c7c5823f36a3a2a0
6e5e9db25c606865ab7e5f84a58fabaaa29db7a2a7d4c2c3d2822b398f9c9ba2
686balac5d636754b38d749e8f8f839a838a359eb8b9c7c154d732378fa5a09d
6a6996ae66606d6cbe8f827e8c823e763b2fcd0b8ca6592bc322c70a4a8a6a1
6e5f96ae645963705d72847a424331505abcc2a7c2955fcc343e3c879ea5a293
6c6e96ae6862697d657a8143324434229fbfblbe8c83350323538a6979e999b
666b99b664666479867d60413f342ca0c794afbfc9d53b8b3d3788a6a498988c
626c98af686e6d809c60316f6d3283b9646d7ca3ca4848b3453e8ca3a59aa38d
6a6594b76d696fac37356639283fc47e69c7849a83ad37b13630a299999c8e8a
6b6088cl736f7832863b843635bb5f9aaeae898cb57f34a22f90al9a9982acc9
50548fb26e7232318e36328f68a7678ba6b2908e9f893e6c3f969a9a9584bbd2
5d498bb7756d709b9b382d2bb133628a9ca27876bl8c3e2f24a29b9284bed4cb
56439db66c6b41406c68354f343a6d8a8a9997b3953141386c96989a83c5ccd0
3d3e9ab36d9a29447134d42f3e534d8a95707a85993962389c98928784d3cbce
383888b274642e6b7c4cb533363b456d89868595393f5957a09b96987ecbd8d0
31348dbd6f3d4c2d66852b4035313e3b3789b28e2e376a36998e9d8b7cdld5b8
813c7ab3702626654366a93f303a366a7b9597c4cb606a4897816b6e93d49f4d
4a9b73b0a932243e6d647b693e3e4a5f8a869bb6c0d4834a978c9073cdd05d7c
2e9e72afa2363227307a71a73c3c6764898e8fa2abcbad322988b77bade746a65
27b75dbcal303848265c652f5737806e91849a9fb9cbcf31999280cdd27c6c6c
28b759b8823f4c4b308c412c2b5c7e85928e9494abbbdcde56698b8fcl54746f
```

Colour Images



R



G



B

Images in Matlab/Octave/OpenCV

- Images represented as a matrix
- Suppose we have a NxM RGB image called “im”
 - $\text{im}(1,1,1)$ = top-left pixel value in R-channel
 - $\text{im}(y, x, z)$ = y pixels down, x pixels to right in the z^{th} channel
 - $\text{im}(N, M, 3)$ = bottom-right pixel in B-channel
- `imread(filename)` returns a uint8 image (values 0 to 255)
 - Convert to double format (values 0 to 1) with `im2double`

column →

↓ **row**

0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

R

0.92	0.99
0.95	0.91
0.91	0.92
0.97	0.95
0.79	0.85
0.45	0.33
0.49	0.74
0.82	0.93
0.90	0.99
0.93	0.97
0.99	0.93

G

0.92	0.99
0.95	0.91
0.91	0.92
0.97	0.95
0.79	0.85
0.45	0.33
0.49	0.74
0.82	0.93
0.90	0.99
0.93	0.97
0.99	0.93

B

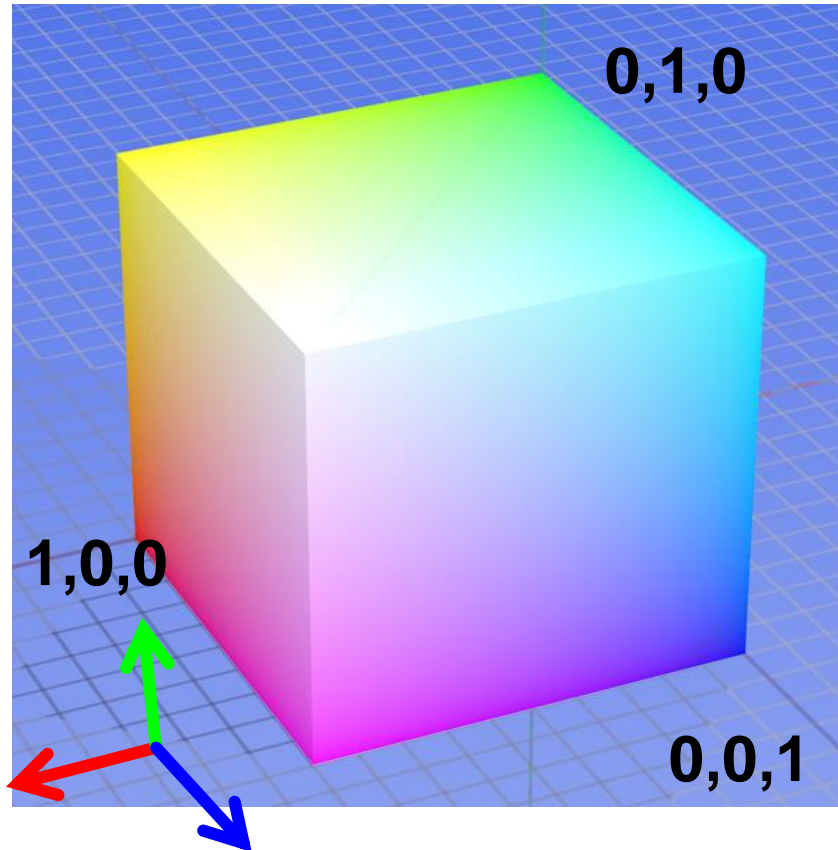
Color spaces

- How can we represent colour?



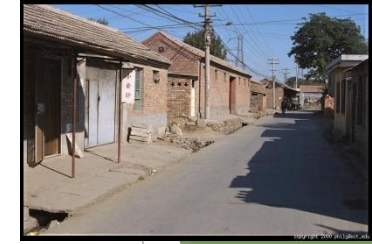
Color spaces: RGB

Default color space



Some drawbacks

- Strongly correlated channels
- Non-perceptual



R

(G=0,B=0)



G

(R=0,B=0)

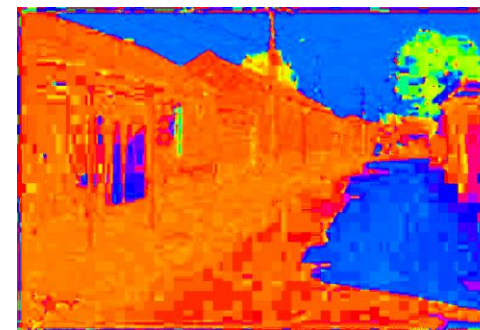
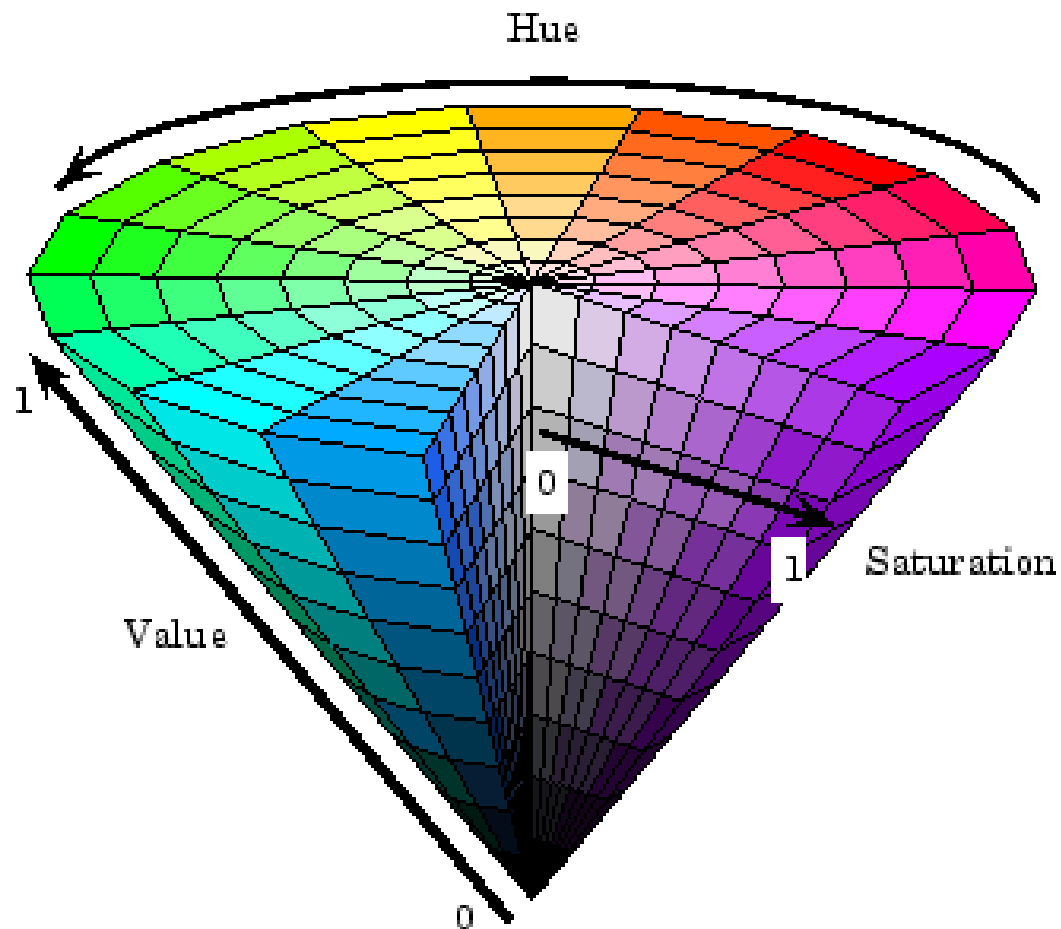


B

(R=0,G=0)

Color spaces: HSV

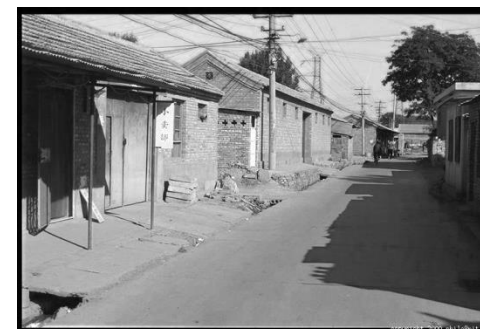
Intuitive color space



H
(S=1,V=1)



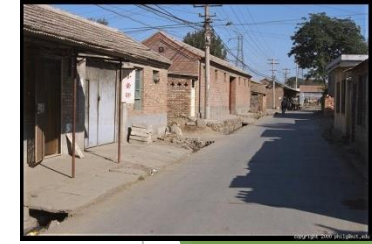
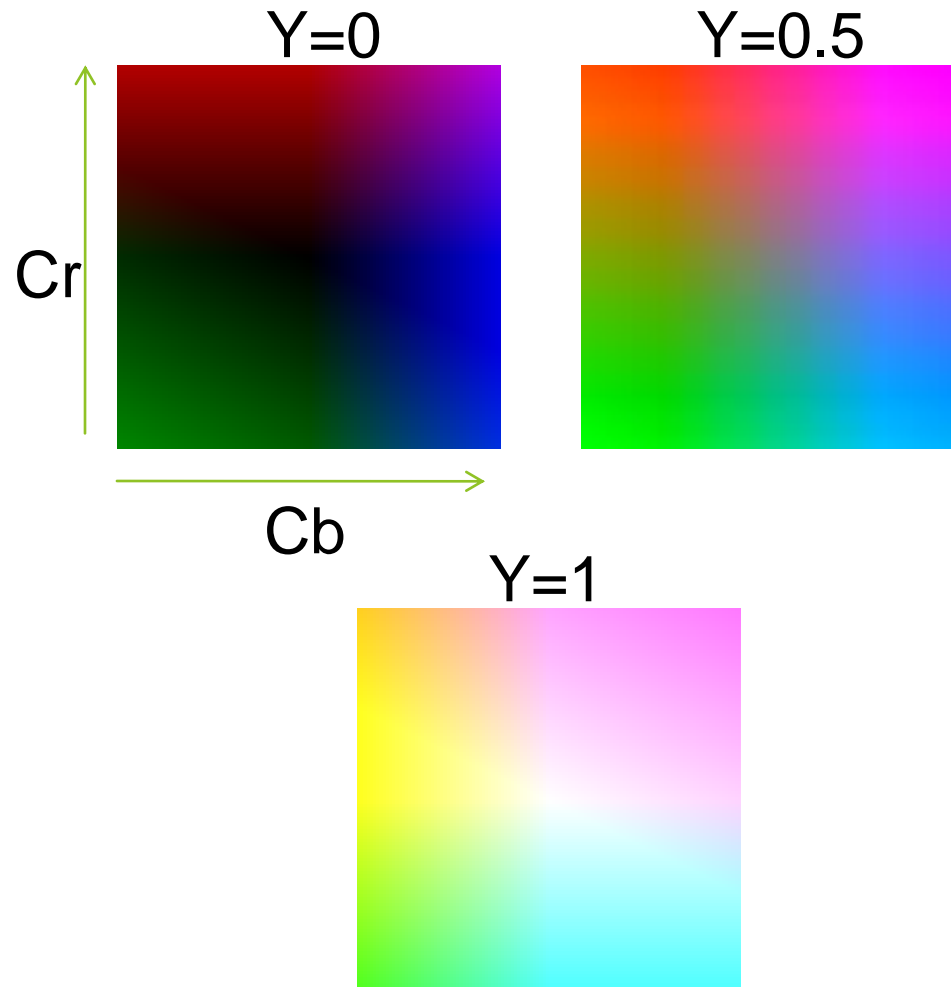
S
(H=1,V=1)



V
(H=1,S=0)

Color spaces: YCbCr

Fast to compute, good for compression, used by TV



Y (luma)
(Cb=0.5,Cr=0.5)



Cb (Chrominance - blue)
(Y=0.5,Cr=0.5)



Cr (Chrominance - red)
(Y=0.5,Cb=0.5)

Features in Computer Vision

- ▶ We've seen edges
- ▶ Corner detection also commonly used
- ▶ Like making a jigsaw puzzle
 - ▶ How does a computer know what it's seeing from a small part of an image
 - ▶ How does a human do it
 - ▶ Look for specific patterns or specific features which are unique, which can be easily tracked and easily compared

Features



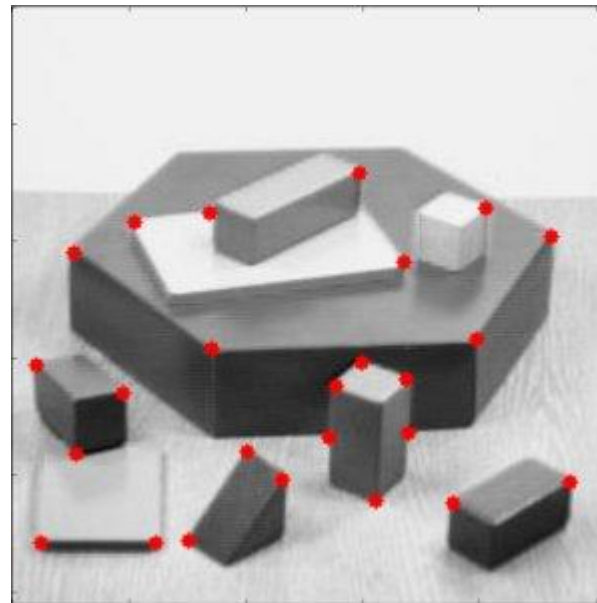
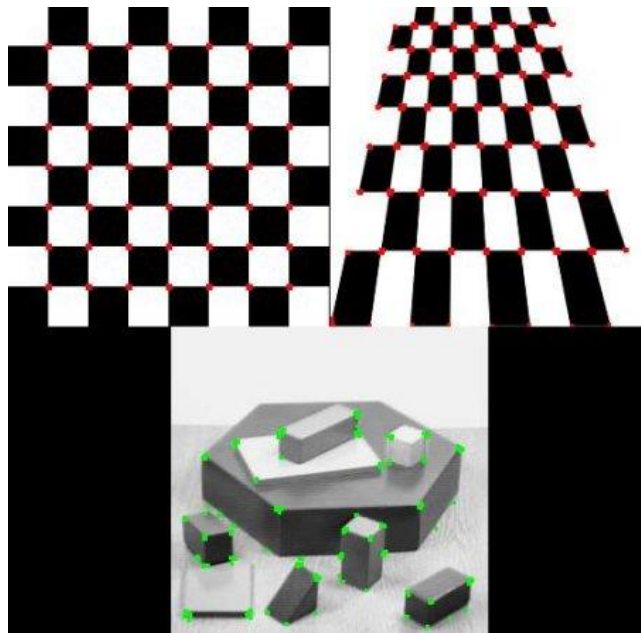
Features

- ▶ A and B are flat surfaces, and they are spread in a lot of area. It is difficult to find the exact location of these patches.
- ▶ C and D are much simpler. They are edges of the building. You can find an approximate location, but exact location is still difficult. It is because, along the edge, it is same everywhere. So an edge is a much better feature compared to flat area, but not good enough (It is good in jigsaw puzzle for comparing continuity of edges).
- ▶ Finally, E and F are some corners of the building. And they can be easily located. Because at corners, wherever you move this patch, it will look different. So they can be considered as a good feature.

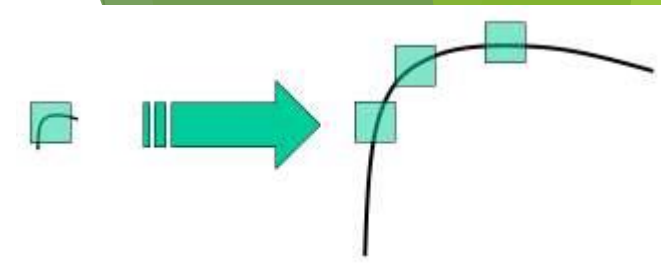


Several corner detection algorithms

- ▶ Corners are regions in the image with large variation in intensity in all the directions.
 - ▶ Harris - Early attempt (1988)
 - ▶ Shi Tomasi - Improvement (1994)
 - ▶ More appropriate for tracking



Several corner detection algorithms

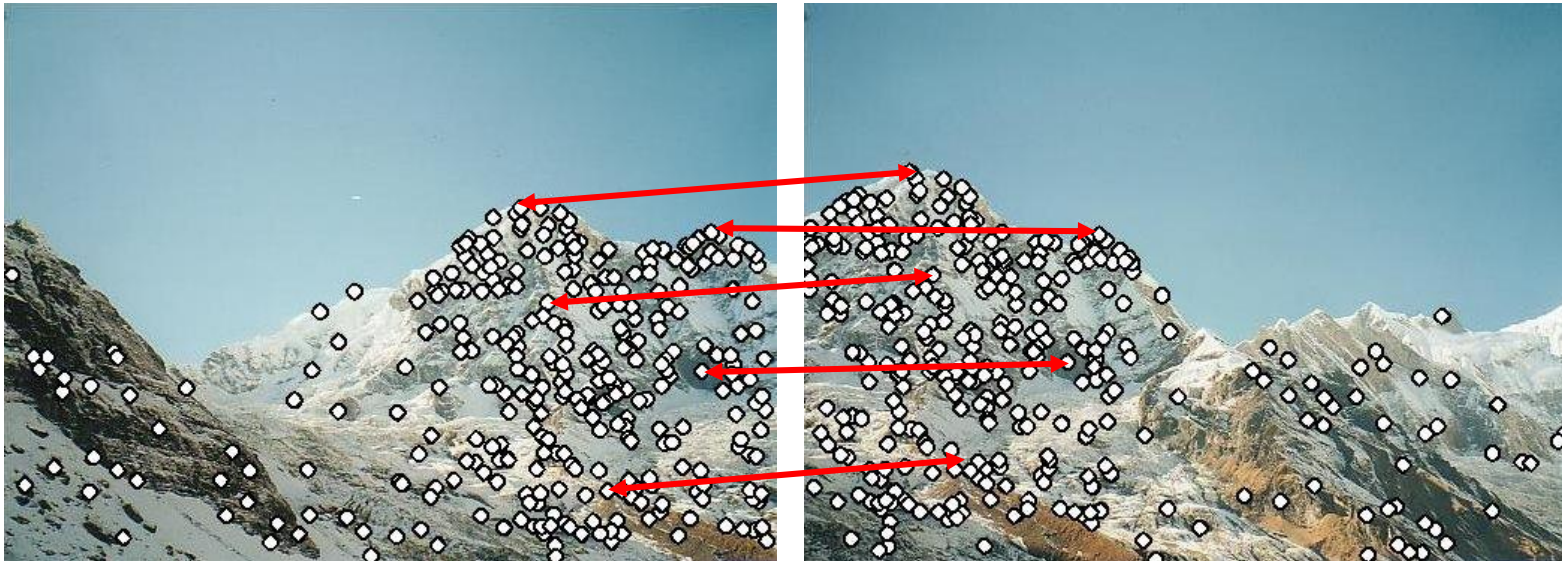


- ▶ Harris and Shi Tomasi algorithms are rotation-invariant, which means, even if the image is rotated, we can find the same corners. It is obvious because corners remain corners in rotated image also.
- ▶ What about scaling? A corner may not be a corner if the image is scaled. For example, check a simple image above. A corner in a small image within a small window is flat when it is zoomed out. So Harris/Shi Tomasi is not scale invariant.
 - ▶ SIFT/SURF
 - ▶ Scale invariant



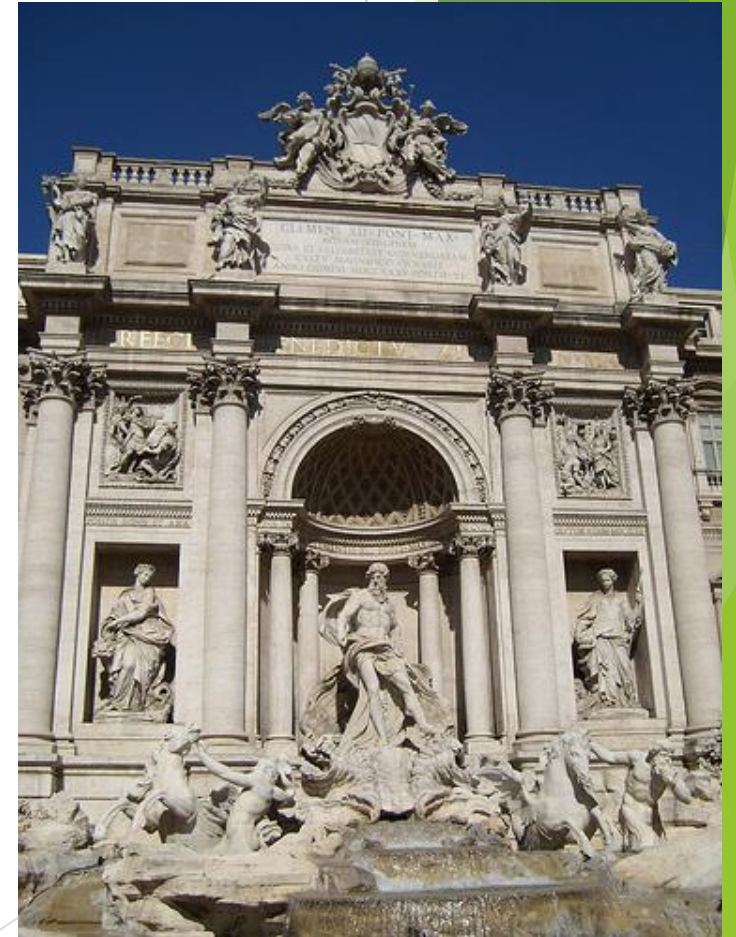
Feature Matching

- What things in this image matches with things in the other image?



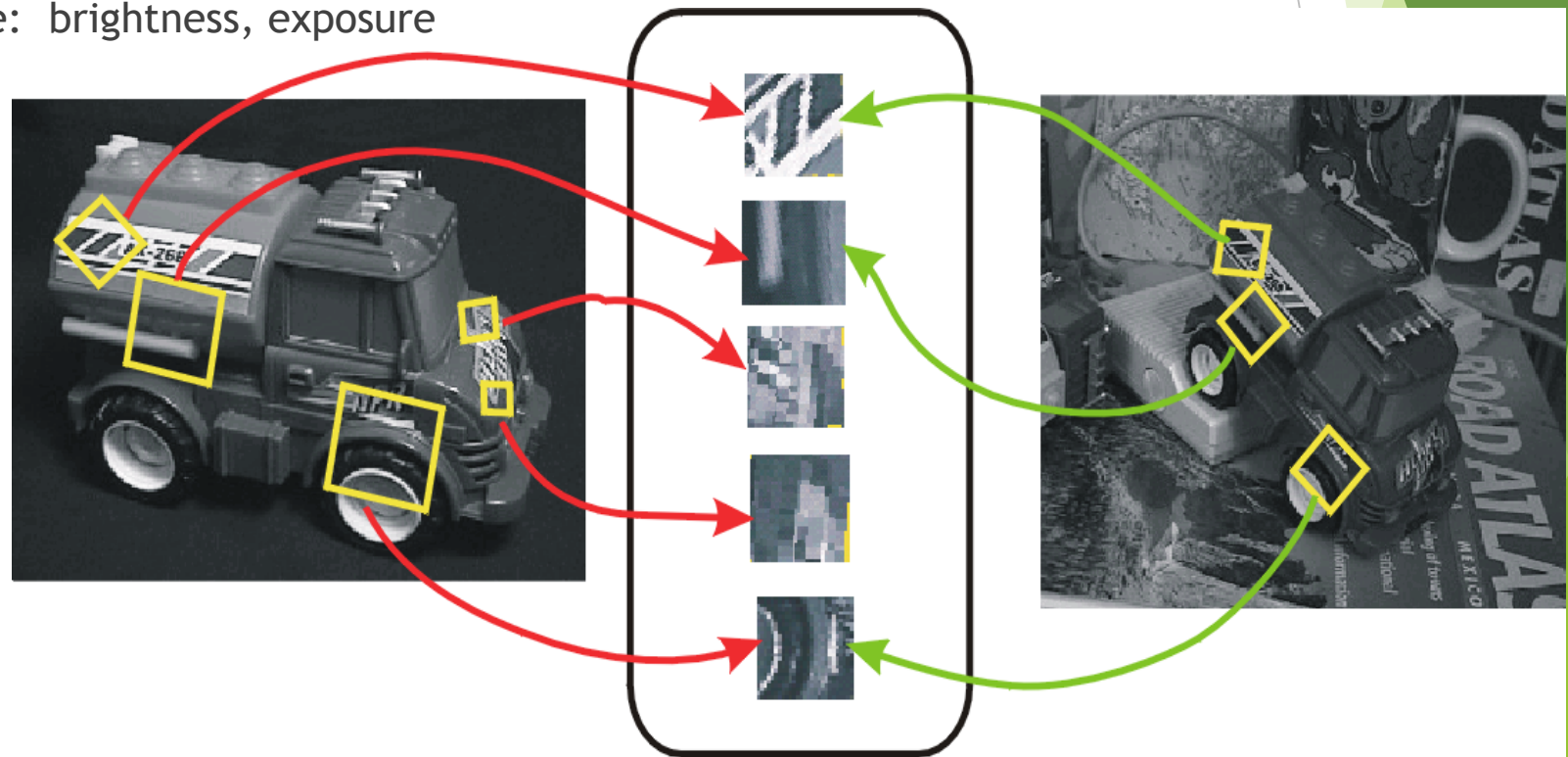
Feature Matching

- What things in this image matches with things in the other image?



Invariant local features

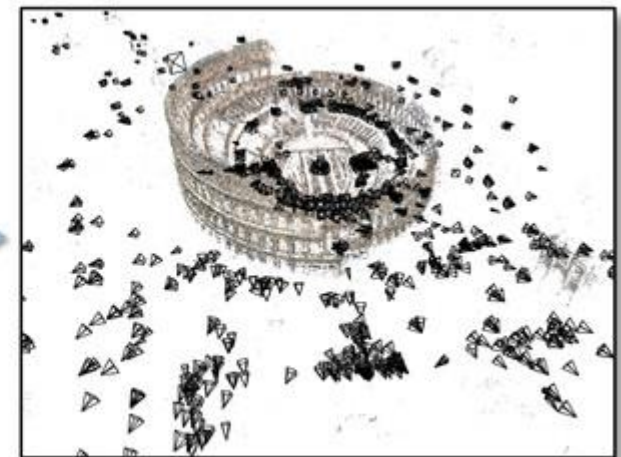
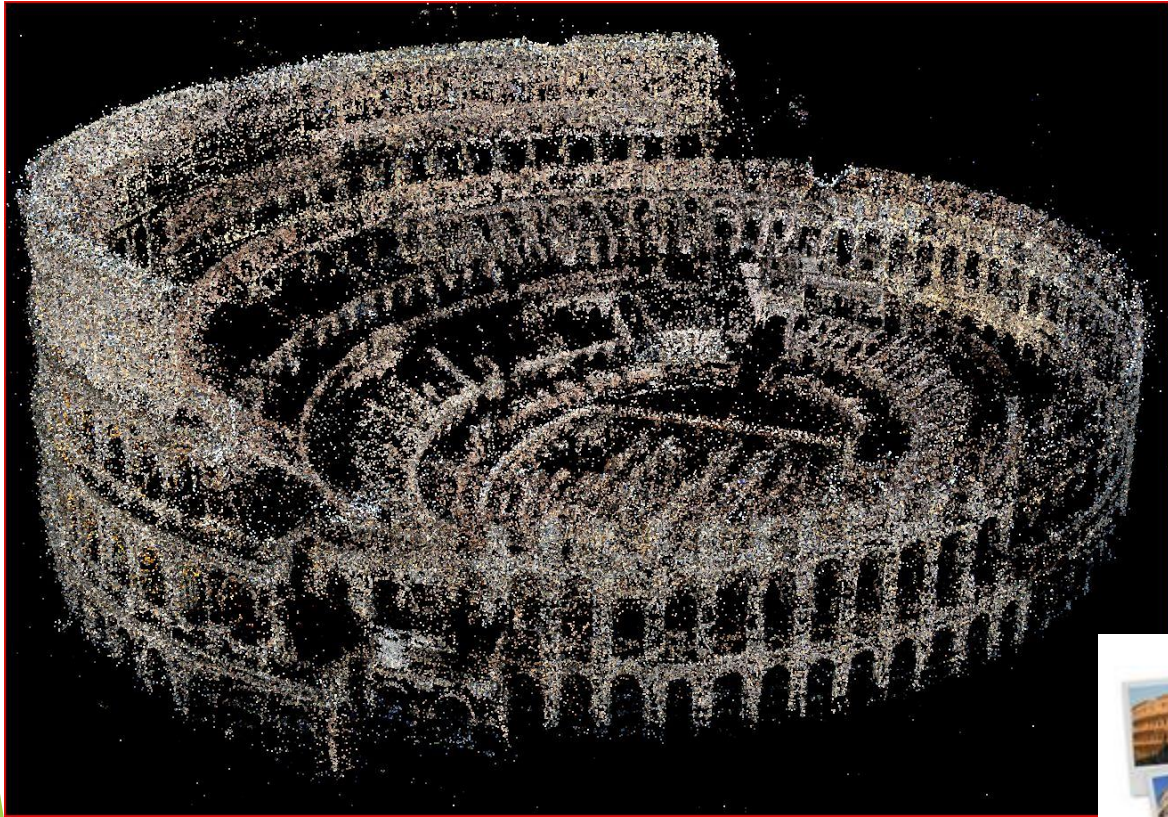
- Buzzword is “invariance”
 - Geometric invariance: translation, rotation, scale
 - Photometric invariance: brightness, exposure



Feature Detection/Matching Applications

- ▶ Object detection and tracking
- ▶ Panorama stitching
- ▶ Stitching photos together in animation
- ▶ Training of object detection, object recognition, and image retrieval systems
- ▶ Camera calibration for single and stereo cameras
- ▶ Stereo vision, including rectification, disparity calculation, and 3-D reconstruction
- ▶ 3-D point cloud processing

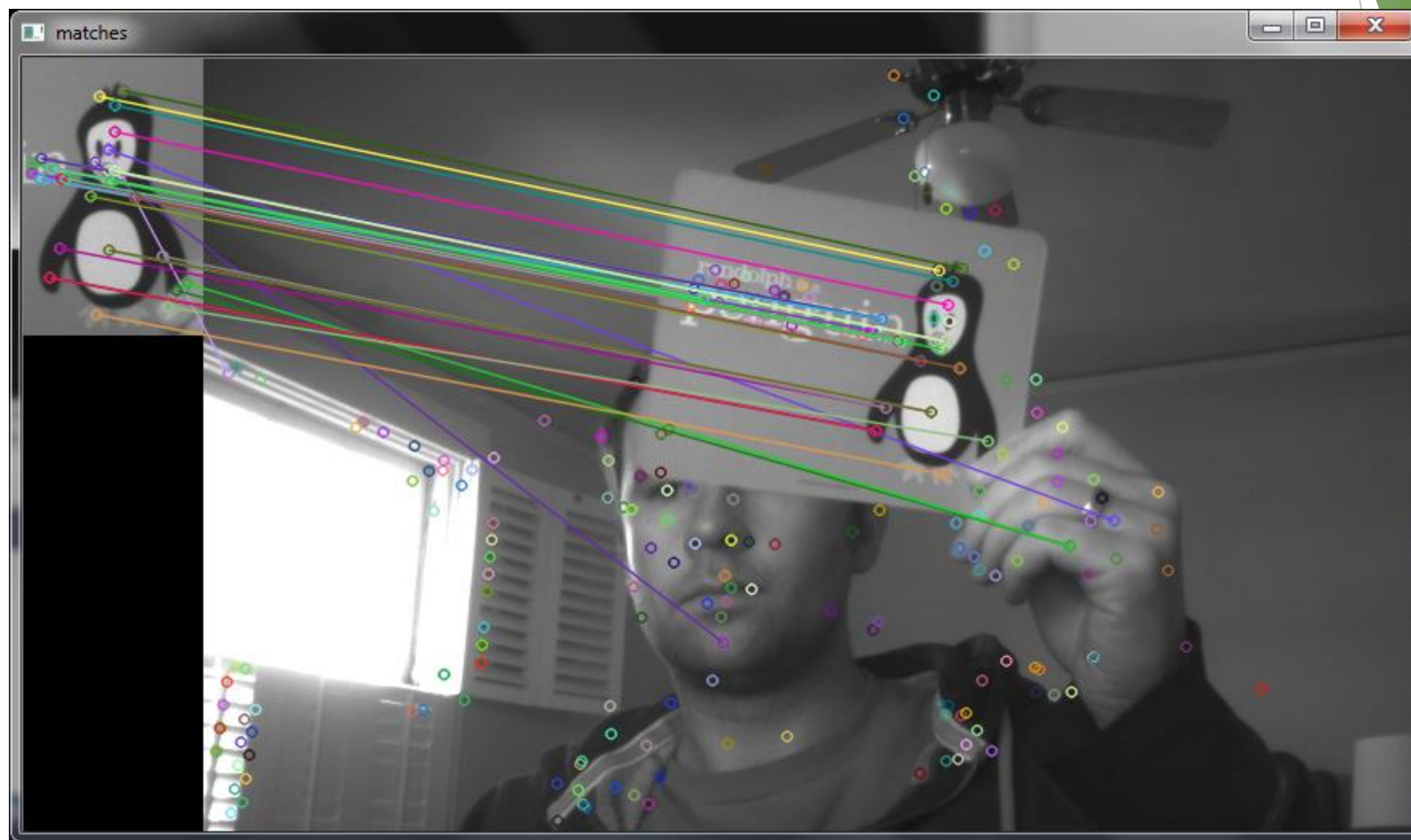
Structure from Motion (SFM)

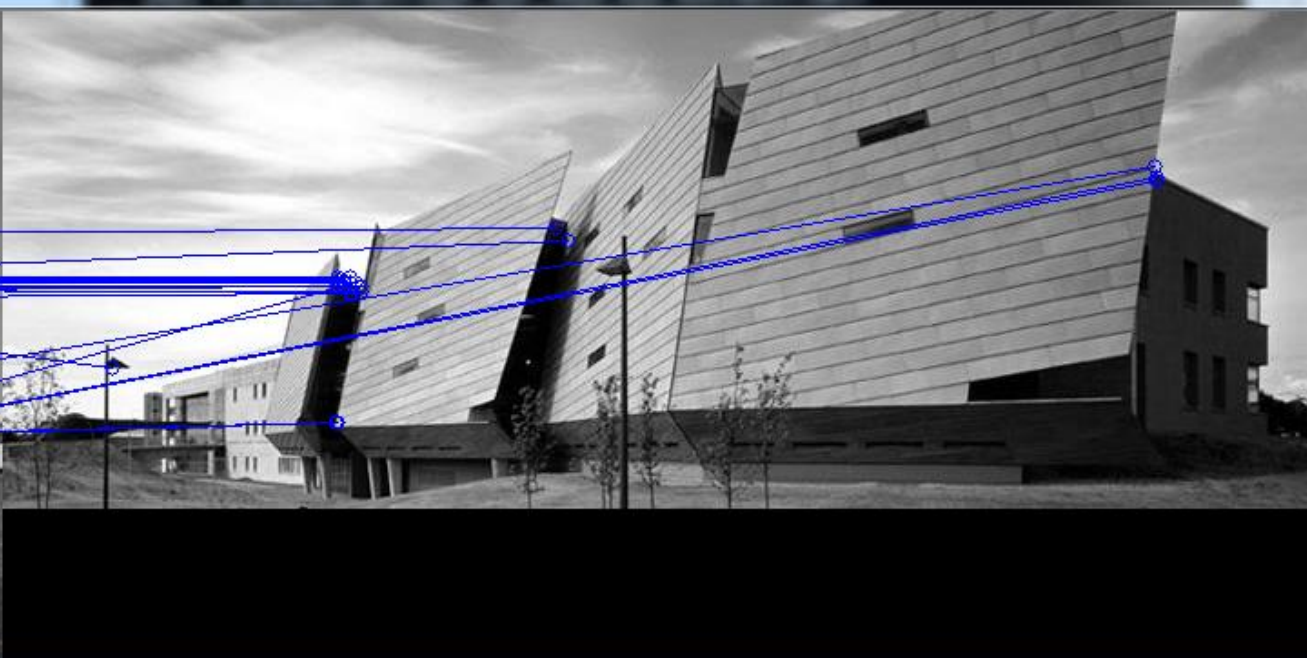
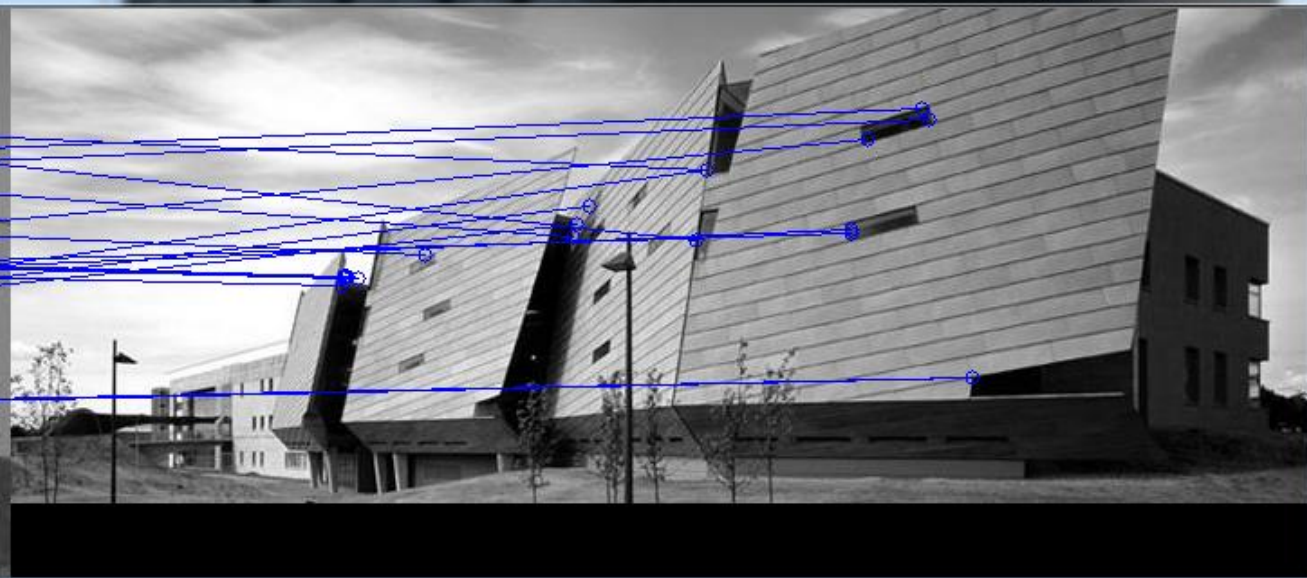
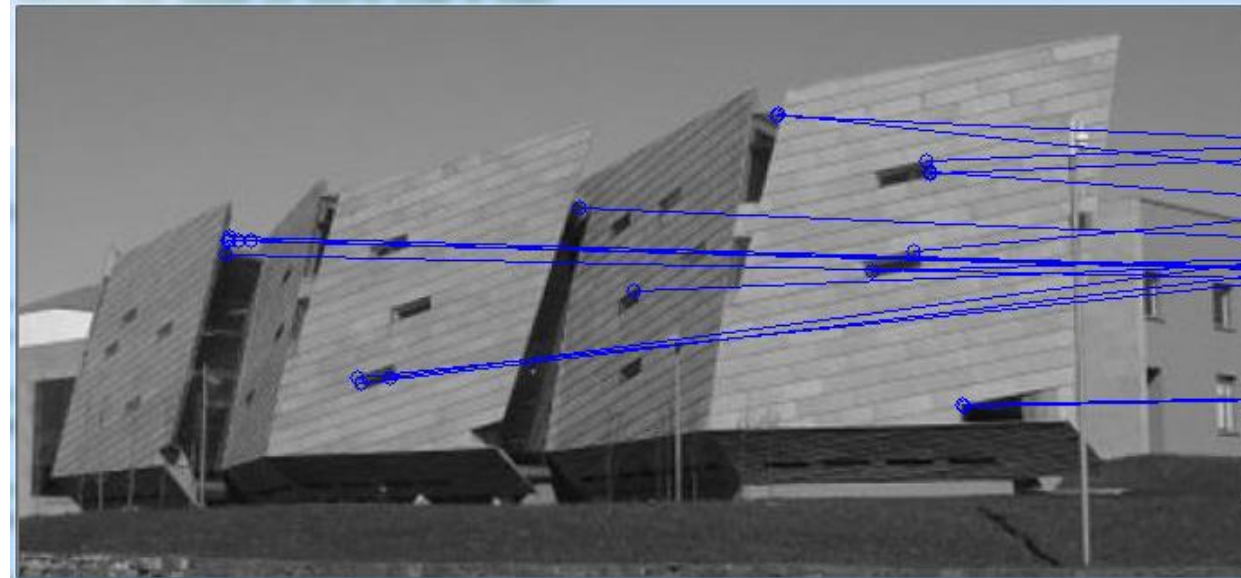


Feature Matching

- ▶ OpenCV provides two feature matching algorithms
 - ▶ Brute-force method
 - ▶ Takes one feature in first image and is matched with all other features in second image using some distance calculation. And the closest one is returned.
 - ▶ `cv2.BFMatcher()` takes 2 parameters
 - ▶ `normType` - set to `cv2.NORM_L2`
 - ▶ `crossCheck` - experiment with true and false settings (false by default)
 - ▶ `cv2.drawMatches()` displays the feature matches between the two images
 - ▶ FLANN (Fast Library for Approximate Nearest Neighbors)
 - ▶ Much faster than `BFMatcher` for large datasets.







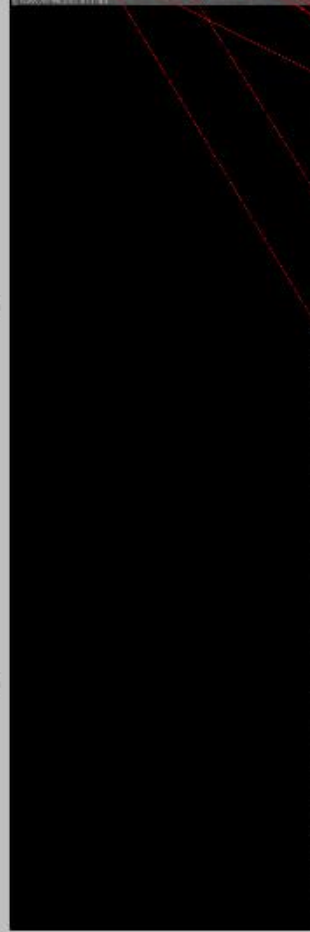
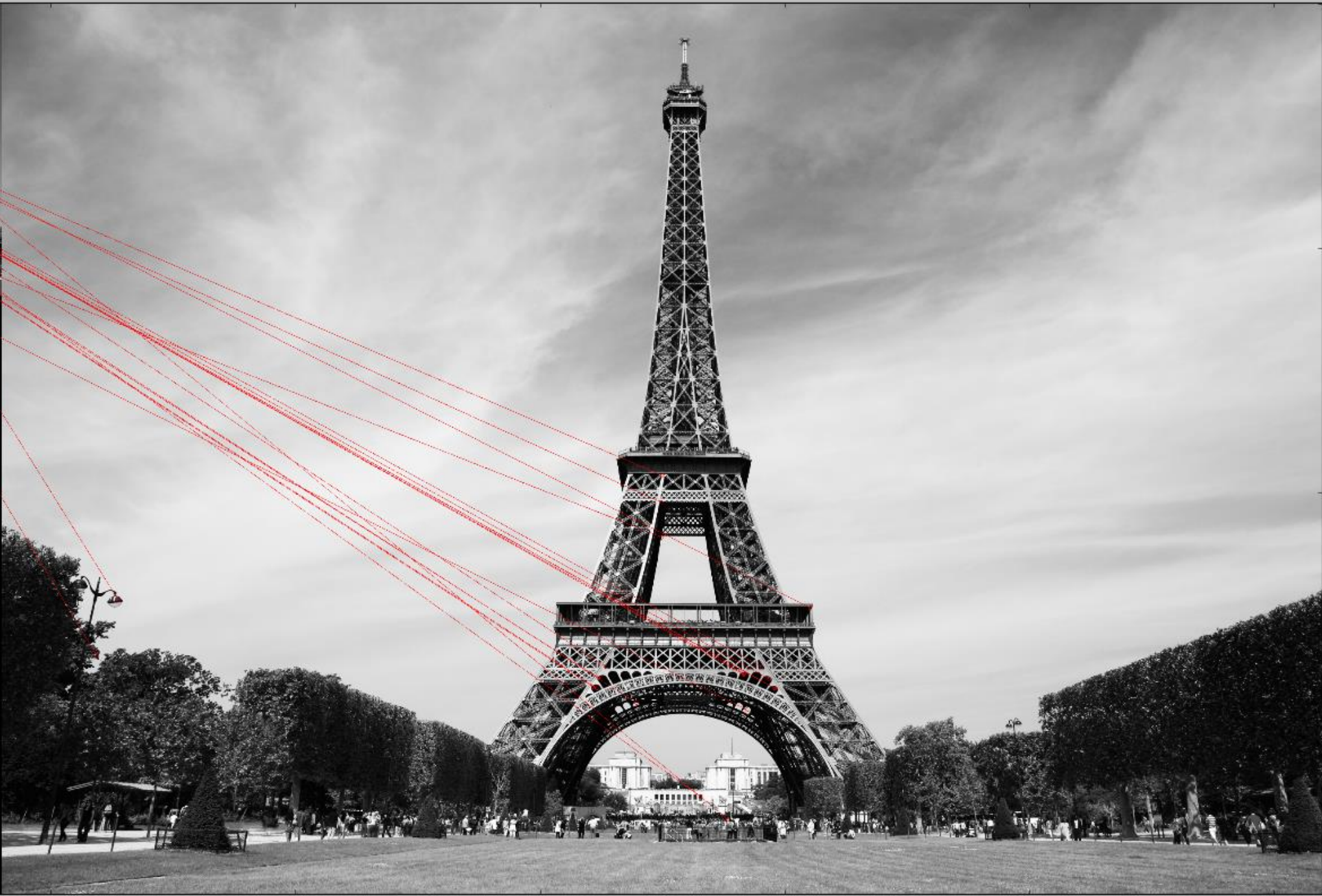
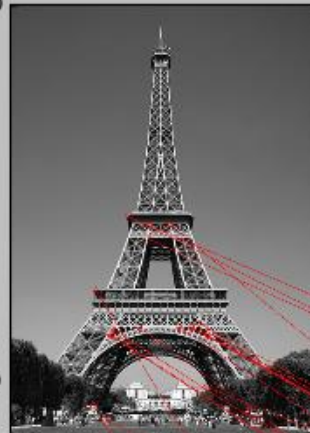
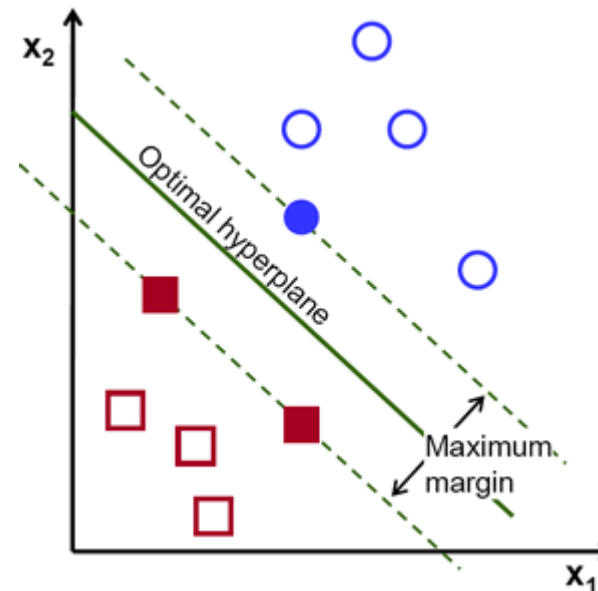
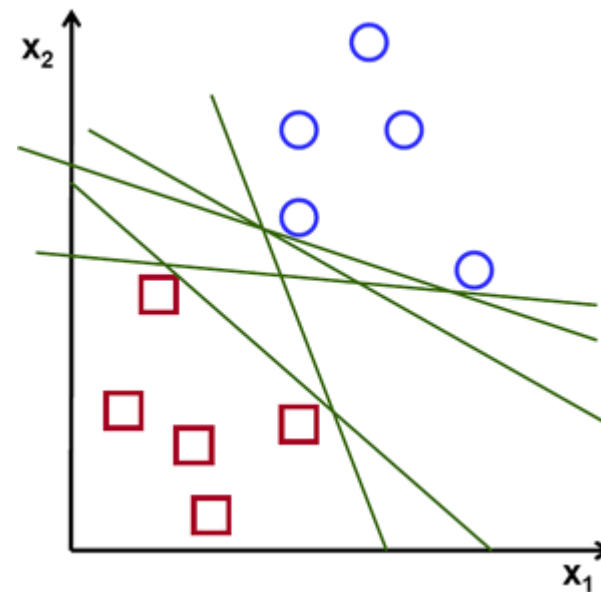
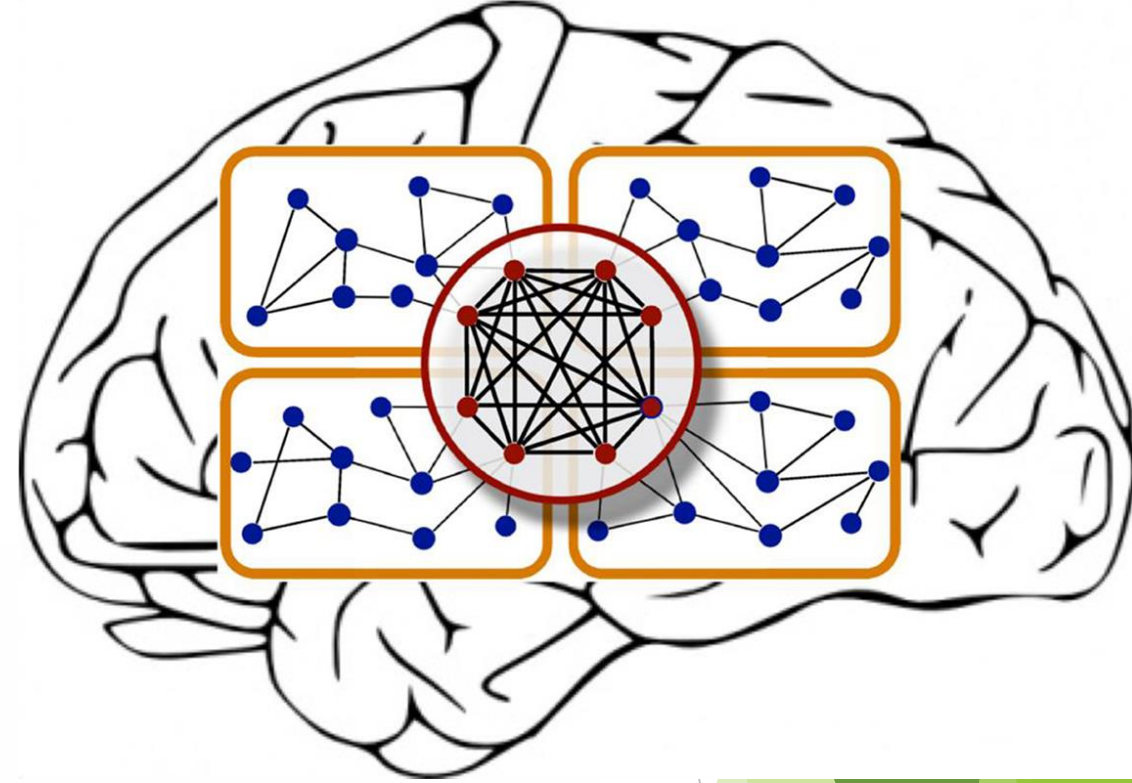
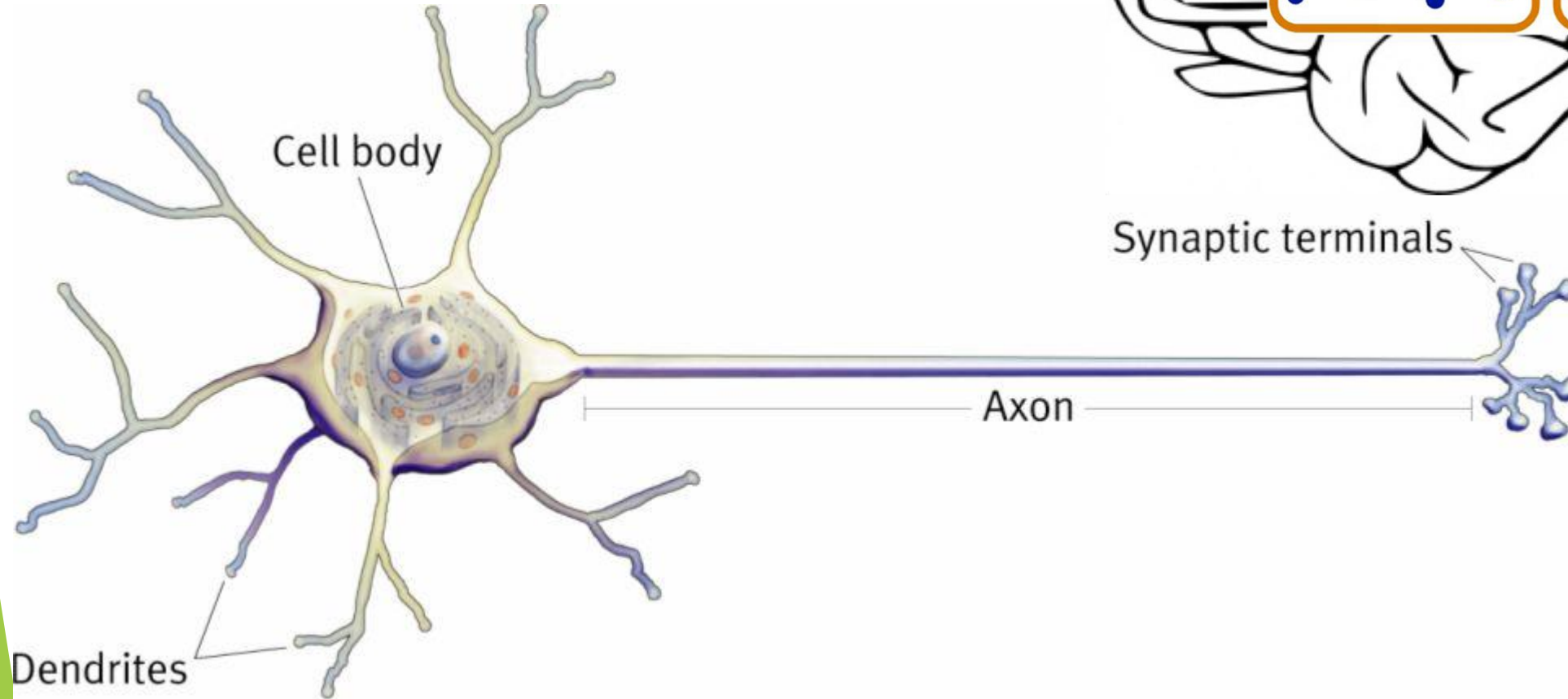


Image Classification

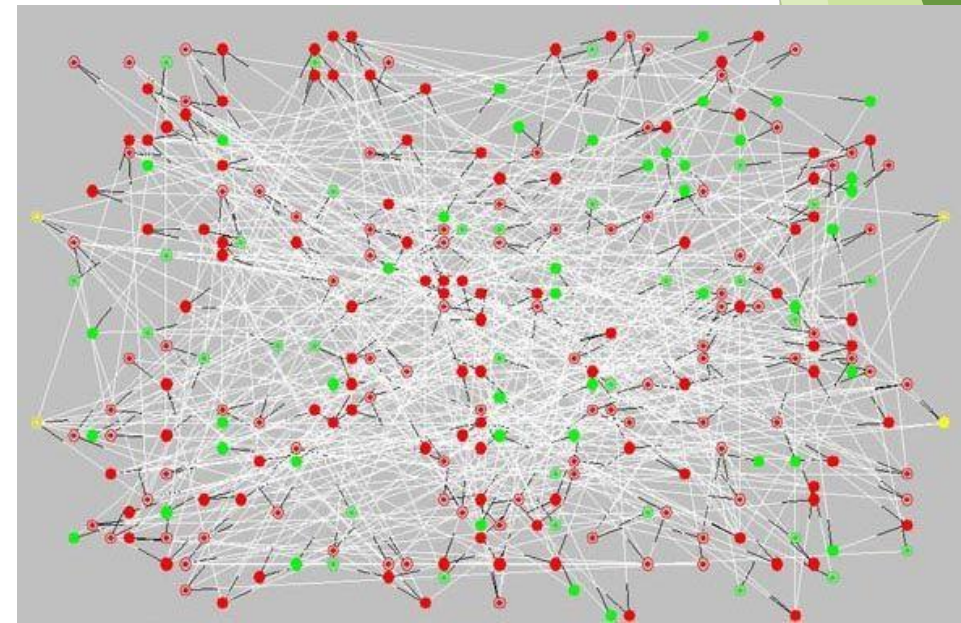
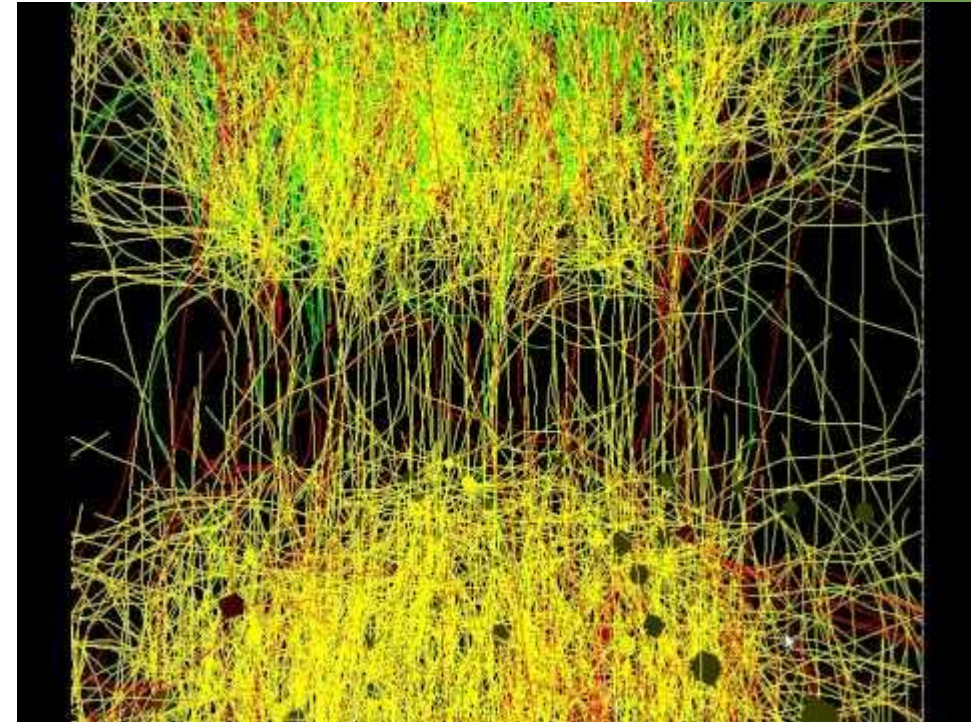
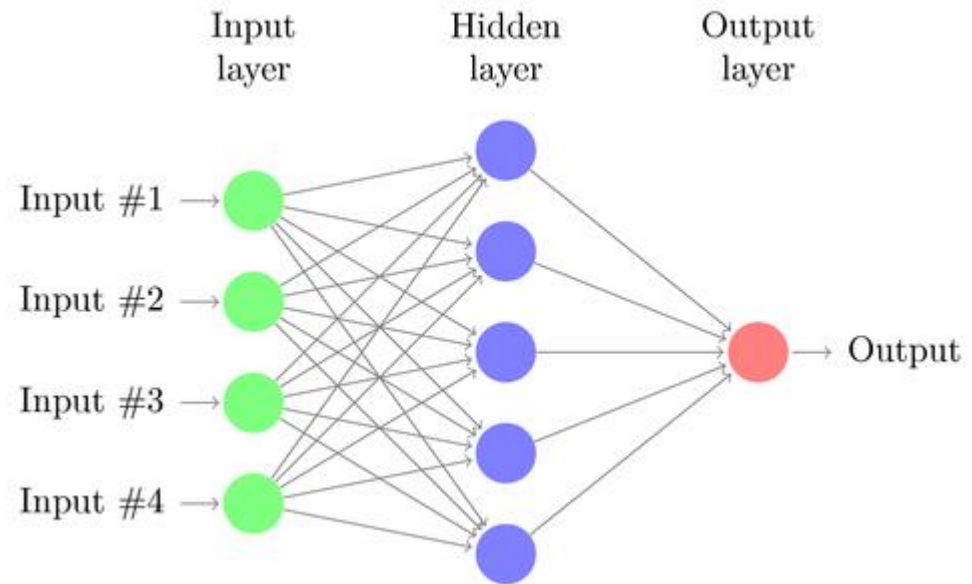
- Different classifiers
 - Neural networks
 - Support Vector Machines (SVM)
 - Boosted cascade classifiers (Adaboost)



Neural Networks



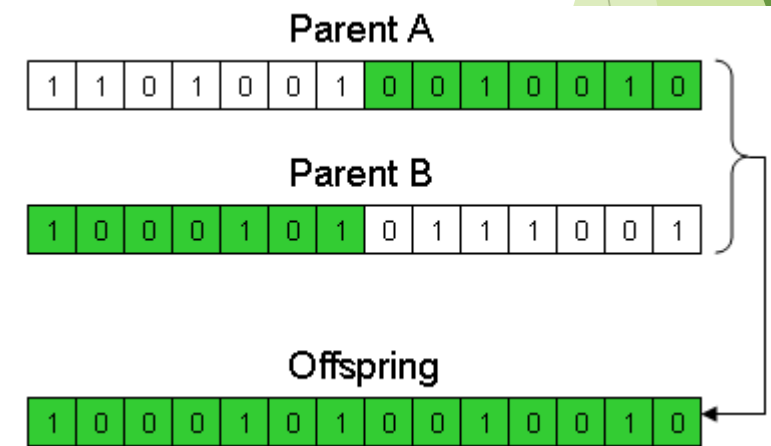
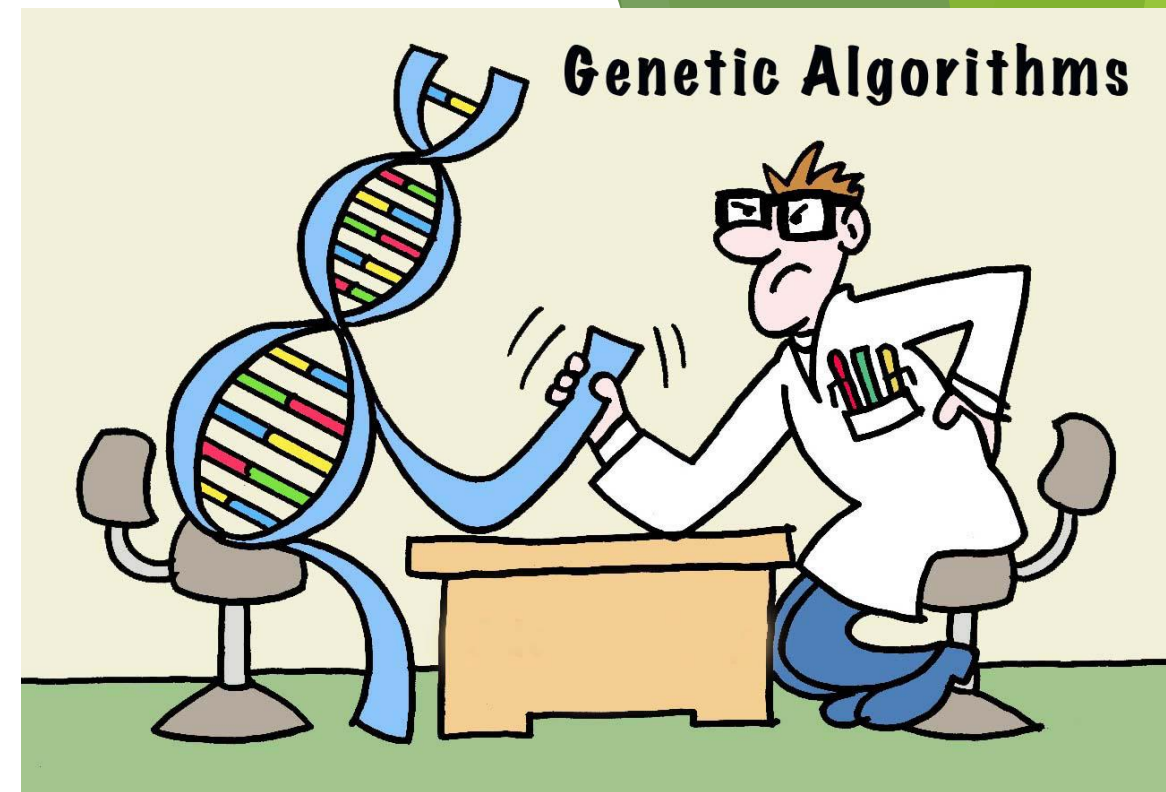
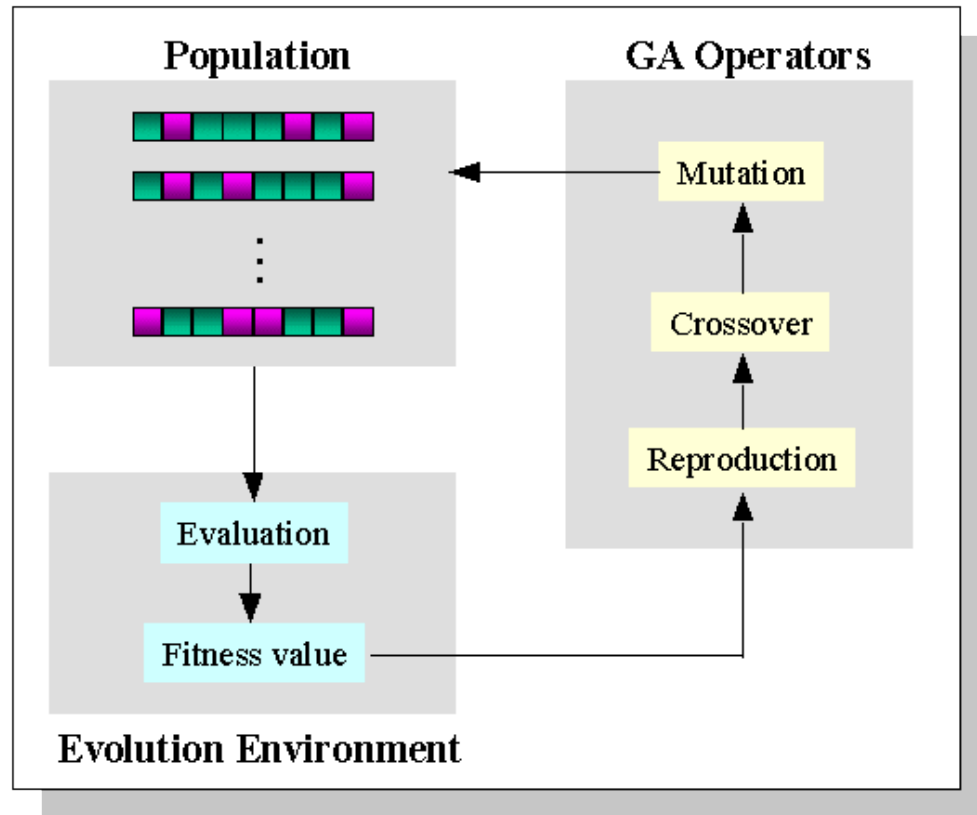
Neural Networks



What is an Artificial Neural Network?

- ▶ A large set of nodes (neurons)
 - ▶ Each neuron has input and output
 - ▶ Each node performs a simple computation
- ▶ Weighted connections between nodes
 - ▶ Connectivity determines the intelligence of the network
 - ▶ Connections have direction
 - ▶ What can be computed by the SNN is determined by connections and strengths of connections.

Genetic Algorithms



Deep Learning

- ▶ Can caption photos
 - ▶ **Human:** “A group of men playing Frisbee in the park.”
Computer model: “A group of young people playing a game of Frisbee.”



Deep Learning

- ▶ Can caption photos
 - ▶ **Human:** “A young hockey player playing in the ice rink.”
Computer model: “Two hockey players are fighting over the puck.”



Deep Learning

- ▶ Can caption photos
 - ▶ Human: “A person riding a dirt bike is covered in mud.”
 - Computer model: “A person riding a motorcycle on a dirt road.”



Deep Learning

- ▶ Can caption photos
 - ▶ **Human:** “A green monster kite soaring in a sunny sky.”
 - Computer model:** “A man flying through the air while riding a snowboard.”



Deep Learning

- ▶ Google is using Deep Learning in its image search
- ▶ Facebook is using deep learning to make inferences about our photos and about our intentions
- ▶ Qualcomm is using Deep Learning to help robots recognise objects the haven't seen before or navigate themselves to a new location
- ▶ Still a new field

Humans vs Computers

- ▶ <http://www.nytimes.com/2016/03/14/world/asia/south-korean-gets-priceless-victory-over-computer-in-go-match.html?mabReward=A1&moduleDetail=recommendations-0&action=click&contentCollection=Science®ion=Footer&module=WhatsNext&version=WhatsNext&contentID=WhatsNext&src=recg&pgtype=article>

Evolved Neural Network Art

Deep Learning - GO



<http://www.wired.com/2012/06/google-x-neural-network/>

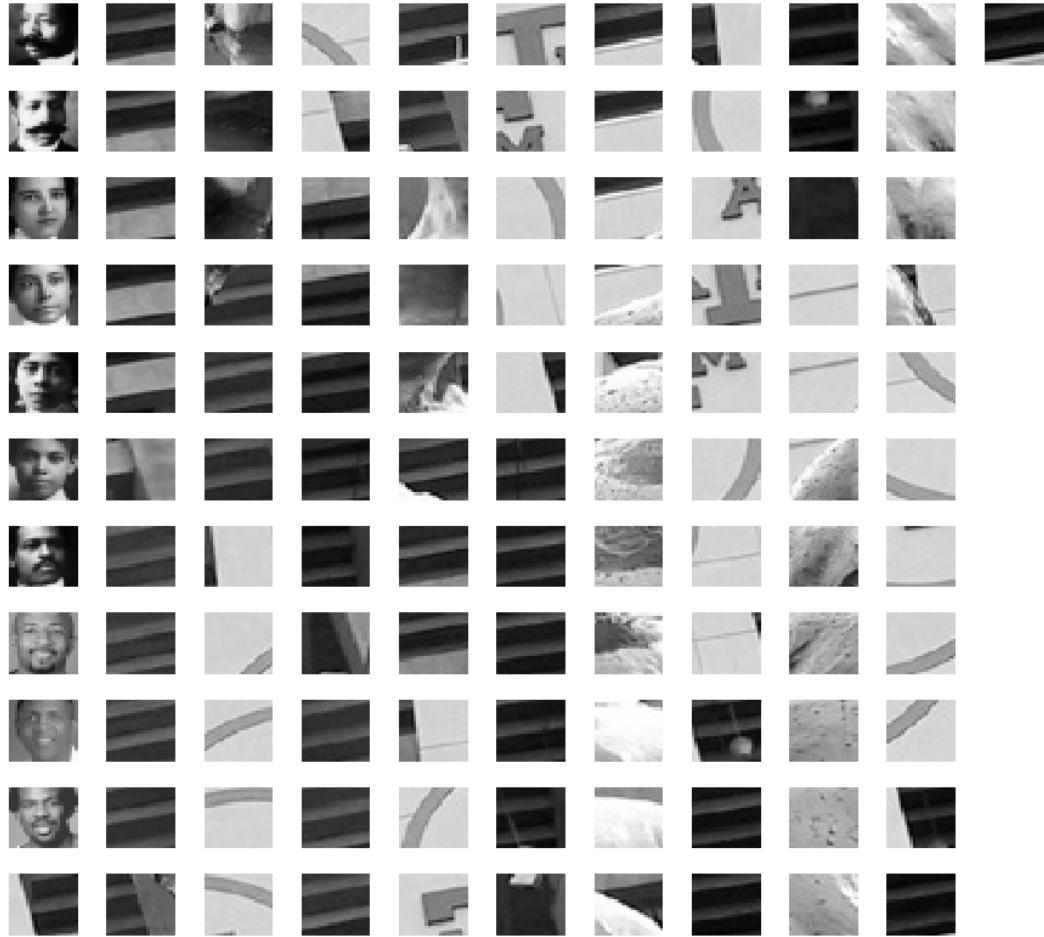
Learning to Detect Faces (Viola Jones)

- Training - Examples of 24x24 images with faces

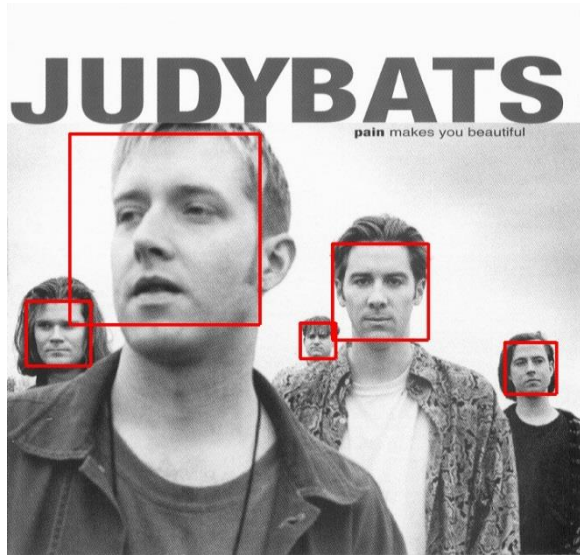


Learning to Detect Faces (Viola Jones)

► Training



Notice detection at multiple scales



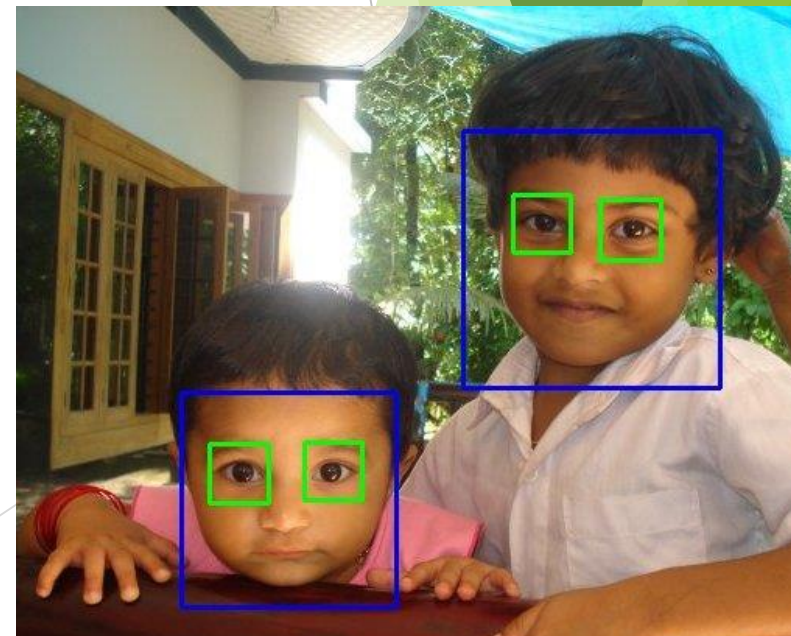
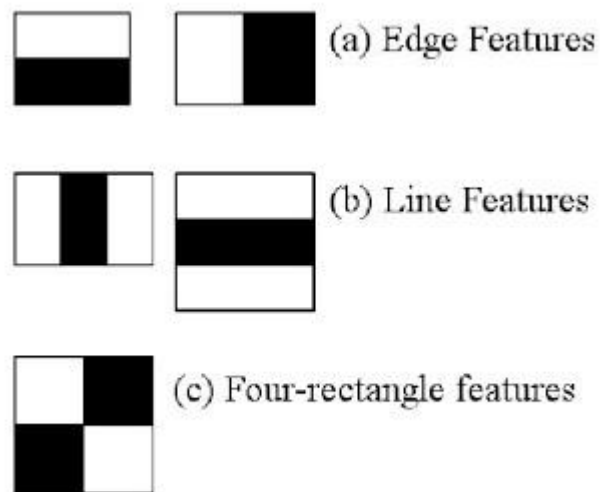
Viola Jones face detection

```
import numpy as np
import cv2

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

img = cv2.imread('sachin.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it.
- For this, haar features shown in below image are used.
- Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle.



Lab Submission

- ▶ Lab 10,11 and 12
- ▶ Python files and screenshots from labs
- ▶ Due 11th December, 2016