**Abstract**

Automation of day to day tasks has become a lot easier thanks to many researches in the field of Artificial Intelligence. There are many ways to implement these concepts. One such is Image processing. We used image processing in order to find whether a person in the image is wearing spectacles. The application is able to detect a person’s face, normalize it and classify whether he or she is wearing spectacles or not using the images from the live feed. The results are pretty accurate and needed only minor tweaking depending on the input device.

**Related Existing Resources**

Although there are many researches going on in the field of image recognition, there aren’t as many for detection of spectacles.

‘Glasses Detection on Real Images Based on Robust Alignment’ and ‘Novel Method for Eyeglasses Detection in Frontal Face Images’ are the papers that I read through and decided to implement. The first paper pointed out how we need to first align the face using the facial landmarks like mouth nose and eyes. And then normalize the face and only use the specific parts that are necessary for detection of the eye glasses. The second paper showed how the most accurate and consistent results were found when we used edge detection in region below our temple, between the eyes.

I also needed a face recognition algorithm and hence ended up using the implementation of ‘One Millisecond Face Alignment with Ensemble of Regression Trees by Vahid Kazemi and Josephine Sullivan, CVPR20’ model for finding face landmarks.

**Implementation**

import datetime

import pytz

import cv2

import numpy as np

import pandas as pd

import math

import time

import datetime

import warnings

from mlxtend.image import extract\_face\_landmarks

pic\_info = {'Year':[],'Month':[],'Day':[],'Hour':[],'Minute':[],'Second':[],'Microsecond':[],'Timezone':[],'isWearingSpecs':''}

df = pd.DataFrame(pic\_info,columns=['Year','Month','Day','Hour','Minute','Second','Microsecond','Timezone','isWearingSpecs'])

#If we want to append data instead of creating new datasets each time

#df = pd.read\_csv("pic\_info.csv")

warnings.filterwarnings('error','No face detected.')

url = "http://192.168.43.1:8080/video"

cap = cv2.VideoCapture(0)

wi = 1

ni = 1

while True:

    return\_value,frame = cap.read()

    # Resize frame of video to 1/4 size for faster face recognition processing

    small\_frame = cv2.resize(frame, (0, 0), fx=0.5, fy=0.5)

    process\_frame = cv2.resize(frame, (0, 0), fx=1, fy=1)

    cv2.imshow('test',small\_frame)

    # Hit 'q' on the keyboard to quit!

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

    img = cv2.cvtColor(process\_frame, cv2.COLOR\_BGR2RGB)

    try:

        landmarks = extract\_face\_landmarks(img)

        #Individual facial features

        left\_outer\_eye = landmarks[36]

        right\_outer\_eye = landmarks[45]

        #We will use trigonometry to find roatation angle required to normalize the face

        h = left\_outer\_eye[1]-right\_outer\_eye[1] #Height of the triangle

        w = left\_outer\_eye[0]-right\_outer\_eye[0] #Width of the triangle

        angle = math.degrees(math.atan(h/w)) #Finding tan inverse and converting to degress for angle of rotation

        image\_center = tuple(np.array(img.shape[1::-1])/2)

        rot\_mat = cv2.getRotationMatrix2D(image\_center, angle, 1.0)

        rotated\_img = cv2.warpAffine(img, rot\_mat, img.shape[1::-1], flags=cv2.INTER\_LINEAR)

        landmarks = extract\_face\_landmarks(rotated\_img)

    except UserWarning:

        continue

    img = rotated\_img[landmarks[24][1]:landmarks[28][1],landmarks[36][0]:landmarks[45][0]] #Area to crop

    edges = cv2.Canny(img,100,100)

    h,w = edges.shape

    #Extracting coordinates from edges

    indices = np.where(edges != [0])

    coordinates =  list(zip(indices[1], indices[0]))

    #Declaring iterator and list

    i = 0

    count = []

    for x in range(int(w\*0.45),int(w\*0.55)):

        count.append(0)

        for y in range(0,h):

            if (x,y) in coordinates:

                count[i] += 1

        i += 1

    #print(count)

    flag = 1

    cur\_time = datetime.datetime.now(tz=pytz.timezone('Asia/Kolkata'))

    for x in count:

        if x<2:

            flag = 0

            print("Spectacles not Detected\n")

            df = df.append({'Year':cur\_time.year,'Month':cur\_time.month,'Day':cur\_time.day,'Hour':cur\_time.hour,

            'Minute':cur\_time.minute,'Second':cur\_time.second,'Microsecond':cur\_time.microsecond,'Timezone':cur\_time.tzinfo,

            'isWearingSpecs':'No'}, ignore\_index = True)

            if(ni==21):

                ni=1

            cv2.imwrite(filename = 'NotWearing\ '+str(ni)+'.jpg', img = frame)

            ni+=1

            break

    if (flag==1):

        print("The image has spectacles\n")

        df = df.append({'Year':cur\_time.year,'Month':cur\_time.month,'Day':cur\_time.day,'Hour':cur\_time.hour,

        'Minute':cur\_time.minute,'Second':cur\_time.second,'Microsecond':cur\_time.microsecond,'Timezone':cur\_time.tzinfo,

        'isWearingSpecs':'Yes'}, ignore\_index = True)

        if(wi==21):

            wi=1

        cv2.imwrite(filename = 'Wearing\ '+str(wi)+'.jpg', img = frame)

        wi+=1

    #time.sleep(1)

df.to\_csv(r'picture\_info.csv', index = False, header = True)

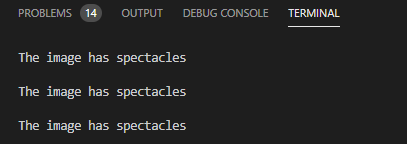
print(df)

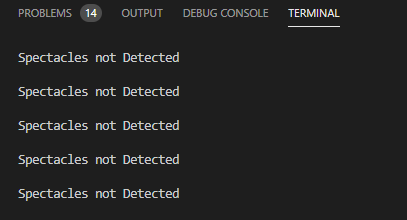
cap.release()

cv2.destroyAllWindows()

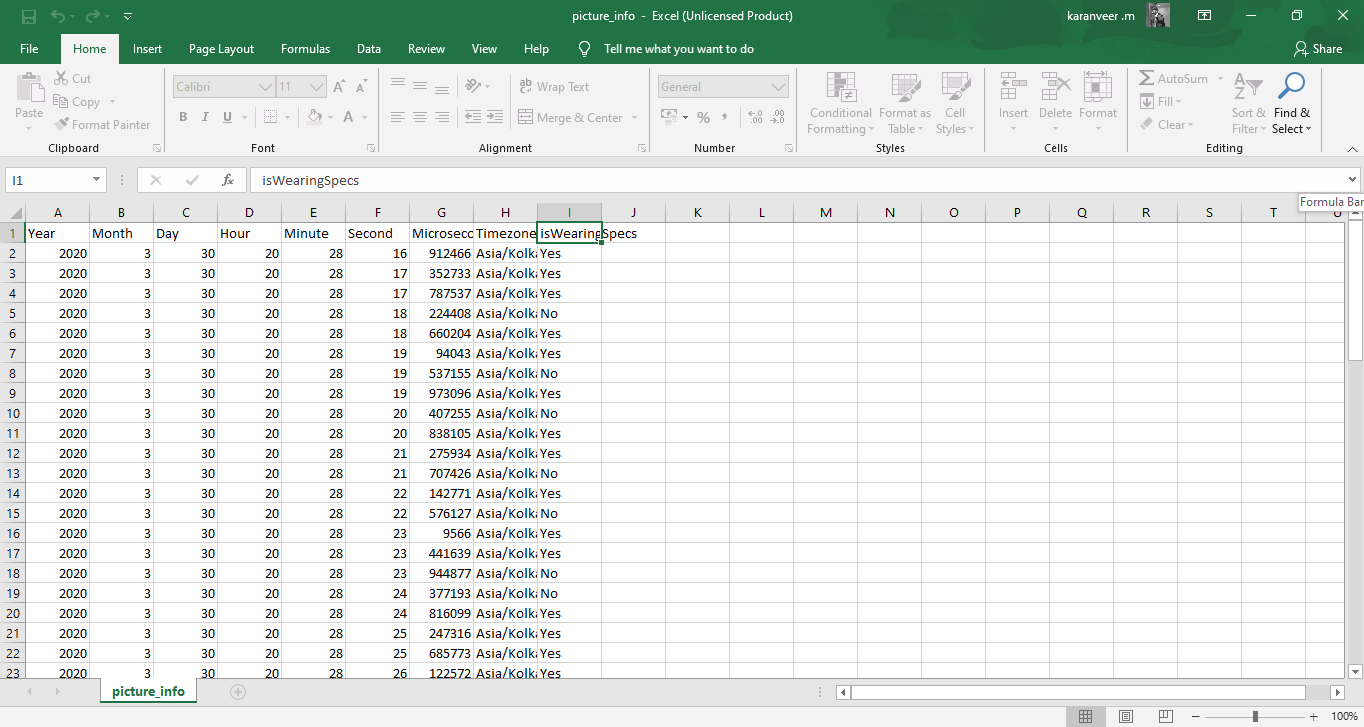
**Result**

* Terminal Output

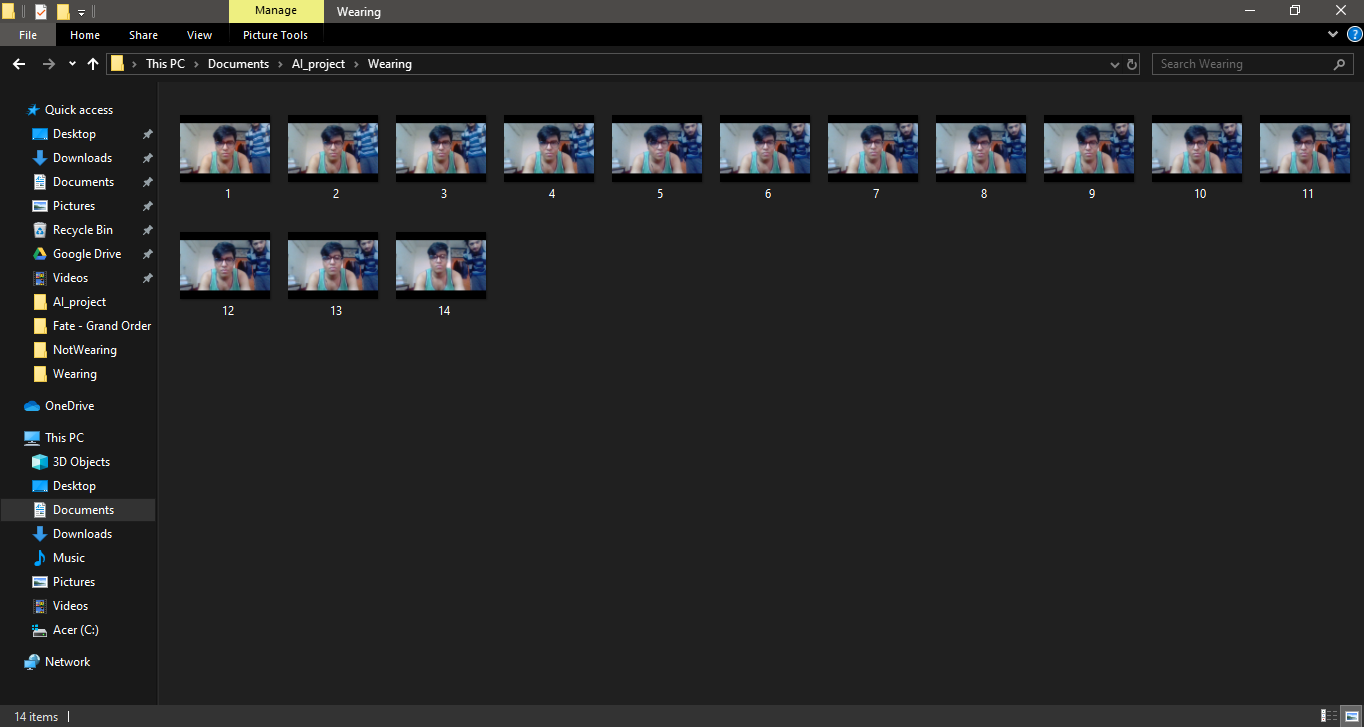




* Dataset stored

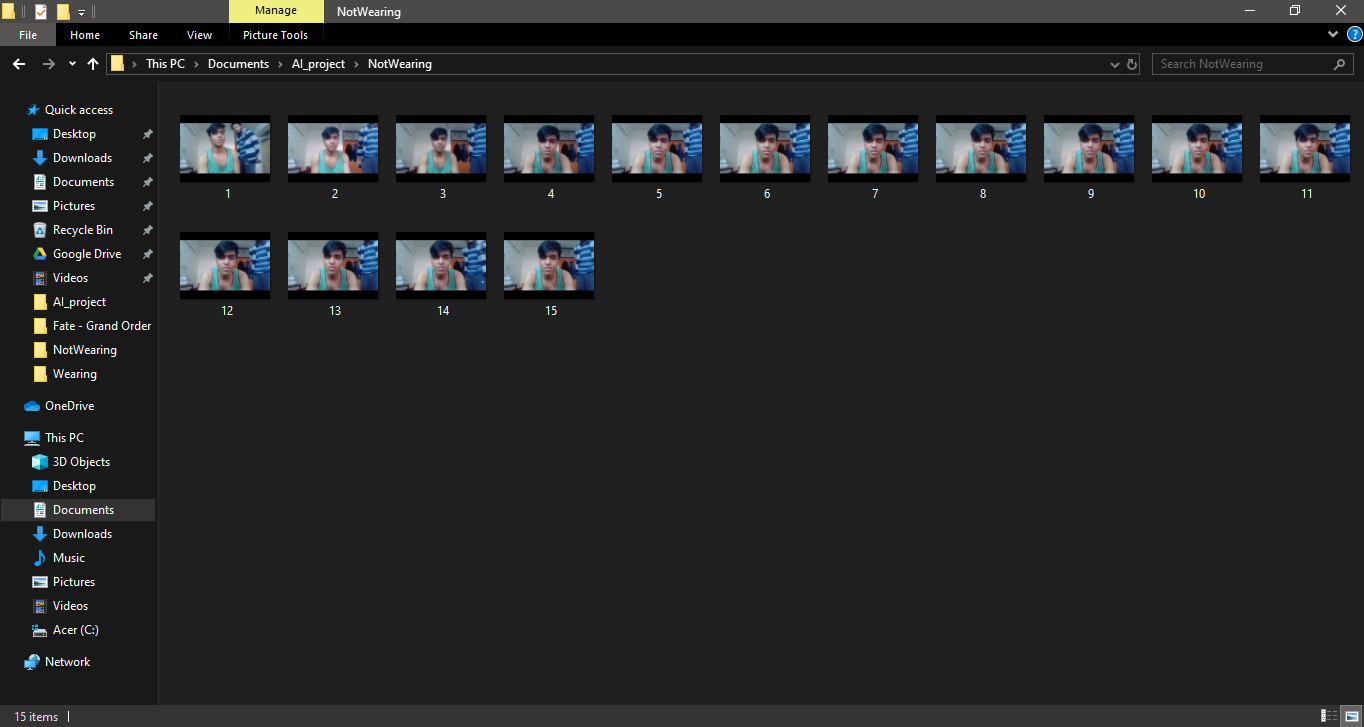


* Images stored in folder ‘Wearing’ from webcam



My roommate helping me out in testing

* Images stored in folder ‘NotWearing’ from webcam





**Libraries Used**

* Numpy: Used for array manipulation
* Matplotlib: Visualization of acquired results from each step of the process
* Dlib: Implementation of ‘One Millisecond Face Alignment with Ensemble of Regression Trees by Vahid Kazemi and Josephine Sullivan, CVPR20’ model for finding face landmarks
* Math: Trigonometric functions for face normalization
* Opencv: Used for image input and manipulation
* Datetime: Used for storing data of the user and classification
* Pytz: Timezone implementation for UTC
* Pandas: Dataframe manipulation and storage
* Time: Adding delays for taking images
* Warnings: Warning exception handling

**References**

* [opencv.org](http://www.opencv.org)
* dlib.net
* pypi.org
* docs.python.org