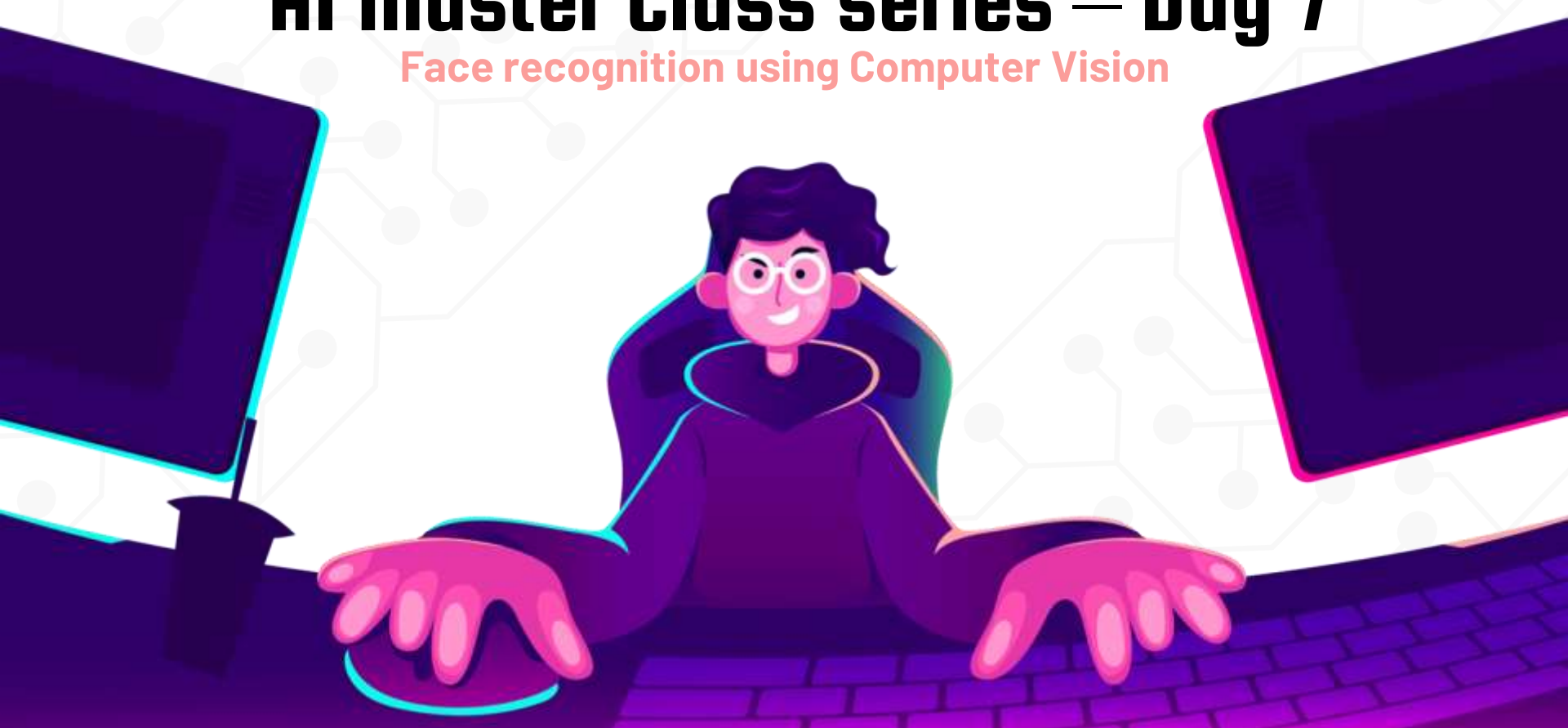
PANTECH SOLUTIONS
Technology Beyond the Dreams

AI Master Class series – Day 7

Face recognition using Computer Vision



Day-6 Agenda.

01.

Face recognition

Face recognition & its application

02.

Overview on Recognizers

LBPHFaceRecognizer &
FisherFaceRecognizer

03.

Face recognition

Creating dataset &
Face recognition

Installing Libraries

OpenCV-Contrib: pip install opencv-contrib-python

```
C:\Windows\system32>pip install opencv-contrib-python
Collecting opencv-contrib-python
  Using cached opencv_contrib_python-4.4.0.44-cp37-cp37m-win_amd64.whl (40.1 MB)
Requirement already satisfied: numpy>=1.14.5 in c:\program files (x86)\microsoft visual studio\shared\pyth
te-packages (from opencv-contrib-python) (1.19.2)
Installing collected packages: opencv-contrib-python
Successfully installed opencv-contrib-python-4.4.0.44
```

Face recognition.

Face recognition is a method of identifying or verifying the identity of an individual using their face. Face recognition systems can be used to identify people in photos, video, or in real-time.

- Preventing crime
- Unlock device
- Blind assistance
- Attendance system
- Payments

Fisherface Recognizer.

- Fisherfaces algorithm extracts principle components that separates one individual from another. So , now an individual's features can't dominate another person's features.

- Fisherface method will be applied to generate feature vector of facial image data used by system and then to match vector of traits of training image with vector characteristic of test image using euclidean distance formula

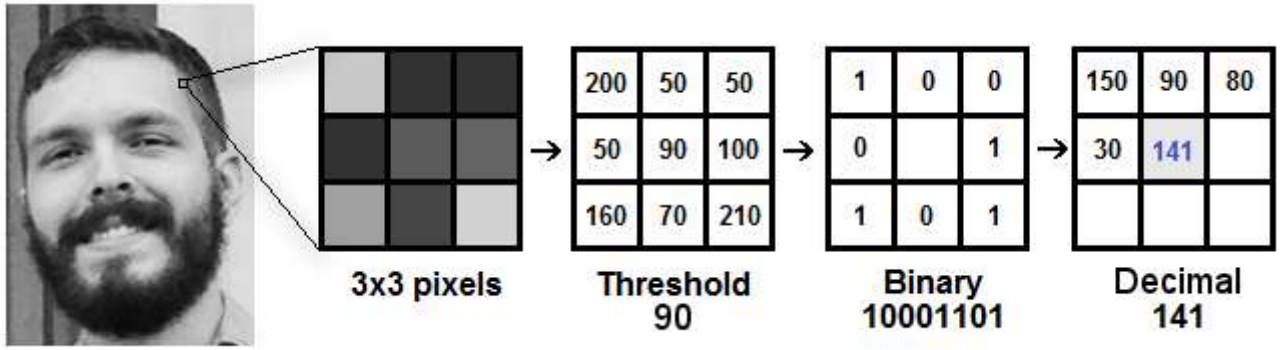
LBPHFaceRecognizer.

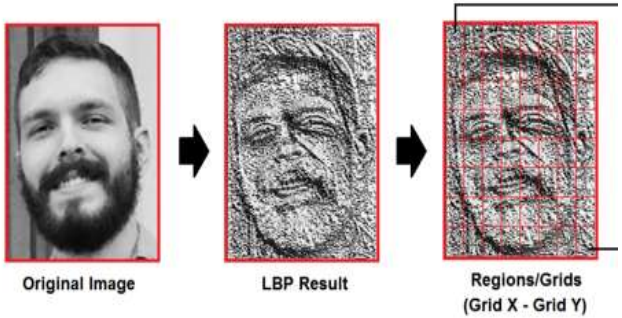
Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It doesn't look at image as a whole, but instead tries to find its local structure by comparing each pixel to its neighboring pixels.

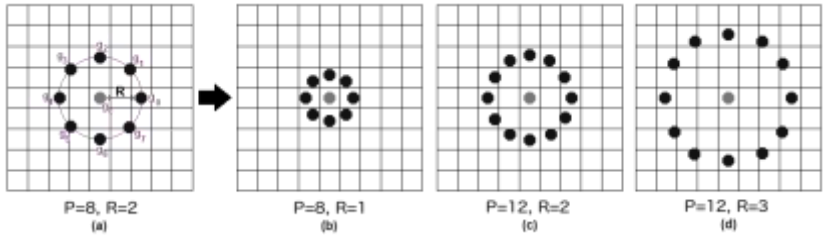
LBPH uses 4 parameters

- **Radius** - to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbors** - : The more sample points you include, the higher the computational cost. It is usually set to 8
- **X Grid** - the number of cells in the horizontal direction.
- **Y Grid** - the number of cells in the vertical direction.

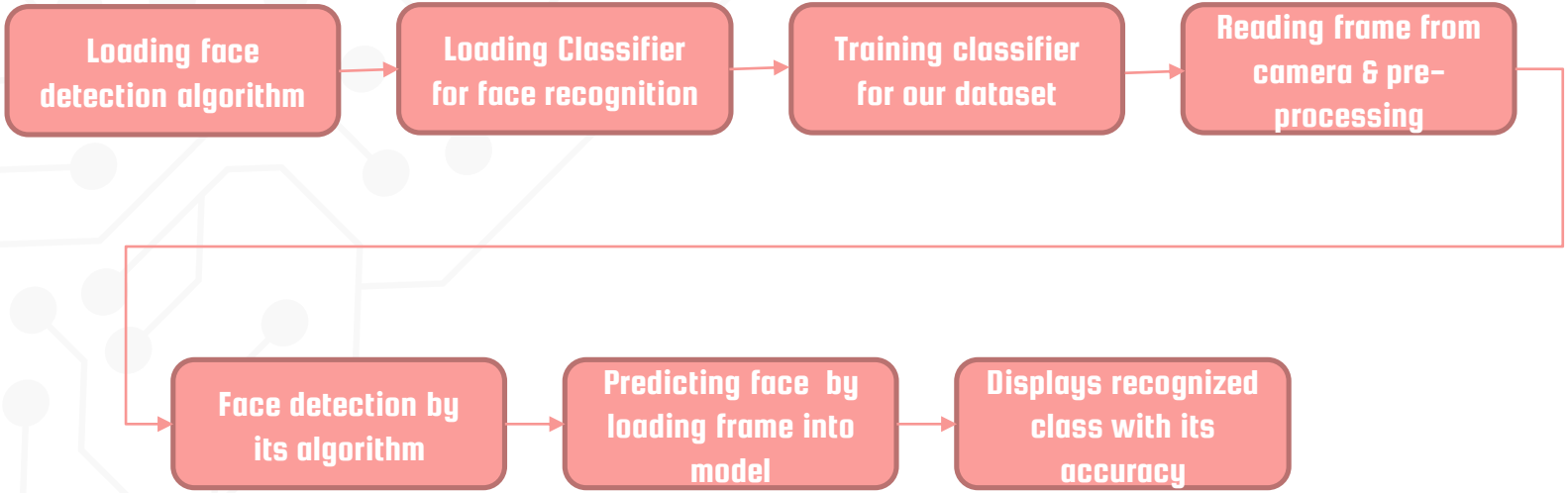




Circular LBP



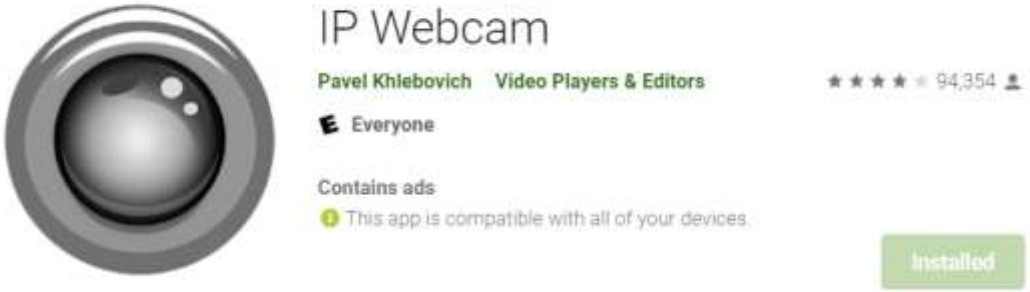
Block Diagram – Workflow of Face recognition.



Practical session

Using Android Mobile Camera for your application

IP WEBCAM



```
import urllib.request
import cv2
import numpy as np
import imutils
url='http://192.168.1.9:8888/shot.jpg'
while True:
    imgPath = urllib.request.urlopen(url)
    imgNp = np.array(bytearray(imgPath.read()), dtype=np.uint8)
    img = cv2.imdecode(imgNp, -1)
    img = imutils.resize(img, width=450)
    cv2.imshow("CameraFeed",img)
    if ord('q') ==  cv2.waitKey(1):
        exit(0)
```

Creating Dataset

Face Recognition

```
import cv2, numpy, os
size = 4
haar_file = 'haarcascade_frontalface_default.xml'
datasets = 'datasets'
print('Training...')
(images, labels, names, id) = ([ ], [ ], {}, 0)
for (subdirs, dirs, files) in os.walk(datasets):
   for subdir in dirs:
      names[ id ] = subdir
      subjectpath = os.path.join(datasets, subdir)
      for filename in os.listdir(subjectpath):
         path = subjectpath + '/' + filename
         label = id
         images.append(cv2.imread(path, 0))
         labels.append(int(label))
      id += 1
(width, height) = (130, 100)

(images, labels) = [ numpy.array(lis)for lis in [ images, labels ]]


model = cv2.face.LBPHFaceRecognizer_create()
#model =  cv2.face.FisherFaceRecognizer_create()
model.train(images, labels)

# Part 2: Use fisherRecognizer on camera stream
face_cascade = cv2.CascadeClassifier(haar_file)
webcam = cv2.VideoCapture(0)
cnt=0
```

```
while True:
    (_, im) = webcam.read()
    #im = imutils.resize(im, width=200)
    gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(im,(x,y),(x+w,y+h),(255,255,0),2)
        face = gray[ y:y + h, x:x + w ]
        face_resize = cv2.resize(face, (width, height))
        #Try to recognize the face
        prediction = model.predict(face_resize)
        cv2.rectangle(im, (x, y), (x + w, y + h), (0, 255, 0), 3)
        if prediction[1]<800:
            cv2.putText(im,'%s - %.0f' % (names[ prediction[0] ],prediction[1]),(x-10, y-10), cv2.FONT_HERSHEY_PLAIN,1,(255, 0, 0))
            print (names[ prediction[0] ])
            cnt=0
        else:
            cnt+=1
            cv2.putText(im,'Unknown',(x-10, y-10), cv2.FONT_HERSHEY_PLAIN,1,(0, 255, 0))
            if(cnt>100):
                print("Unknown Person")
                cv2.imwrite("input.jpg",im)
                cnt=0
    cv2.imshow('OpenCV', im)
    key = cv2.waitKey(10)
    if key == 27:
        break

webcam.release()
cv2.destroyAllWindows()
```

AI News – Day 7.

Sept – 2020

These Robots Use AI to Learn How to Clean Your House

At Toyota, researchers are experimenting with prototypes that swoop from the ceiling to take care of chores with the help of machine learning.

- The cleaning robots live inside a mock home located at the Toyota Research Institute in Los Altos, California. The institute's researchers are testing a range of robot technologies designed to help finally realize the dream of a home robot.

- Home, a robot suspended from the ceiling slowly expands arms holding a sponge, before carefully wiping a kitchen surface clean. Nearby, another robot gently cleans a flat-screen television, causing it to wobble slightly.

Thanks!

Do you have any questions?
sanjay@pantechmail.com

www.pantechsolutions.net

Tomorrow session

Face Emotion using OpenCV