

Python – OpenCV

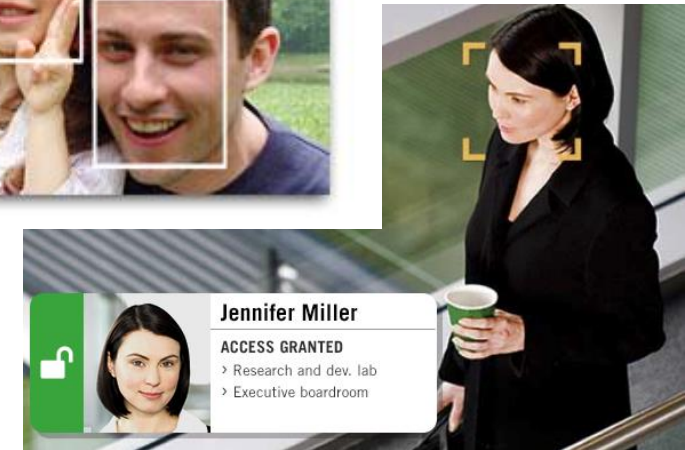
cross-platform open source computer vision library written in C and C++

Dr. Sarwan Singh
NIELIT Chandigarh

Agenda

- Introduction –History
- Overview
- Applications
- Modules
- Examples

Artificial Intelligence
Machine Learning
Deep Learning



“computer vision creates meaningful interpretation/
descriptions of objects from their images”

Computer Vision

- Computer Vision is an interdisciplinary field that deals with how computers can be made to gain a high-level understanding from digital images or videos.
- The idea here is to automate tasks that the human visual systems can do. So, a computer should be able to recognize objects such as that of a face of a human being or a lamppost or even a statue



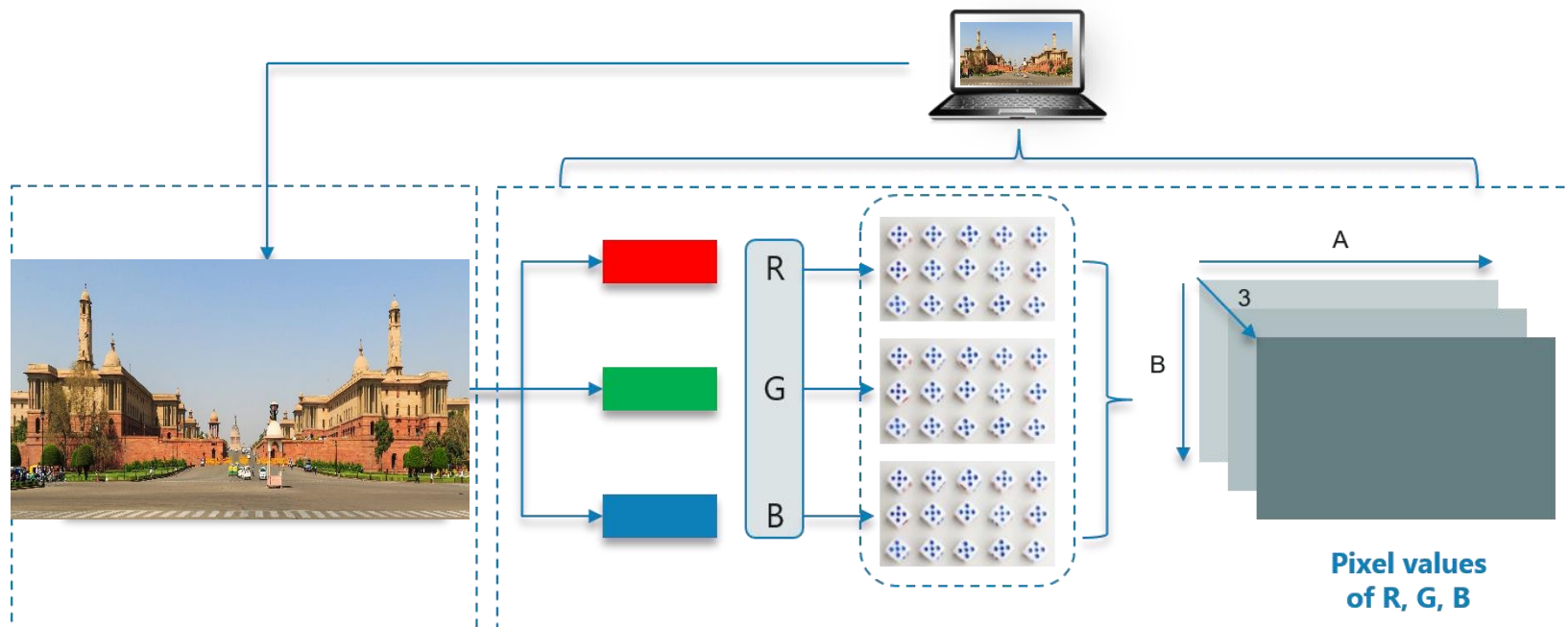
How does computer read an image

- Lets take an example of famous *Raisina Hills*
- The computer reads any image as a range of values between 0 and 255.
- For any color image, there are 3 primary channels – **Red**, **Green** and **Blue**



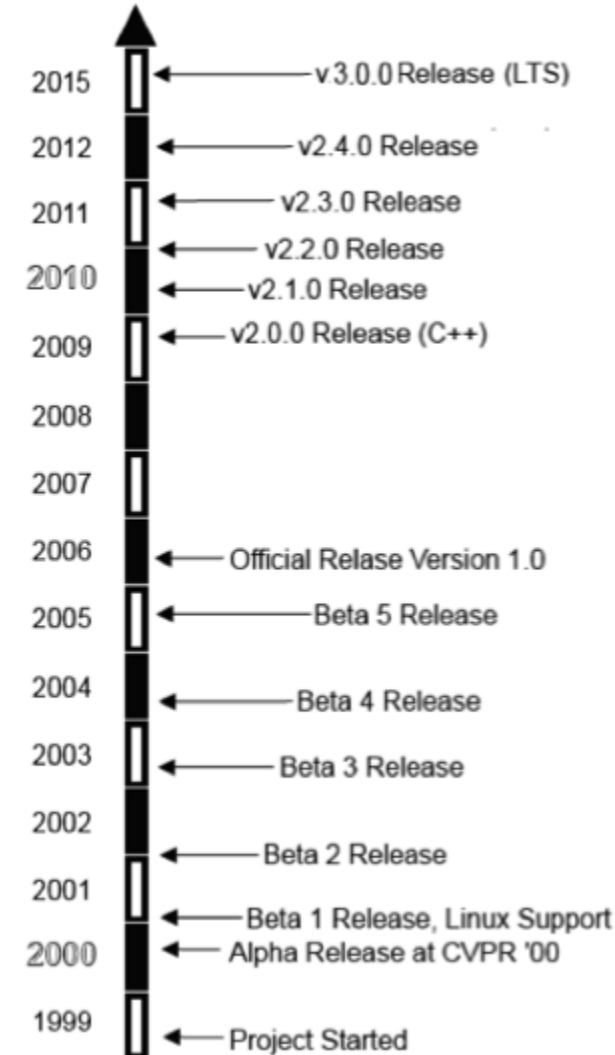
How does computer read an image

- A matrix is formed for every primary color and later these matrices combine to provide a Pixel value for the individual R, G, B colors.
- Each element of the matrices provide data pertaining to the intensity of brightness of the pixel.



OpenCV - Introduction

- ▶ OpenCV - Open Source Computer Vision Library.
- ▶ Free for commercial and research use
- ▶ founded at Intel in 1999
- ▶ now under active development, now receiving ongoing support from Willow Garage.
- ▶ It has a BSD license, 10M downloads, 500K+ lines of code.
- ▶ Cross platforms support-Linux, Windows and Mac OS.
- ▶ Portable – iPhone, Android.
- ▶ Language support – C/C++ ,Python



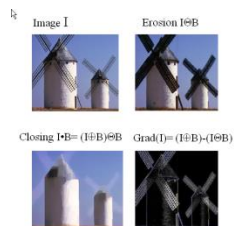
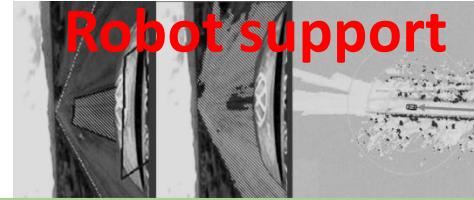
History

- OpenCV was started at Intel in 1999 by **Gary Bradsky**, and the first release came out in 2000.
- **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team.
- In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge.
- Later, its active development continued under the support of Willow Garage with Gary Bradsky and Vadim Pisarevsky leading the project.
- OpenCV now supports a multitude of algorithms related to Computer Vision and Machine Learning and is expanding day by day.
- OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc.

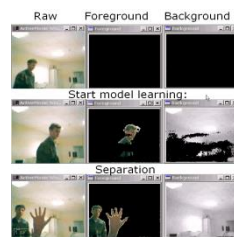
Source : <https://docs.opencv.org/>

OpenCV Overview: > 500 functions

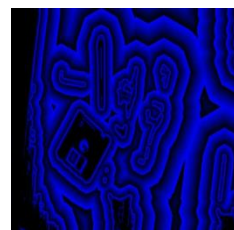
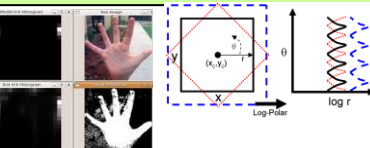
opencv.willowgarage.com



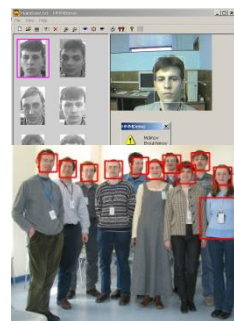
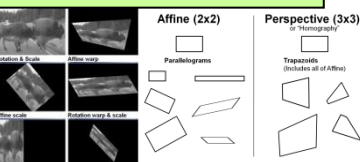
General Image Processing Functions



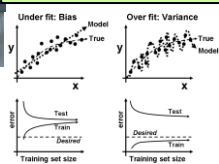
Segmentation



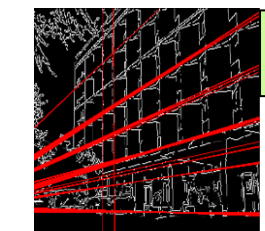
Transforms



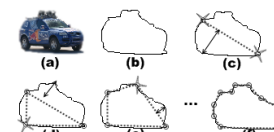
Machine Learning: •Detection, •Recognition



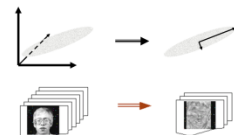
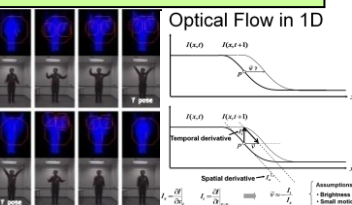
Geometric Descriptors



Features



Tracking



Matrix Math

sarwan@NIELIT

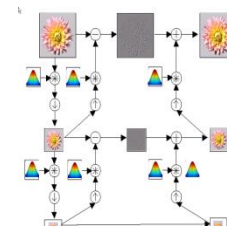
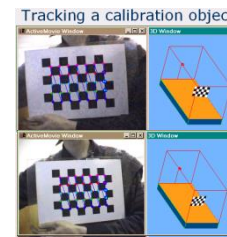
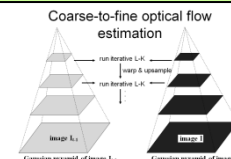
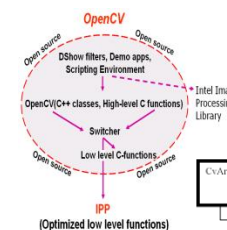


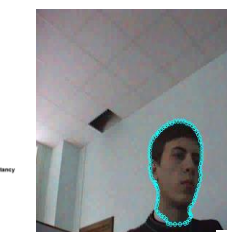
Image Pyramids



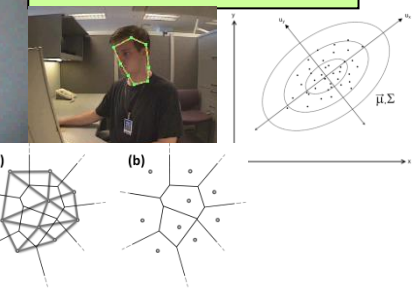
Camera Calibration, Stereo, 3D



Utilities and Data Structures



Fitting



Computer vision application

- **Robotics**

- Localization-determine robot location automatically
- Navigation
- Obstacles avoidance
- Assembly (peg-in-hole, welding, painting)
- Manipulation (e.g. PUMA robot manipulator)
- Human Robot Interaction (HRI): Intelligent robotics to interact with and serve people

- **Medicine**

- Classification and detection (e.g. lesion or cells classification and tumor detection)
- 2D/3D segmentation
- 3D human organ reconstruction (MRI or ultrasound)
- Vision-guided robotics surgery

Security

- Biometrics (iris, finger print, face recognition)
- Surveillance-detecting certain suspicious activities or behaviors

Transportation

- Autonomous vehicle
- Safety, e.g., driver vigilance monitoring

Industrial Automation Application

- Industrial inspection (defect detection)
- Assembly
- Barcode and package label reading
- Object sorting
- Document understanding (e.g. OCR)

source: tutorialspoint.com



OpenCV-Python



- OpenCV-Python is a library of Python bindings
- OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.
- OpenCV-Python makes use of **Numpy**

Modules

- openCV has a modular structure including several shared/static libraries
 - core – basic structures and algorithms
 - imgproc – image processing algorithms (image filtering, geometrical image transformations, histograms, etc.)
 - video – video analysis (such as motion estimation and object tracking)
 - highgui – built-in simple UI, in addition we use Qt
 - Calib3d – camera calibrations and 3d reconstruction
 - features2d -2D features framework (feature detectors, descriptors and descriptor matchers)
 - objdetect – detection of objects and other items (e.g. faces, eyes, etc)
 - ml – machine learning classes used for statistical classification, regression and clustering of data
 - gpu- GPU-accelerated algorithms

OpenCV functions for Reading, Showing, Writing an Image File

- **imread() function** – reading an image.
 - supports various image formats like PNG, JPEG, JPG, TIFF, etc.
- **imshow() function** – showing an image in a window.
 - The window automatically fits to the image size
 - supports various image formats like PNG, JPEG, JPG, TIFF, etc.
- **imwrite() function** – writing an image.
 - supports various image formats like PNG, JPEG, JPG, TIFF, etc.



रा.इ.सू.प्रौ.सं
NIELIT

cv2.imshow()

VS

matplotlib.show()



```
1 import cv2
2 # read and load an image
3 img = cv2.imread('kapany.jpg')
4 cv2.imshow('image_Kapany',img)
5 cv2.waitKey(0)
6 cv2.destroyAllWindows('image_Kapany')
7
8 #writing same image to some other format
9 #cv2.imwrite('image_kapany.png',img)
10
```

```
1 #Display image using cv2 and matplotlib
2 import cv2
3 import matplotlib.pyplot as plt
4 # read and load an image
5 img = cv2.imread('kapany.jpg')
6 # load image using cv2....and do processing.
7 plt.imshow(img)
8 # as opencv loads in BGR format by default, we want to show it in RGB.
9 #use following code
10 #plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
11 plt.show()
```

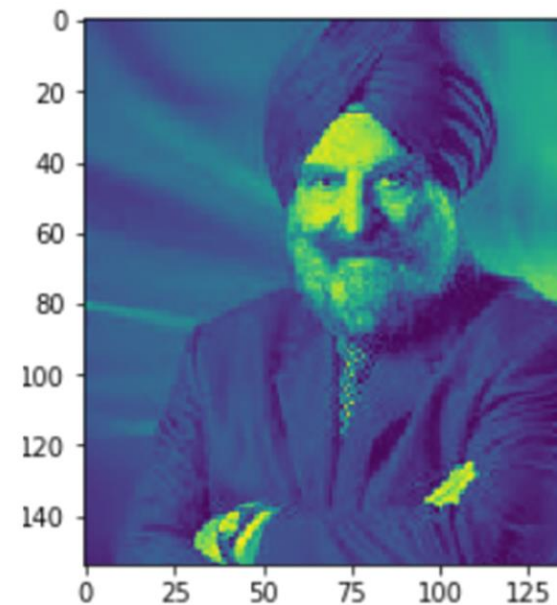


cvtColor() function to convert this image to grayscale.

```
1 plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))  
2 plt.show()
```

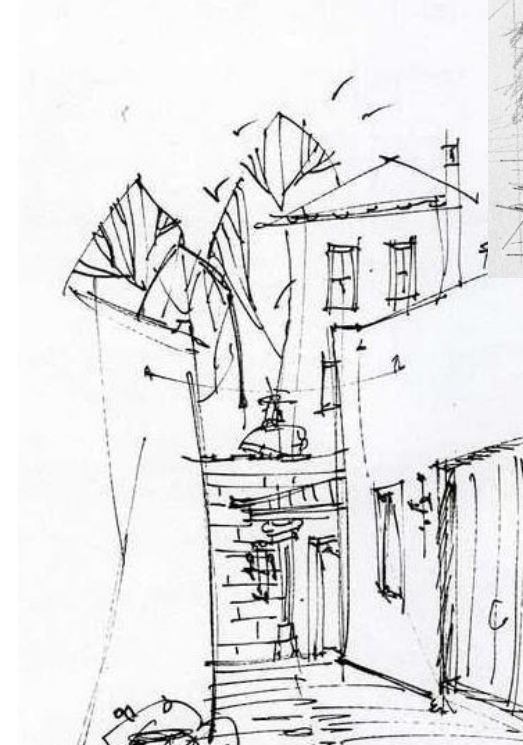
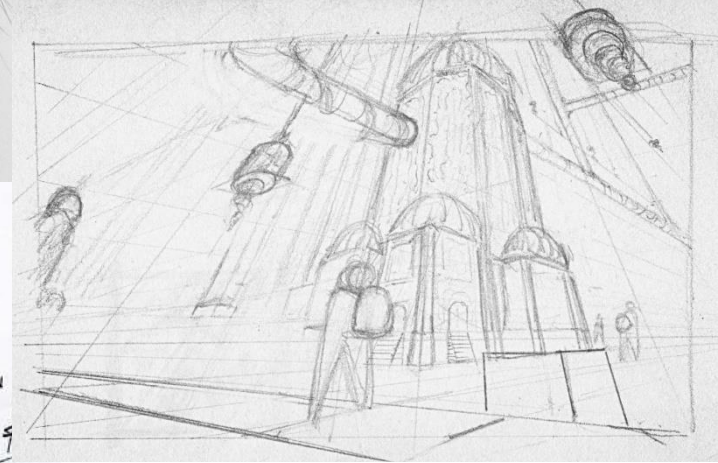


```
1 plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY))  
2 plt.show()
```



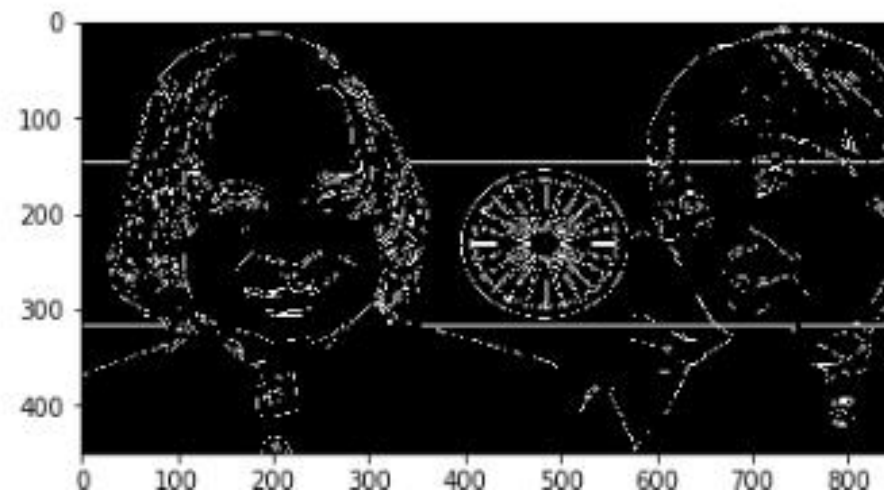
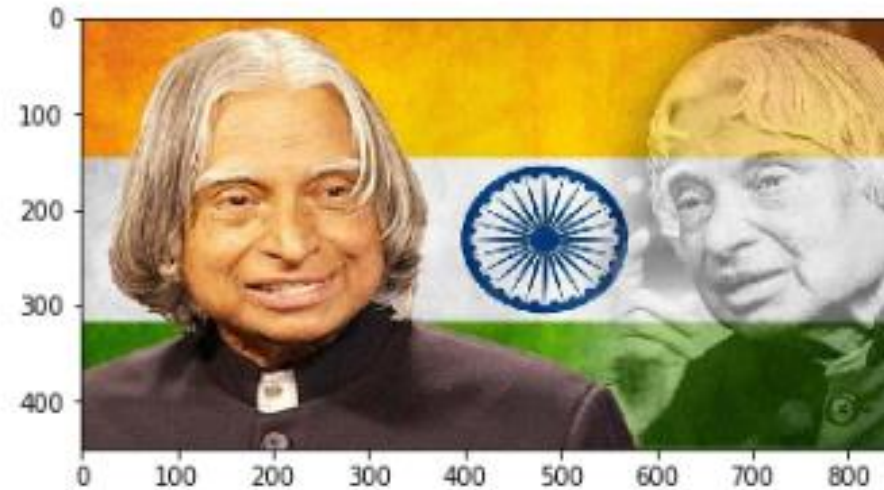
Edge Detection

- Rough sketch with edges differentiating images/objects and their poses from background can be used to identify the object easily by human eye. Same goes with computer vision or motor applications
- openCV has simple and useful function - Canny() for detecting edges.



Edge Detection

```
1 import cv2
2
3 img = cv2.imread("apj.jpg")
4 plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)), plt.show()
5
6 cv2.imwrite('edges_apj.jpg', cv2.Canny(img, 200, 300))
7 plt.imshow(cv2.cvtColor(cv2.imread('edges_apj.jpg'), cv2.COLOR_BGR2RGB))
8 plt.show()
```



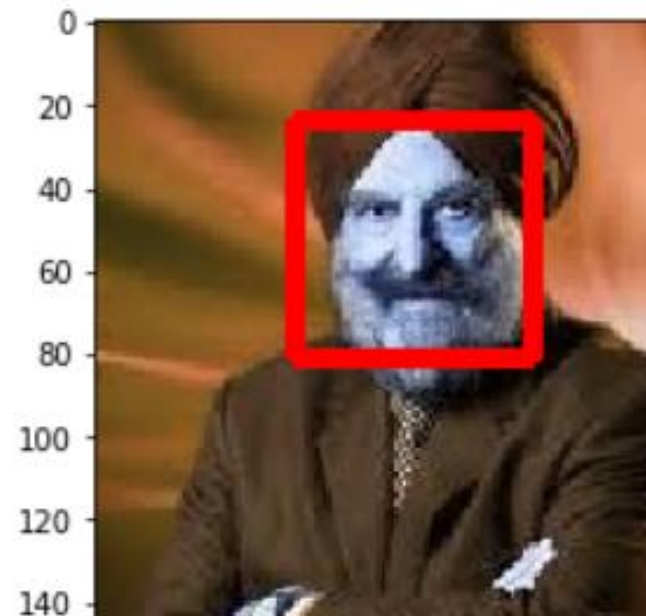


Face Detection



- one of the important and fascinating application of computer vision and brain behind automation of Things around us.
- OpenCV has built-in face detection.
- **Haar** cascade classifier for face detection

```
1 import cv2
2 img = cv2.imread("kapany.jpg")
3 face_detection= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
4 #convert it into grayscale
5 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
6 #using face_detection.detectMultiScale, perform actual face detection
7 faces = face_detection.detectMultiScale(gray, 1.3, 5)
8 for (x,y,w,h) in faces:
9     img = cv2.rectangle(img,(x,y),(x+w, y+h),(255,0,0),3)
10
11 plt.imshow(img), plt.show()
```



Eye Detection

- Prebuilt classifiers for face and eyes in OpenCV are :
 - haarcascade_frontalface_default.xml
 - haarcascade_eye.xml

```
1 #eye detection using haarcascade
2 import cv2
3 img = cv2.imread("apj.jpg")
4 eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
5
6 #convert it into grayscale
7 gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
8
9 eyes = eye_cascade.detectMultiScale(gray, 1.03, 5)
10
11 for (ex,ey,ew,eh) in eyes:
12     img = cv2.rectangle(img,(ex,ey),(ex+ew, ey+eh),(0,255,0),2)
13
14 plt.imshow(img), plt.show()
```

