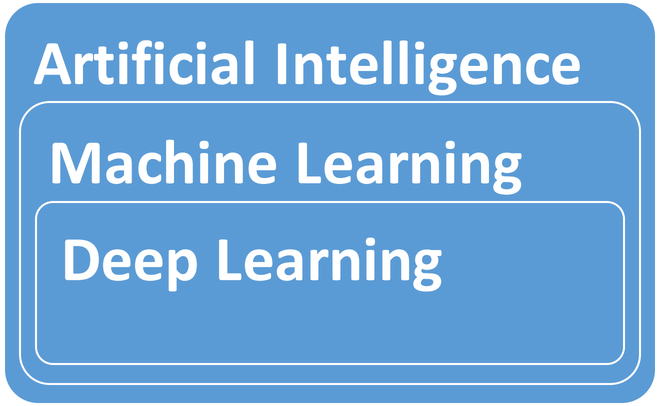
Python – OpenCV

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

Dr. Sarwan Singh

Agenda

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



Artificial Intelligence

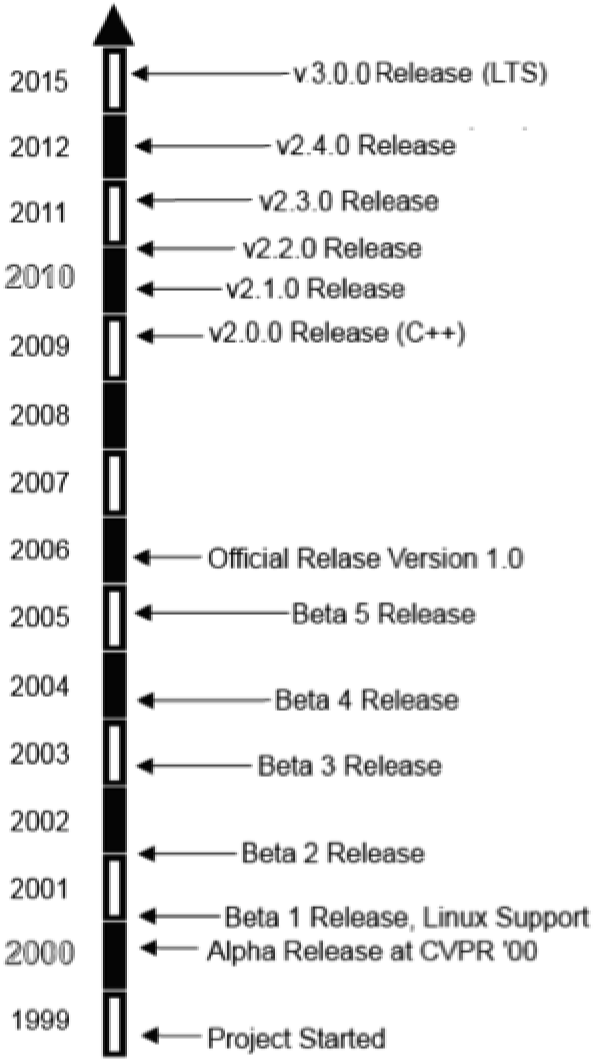
Machine Learning

Deep Learning

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

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



2015 v3.0.0 Release (LTS)
2012 v2.4.0 Release
2011 v2.3.0 Release
2010 v2.2.0 Release
v2.1.0 Release
2009 v2.0.0 Release (C++)
2008
2007
2006 Official Relase Version 1.0
2005 Beta 5 Release
2004 Beta 4 Release
2003 Beta 3 Release
2002
Beta 2 Release
2001
Beta 1 Release, Linux Support
2000 Alpha Release at CVPR '00
1999 Project Started

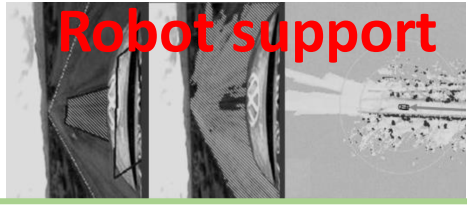
History

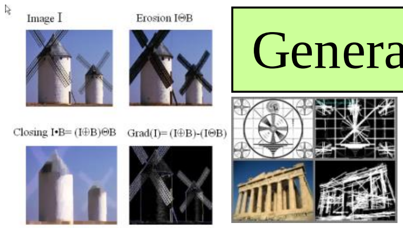
- OpenCV was started at Intel in 1999 by **Gary Bradsky**, and the first release came out in 2000.
- **Vadim Pisarevsky** joined Gary Bradsky to manage Intel's Russian software OpenCV team.
- In 2005, OpenCV was used on Stanley, the vehicle that won the 2005 DARPA Grand Challenge.
- Later, its active 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

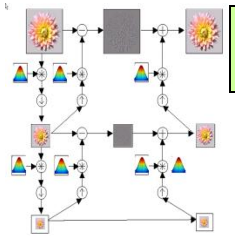
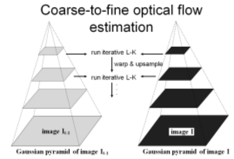
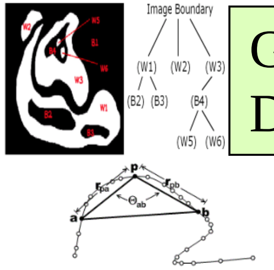
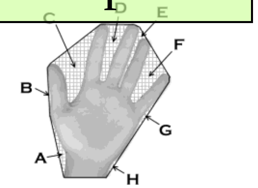


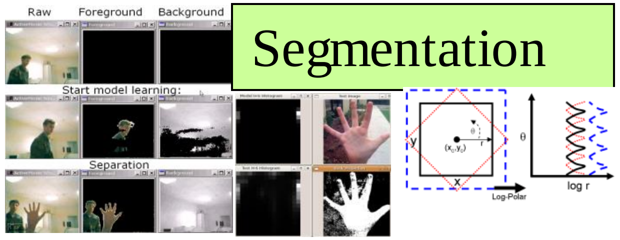
Image Pyramids


Coarse-to-fine optical flow estimation
run iterative L-K
warp & upsample
run iterative L-K
Gaussian pyramid of image I
Gaussian pyramid of image I

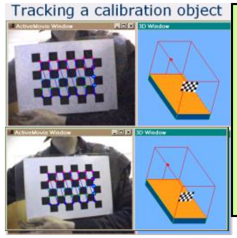


Geometric Descriptors

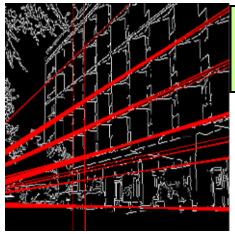




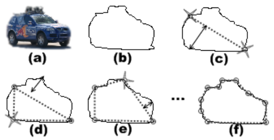
Segmentation



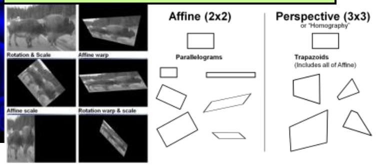
Camera Calibration, Stereo, 3D

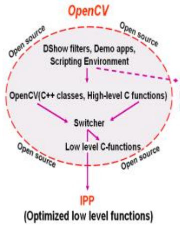


Features

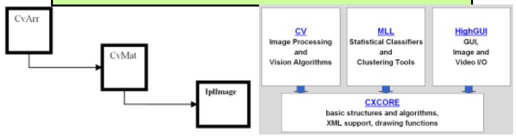


Transforms

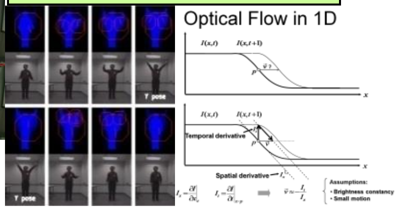


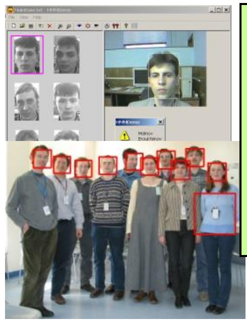

OpenCV
Open source
Open source
DShow filters, Demo apps, Scripting Environment
OpenCV(C++ classes, High-level C functions)
Intel Image Processing Library
Switcher
Low level C-functions
Open source
Open source
IPP
(Optimized low level functions)

Utilities and Data Structures


CvArr
CvMat
IplImage
CV
Image Processing and Vision Algorithms
MLL
Statistical Classifiers and Clustering Tools
HighGUI
GUI, Image and Video I/O
CXCORE
basic structures and algorithms, XML support, drawing functions

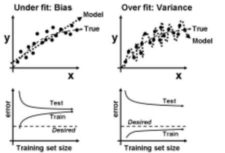
Tracking

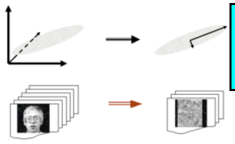




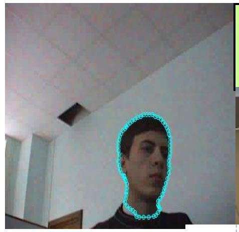
Machine Learning:

- Detection,
- Recognition

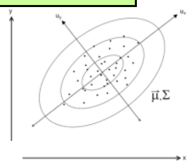


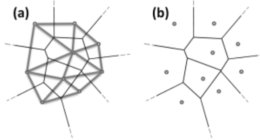


Matrix Math



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**

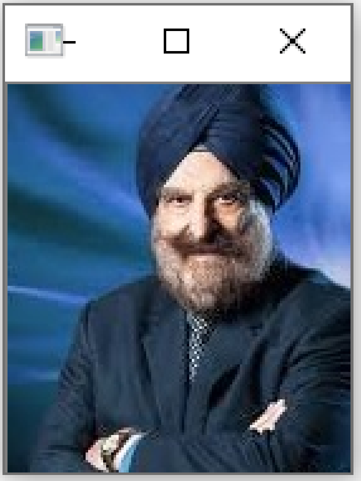
Modules

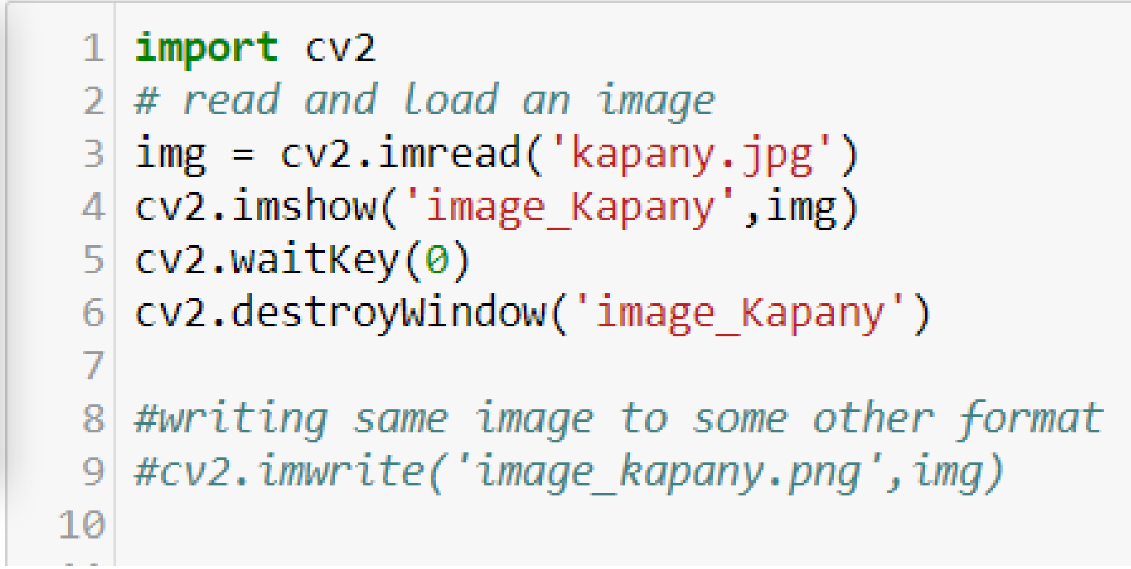
- openCV has a modular structure including several shared/static libraries
  - core – basic structures and algorithms
  - imgproc – image processing algorithms (image filtering, geometrical image transformations, histograms, etc.)
  - video – video analysis (such as motion estimation and object tracking)
  - 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

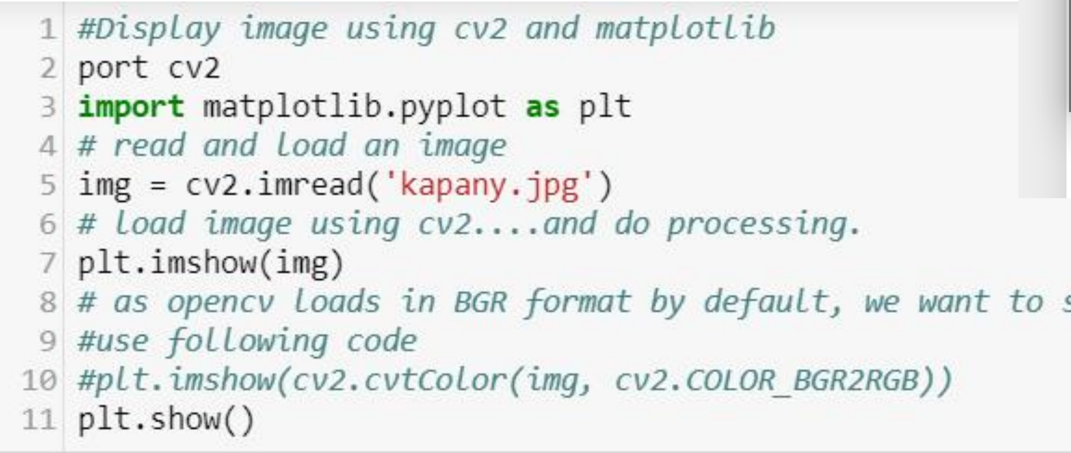
- **imread() function** − reading an image.
  - supports various image formats like PNG, JPEG, JPG, TIFF, etc.
- **imshow() function** − showing an image in a window.
  - The window automatically fits to the image size
  - supports various image formats like PNG, JPEG, JPG, TIFF, etc.
- **imwrite() function**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

cv2.imshow() vs matplotlib.show()

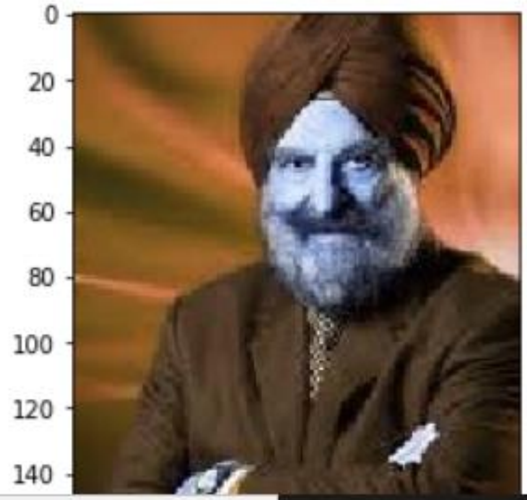




```
import cv2
# read and load an image
img = cv2.imread('kapany.jpg')
cv2.imshow('image_Kapany',img)
cv2.waitKey(0)
cv2.destroyWindow('image_Kapany')

#writing same image to some other format
#cv2.imwrite('image_kapany.png',img)

```

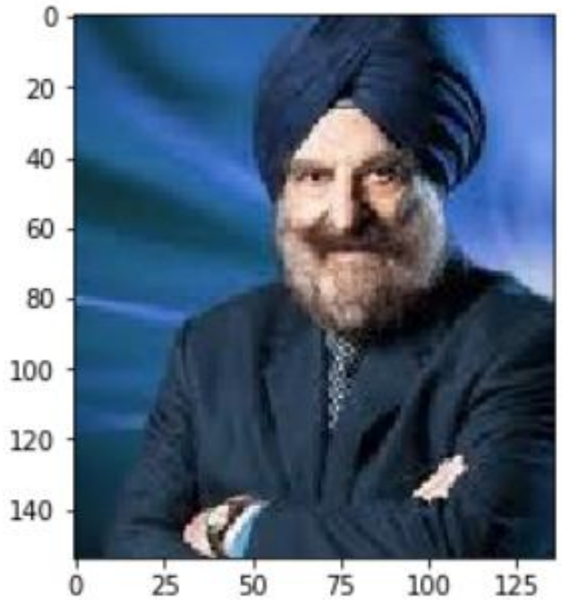


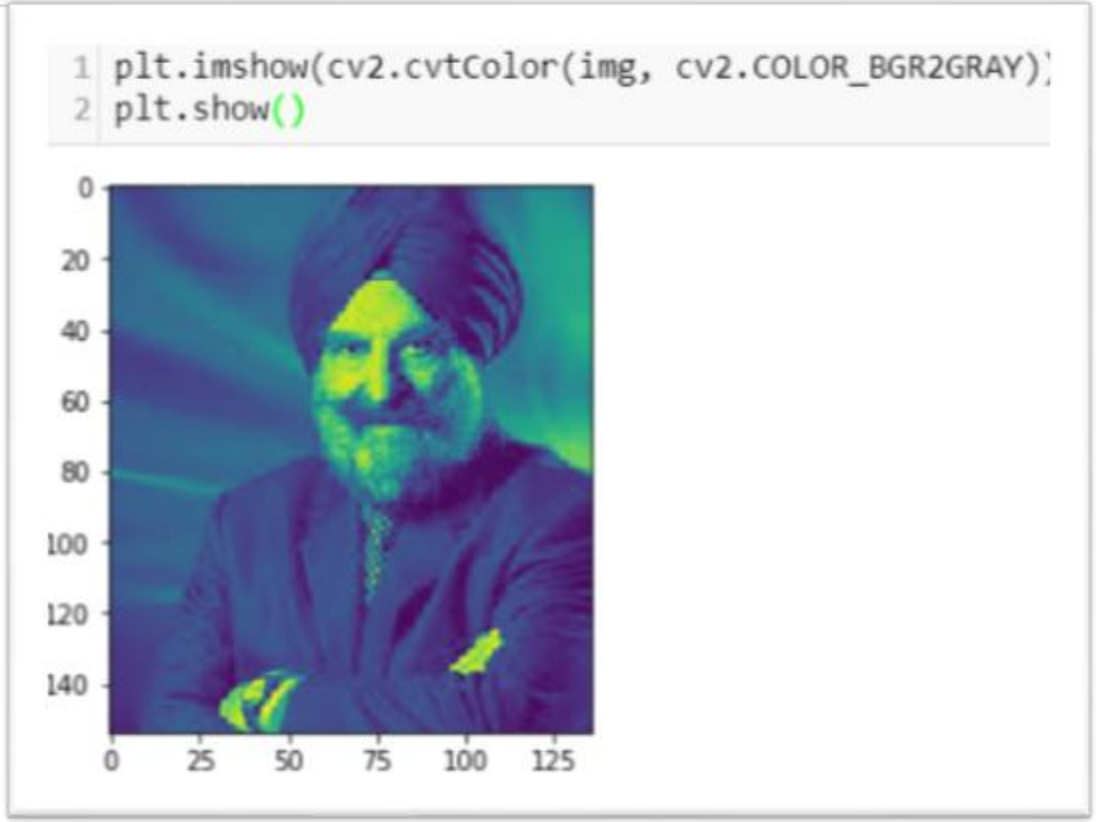
```
#Display image using cv2 and matplotlib
port cv2
import matplotlib.pyplot as plt
# read and load an image
img = cv2.imread('kapany.jpg')
# load image using cv2....and do processing.
plt.imshow(img)
# as opencv loads in BGR format by default, we want to show it in RGB.
#use following code
#plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.show()
```



cvtColor() function to convert this image to grayscale.

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



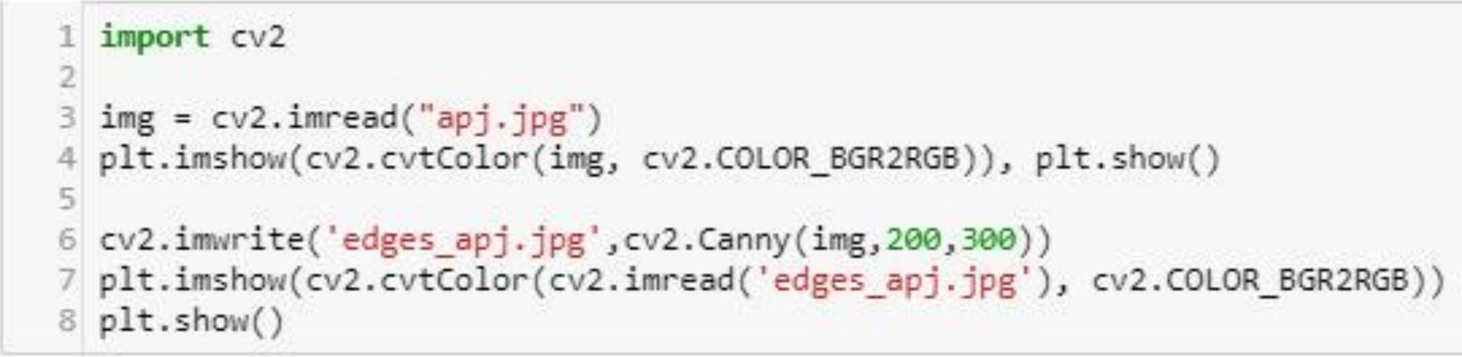


```
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2GRAY))
plt.show()
```

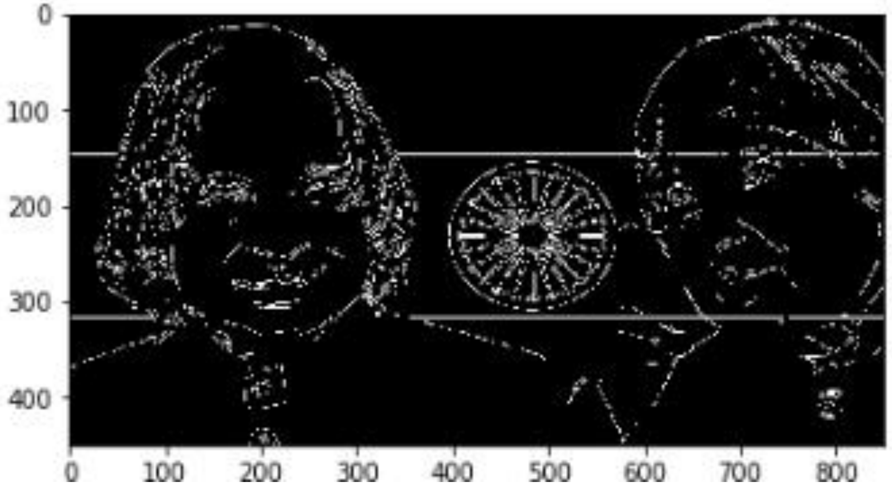
Edge Detection

- 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

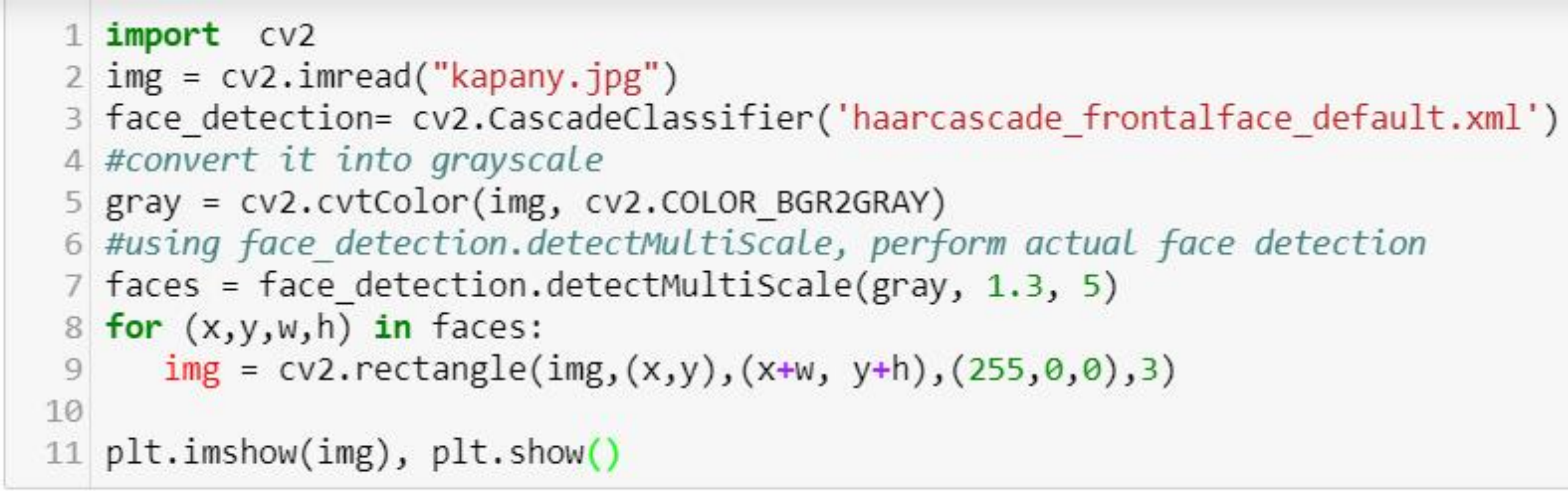


```
import cv2

img = cv2.imread("apj.jpg")
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB)), plt.show()

cv2.imwrite('edges_apj.jpg',cv2.Canny(img,200,300))
plt.imshow(cv2.cvtColor(cv2.imread('edges_apj.jpg'), cv2.COLOR_BGR2RGB))
plt.show()
```

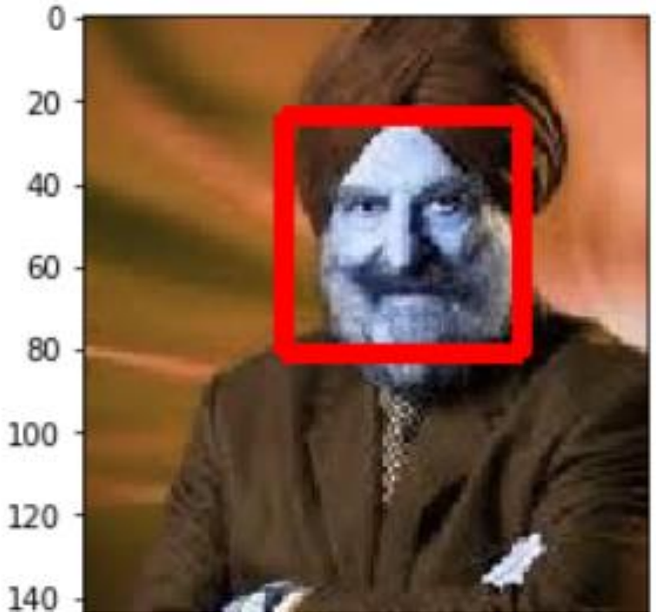


Face Detection

- 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

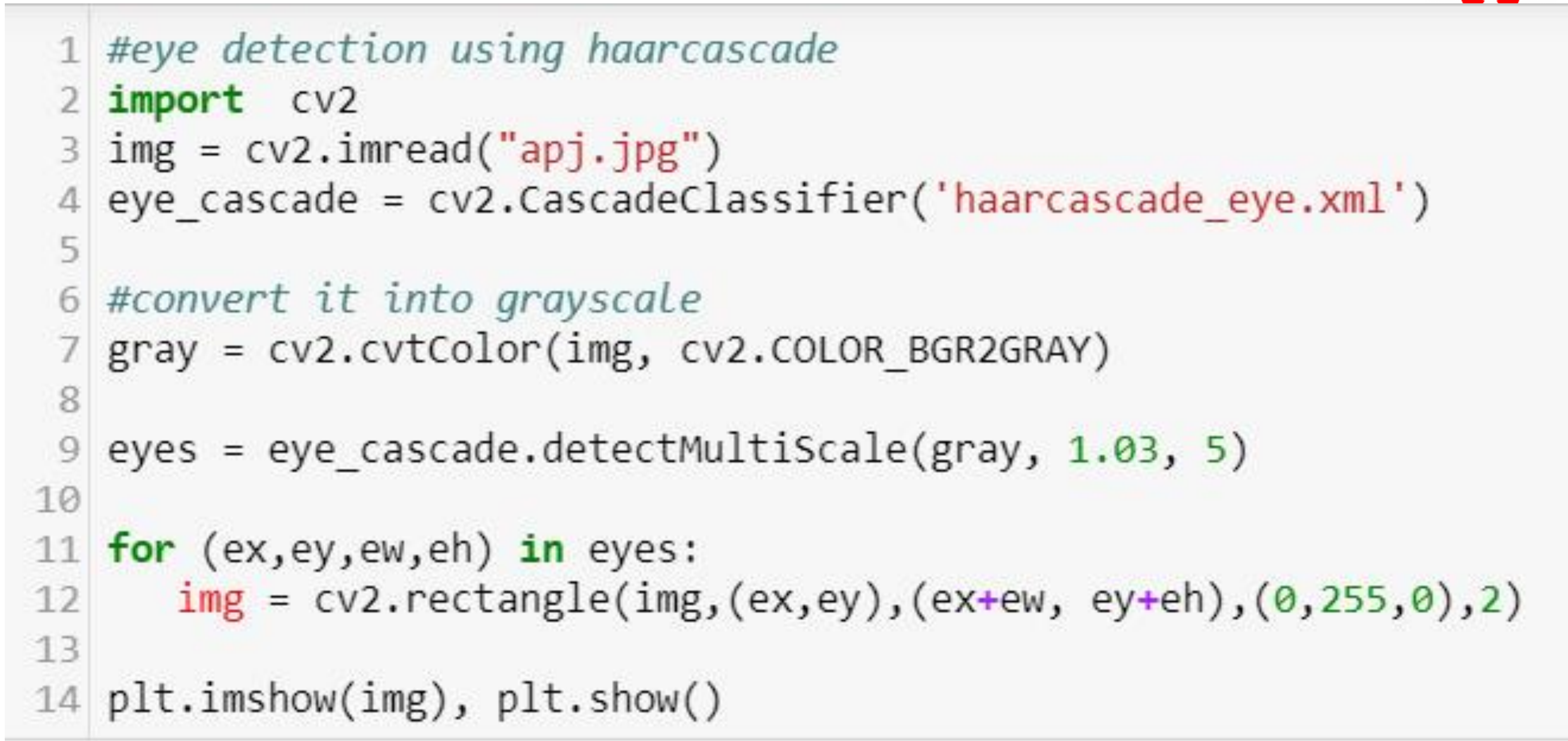


```python
import  cv2
img = cv2.imread("kapany.jpg")
face_detection= cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
#convert it into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#using face_detection.detectMultiScale, perform actual face detection
faces = face_detection.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
   img = cv2.rectangle(img,(x,y),(x+w, y+h),(255,0,0),3)

plt.imshow(img), plt.show()
```

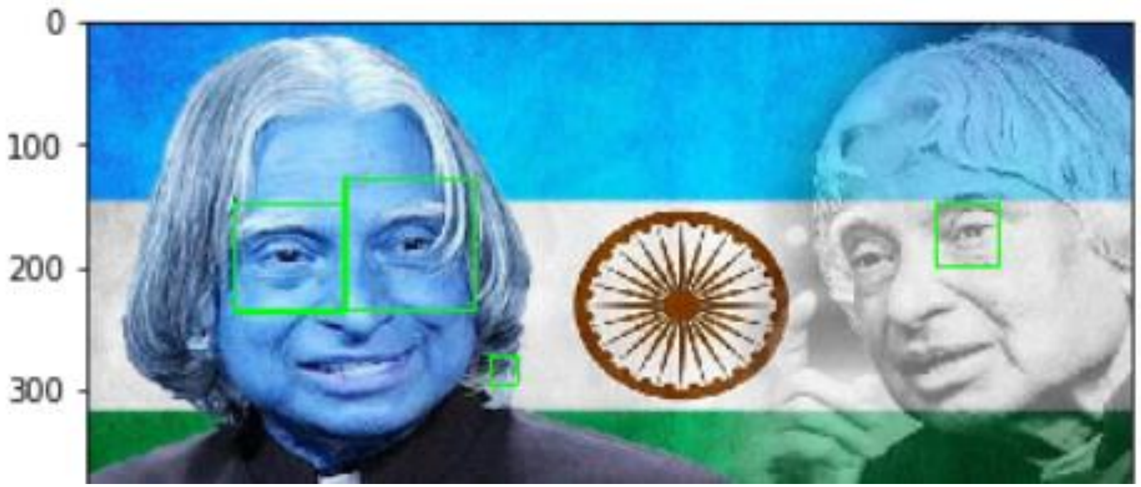


Eye Detection

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



```python
#eye detection using haarcascade
import  cv2
img = cv2.imread("apj.jpg")
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')

#convert it into grayscale
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

eyes = eye_cascade.detectMultiScale(gray, 1.03, 5)

for (ex,ey,ew,eh) in eyes:
    img = cv2.rectangle(img,(ex,ey),(ex+ew, ey+eh),(0,255,0),2)

plt.imshow(img), plt.show()
```


0
100
200
300