NATIONAL INSTITUTE OF TECHNOLOGY

WARANGAL

# PROJECT ON : AUTO CAPTURE SELFIE BY DETECTING SMILE

# PROJECT UNDER : DR.RAVI KUMAR JATOTH.

# PRESENTED BY : KRANTHI GOLLAPATI.

# ON THE OCCASION OF : SUMMER INTERNSHIP PROGRAM-2022.

# ROLL NO: 21MAC2R08.

# MATHEMATICS AND SCIENTIFIC COMPUTING (2021-2023).

0.Abstract

1. Introduction
2. Opencv
3. Datetime
4. Haar features
5. implementation
6. Code
7. Results
8. Conclusion
9. Reference

ABSTRACT:

Smile detection from facial images is a specialized task in facial expression analysis with many potential applications such as smiling payment, patient monitoring and photo selection.

A selfie is **any photograph of yourself taken by yourself**. These days, there are different kinds of cameras you can use to take a selfie. You can use a webcam, a professional camera, or, more commonly, your smartphone's front camera. There are different styles of taking selfies too.

INTRODUCTION:

Selfies are suddenly ubiquitous. In declaring selfie Oxford Dictionaries’ 2013 Word of the Year, Editorial Director Judy Pearsall explained that their big data analyses of English words in use showed “a phenomenal upward trend” in mentions of selfies (Oxford Dictionaries, 2013, para. 3). “Are you sick of reading about selfies?” asks an article in The Atlantic (Garber, 2014, para. 1), announcing that selfies are now boring and thus finally interesting. “Are you tired of hearing about how those pictures you took of yourself on vacation last month are evidence of narcissism, but also maybe of empowerment, but also probably of the click-by-click erosion of Culture at Large?” Indeed, for all its usage, the term—and more so the practice(s)—remain fundamentally ambiguous, fraught, and caught in a stubborn and morally loaded hype cycle.

Before proceeding, a more fundamental question still: What precisely is a selfie? First and foremost, a selfie is a photographic object that initiates the transmission of human feeling in the form of a relationship (between photographer and photographed, between image and filtering software, between viewer and viewed, between individuals circulating images, between users and social software architectures, etc.). A selfie is also a practice—a gesture that can send (and is often intended to send) different messages to different individuals, communities, and audiences. This gesture may be dampened, amplified, or modified by social media censorship, social censure, misreading of the sender’s original intent, or adding additional gestures to the mix, such as likes, comments, and remixes.

The ability to show and interpret emotions is crucial for human interaction. Detecting and modeling user’s emotions can therefore be considered another goal of Perceptual User Interfaces. In this respect, the human smile is a distinct facial configuration (suggesting that it may not be very difficult to detect) and can be very informative. Smile detection can be used in any application that requires to assess the user’s state such as distance learning systems, patient monitoring, film ratings, etc.

Opencv:

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

## **Features of OpenCV Library**

Using OpenCV library, you can −

* Read and write images
* Capture and save videos
* Process images (filter, transform)
* Perform feature detection
* Detect specific objects such as faces, eyes, cars, in the videos or images.
* Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

## **OpenCV Library Modules**

Following are the main library modules of the OpenCV library.

### **Core Functionality**

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array **Mat**, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.core**.

### **Image Processing**

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.imgproc**.

### **Video**

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.video**.

### **Video I/O**

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.videoio**.

### **calib3d**

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.calib3d**.

### **features2d**

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.features2d**.

### **Objdetect**

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.objdetect**.

### **Highgui**

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the features of this module is included in two different packages namely, **org.opencv.imgcodecs** and **org.opencv.videoio**.

Datetime:

In Python, date and time are not a data type of their own, but a module named **datetime** can be imported to work with the date as well as time. **Python Datetime module** comes built into Python, so there is no need to install it externally.

Python Datetime module supplies classes to work with date and time. These classes provide a number of functions to deal with dates, times and time intervals. Date and datetime are an object in Python, so when you manipulate them, you are actually manipulating objects and not string or timestamps.

The DateTime module is categorized into 6 main classes –

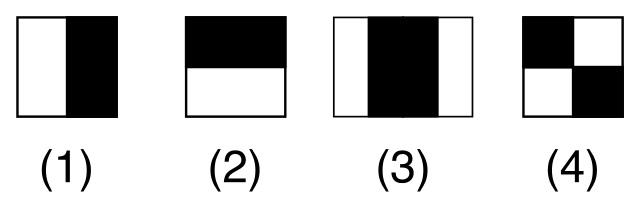
* [**date**](https://www.geeksforgeeks.org/python-datetime-date-class/) – An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.
* [**time**](https://www.geeksforgeeks.org/python-datetime-time-class/) – An idealized time, independent of any particular day, assuming that every day has exactly 24\*60\*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.
* [**datetime**](https://www.geeksforgeeks.org/python-datetime-datetime-class/) – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.
* [**timedelta**](https://www.geeksforgeeks.org/python-datetime-timedelta-class/) – A duration expressing the difference between two date, time, or datetime instances to microsecond resolution.
* **tzinfo** – It provides time zone information objects.
* **timezone** – A class that implements the tzinfo abstract base class as a fixed offset from the UTC (New in version 3.2).

cap=cv2.VideoCapture(0):

To capture a video in Python, use the cv2 VideoCapture class and then create an object of VideoCapture. VideoCapture has the device index or the name of a video file. The device index is just an integer to define a Camera. If we pass 0, it is for the first or primary camera, 1 for the second camera, etc.

Haar features:

Haar features are sequence of rescaled square shape functions proposed by Alfred Haar in 1909. They are similar to convolution kernels taught in the Convolution Neural Networks course. We will apply these haar features to all relevant parts of face so as to detect human face.



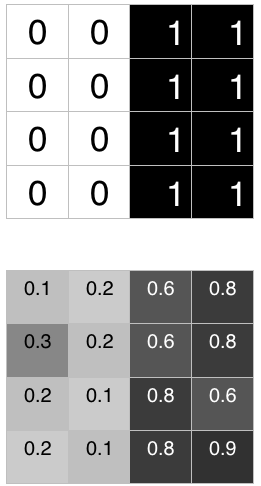
As we see in the above image, there are edge features (1 and 2), line features (3). They are white and black pixels images (value 0 or value 1). But usually we have greyscale/colour image (pixel value range from 0 to 255). For now assume ideal scenario that we have black and white image

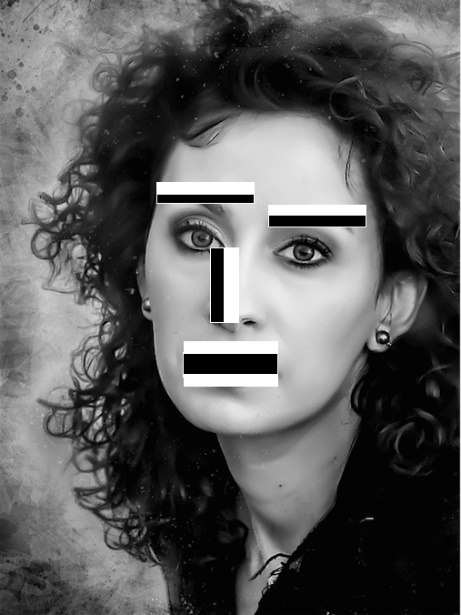


To detect eyebrow, we will use Haar feature (image (1)) because forehead and eyebrow form lighter pixels- darker pixel like image. Similarly, to detect lips we use similar to Haar like feature (image(3)) with lighter-darker-lighter pixels. To detect nose, we might use darker-lighter Harr like feature from (image(1)). And so on.

Lets look at some computations that are required.

For black and white image (refer first box below), pixel values are 0 or 1 (ideal case) but in real cases we have normalised greyscale image as shown in bottom box containing pixel values





According to Viola-Jonas algorithm, to detect Haar like feature present in an image, below formula should give result closer to 1. The closer the value is to 1, the greater the change of detecting Haar feature in image.



Ideal case : Delta = (1/8)\*(8) — (1/8)\*0 = 1

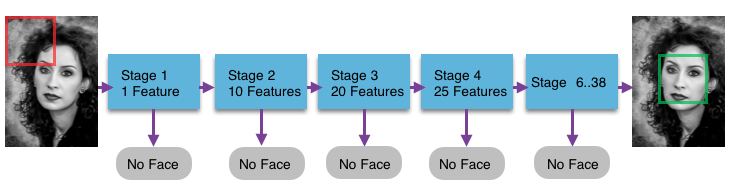
Real case: Delta = (1/8)\*(5.9) — (1/8)\*(1.3)=0.575

*(For greyscale image, assume we have set White-Dark threshold to 0.3. Meaning pixels with value less than or equal to 0.3 are considered white and anything greater that 0.3 is considered as dark)*

We can define another threshold parameter to detect edge or Haar feature.We are calling it Delta *(this is different from White-Dark threshold)*. Assume here we have set threshold to 0.5. Any delta value greater than 0.5 detects Haar feature. In this fashion we can apply, get and detect most relevant features on a given image like eyebrow,lips,nose etc.

Next we will quickly see what are **Haar Cascade Classifiers** —

They are series of classifiers or features (as we have seen above) used to identify object in an image. Using sliding windows and number of haar features (increases as number of stages increase), finally leading to detect face or not. There are total 38 stages defined for Viola Jonas Method. Depending upon the sliding windows size and face location, number of features, face can be detected at a certain stage.



Implementation:

Cascade classifier:

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

\_, frame =cap.read():

This code initiates an infinite loop (to be broken later by a break statement), frame being defined as the cap.read(). Basically,the frame is each frame that is returned. If there is no frame, you wont get an error, you will get None.

gray=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY):

cv2. cvtColor() method is **used to convert an image from one color space to another**.Here we are converting whole frame into gray image.

face=face\_cascade.detectMultiScale(gray,1.3,5):

 MultiScale detects objects of different sizes in the input image and returns rectangles positioned on the faces. The first argument is the image, the second is the scalefactor (how much the image size will be reduced at each image scale), and the third is the minNeighbors (how many neighbors each rectangle should have). The values of 1.3 and 5 are based on experimenting and choosing those that worked best.

for x,y,w,h in face:

We will loop through each rectangle (each face detected) using its coordinates generated by the function we discussed above.

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2):

Here image is frame.starting point is (x,y),ending point is (x+w,y+h),color is aqua and thickness is 2.

**OpenCV-Python**— is a Python bindings library for solving computer vision problems. cv2.rectangle () is used to draw a rectangle on any image.

**Syntax:**cv2.rectangle (image, start\_point, end\_point, color , thickness)

**Parameters:**  
**image:**It is the image on which rectangle is to be drawn.  
**start\_point:**It is the starting coordinates of rectangle. The coordinates are represented as tuples of two values  (**X**coordinate value, **Y**coordinate value).  
**end\_point:**It is the ending coordinates of rectangle. The coordinates are represented as tuples of two values (**X**coordinate value, **Y**coordinate value).  
**color:**It is the color of border line of rectangle to be drawn. For **BGR**, we pass a tuple. eg: (255, 0, 0) for blue color.  
**thickness:**It is the thickness of the rectangle border line in **px**. Thickness of **-1 px**will fill the rectangle shape by the specified color.

# face\_roi=frame[y:y+h,x:x+w]:**face ROI was a rectangular shape positioned automatically to cover the face, hair, and neck of models, whilst fixation coordinates within the rectangular areas were assigned to eyes and mouth ROI for each model**

gray\_roi=gray[y:y+h,x:x+w]:

It will convert face\_roi into gray scale image

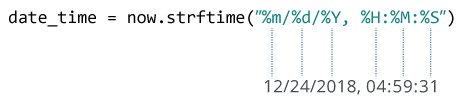
for x1,y1,w1,h1 in smile:

We will loop through each rectangle (smile detected) using its coordinates generated by the function we discussed above.

cv2.rectangle(face\_roi,(x1,y1),(x1+w1,y1+h1),(0,0,255),2):

Here we are creating rectangle to smile

time\_stamp=datetime.datetime.now().strftime('%y-%m-%d-%H-%M-%S'):



We are creating timestamp to the selfie photo.

file\_name=f'selfie-{time\_stamp}.png':

we are creating selfie photo name selfie and also with time stamp.

cv2.imwrite(file\_name,original\_frame):

this is used to save selfie photo in your pc

if cv2.waitKey(10)==ord('q'):

break:

it is used to stop the web cam by pressing ‘q’ letter.

Code:

import cv2

import datetime

cap=cv2.VideoCapture(0)

face\_cascade=cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

smile\_cascade=cv2.CascadeClassifier('haarcascade\_smile.xml')

while True:

\_, frame =cap.read()

original\_frame =frame.copy()

gray=cv2.cvtColor(frame,cv2.COLOR\_BGR2GRAY)

face=face\_cascade.detectMultiScale(gray,1.3,5)

for x,y,w,h in face:

cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)

face\_roi=frame[y:y+h,x:x+w]

gray\_roi=gray[y:y+h,x:x+w]

smile = smile\_cascade.detectMultiScale(gray\_roi,1.3,25)

for x1,y1,w1,h1 in smile:

cv2.rectangle(face\_roi,(x1,y1),(x1+w1,y1+h1),(0,0,255),2)

time\_stamp=datetime.datetime.now().strftime('%y-%m-%d-%H-%M-%S')

#print("Image "+str(cnt)+"Saved")

##path=r'Pictures'+str(cnt)+'.jpg'

file\_name=f'selfie-{time\_stamp}.png'

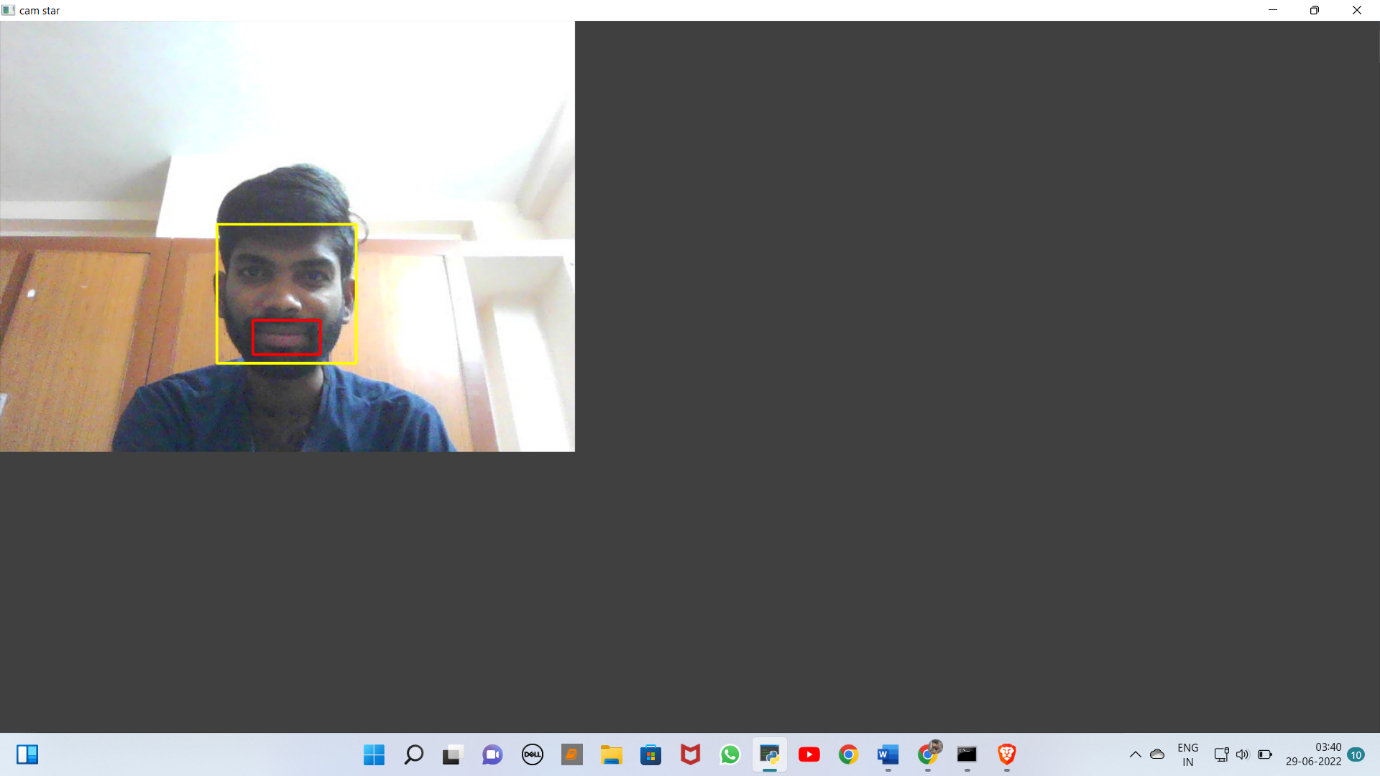
cv2.imwrite(file\_name,original\_frame)

cv2.imshow('cam star',frame)

if cv2.waitKey(10)==ord('q'):

break

Results:



Interface after executing the program.



Picture which is saved in gallery. Which selfie saved in the name of

selfie-22-06-29-03-40-42

here:

22-06-29-03-40-42 is time stamp which is mentioned in code

References:

* <https://www.geeksforgeeks.org/python-datetime-module/#:~:text=Python%20Datetime%20module%20supplies%20classes,and%20not%20string%20or%20timestamps>.
* [www.google.com](http://www.google.com)
* [www.wekipedia.com](http://www.wekipedia.com)
* Smile Detection for User Interfaces O. Deniz, M. Castrillon, J. Lorenzo, L. Anton, and G. Bueno First and last authors: Universidad de Castilla-La Mancha, E.T.S.I.I Campus Universitario, Avda. Camilo Jose Cela s/n, 13071, Spain Oscar.Deniz@uclm.es Rest of the authors: Universidad de Las Palmas de Gran Canaria Dpto. Informatica y Sistemas,Campus de Tafira, Edificio de Informatica 35017 Las Palmas, Spain
* What Does the Selfie Say? Investigating a Global Phenomenon Introduction THERESA M. SENFT New York University, USA NANCY K. BAYM Microsoft Research, USA