FACE DETECTION

Group Project

Venkata Sumanth Nagabhairu(20012395)
Sudheera Ambavaram(20011886)
Yogesh Gutta(20010910)

Introduction

The object of our project is to design software that can detect human faces from images.
Face detection is a computer vision technology that helps to locate/visualise human faces in digital images. This technique is a specific use case of “object detection technology” that deals with detecting instances of semantic objects of a certain class (such as humans, buildings or cars) in digital images and videos. With the advent of technology, face detection has gained a lot of importance especially in fields like photography, security, and marketing.
In face analysis, face detection helps identify which parts of an image or video should be focused on to determine age, gender and emotions using facial expressions. In a facial recognition system -- which maps an individual's facial features mathematically and stores the data as a faceprint -- face detection data is required for the algorithm that discerns which parts of an image or video are needed to generate a faceprint. Once identified, the new faceprint can be compared with stored faceprints to determine if there is a match.

Objective

Whenever we implement a new system it is developed to remove the shortcomings of the existing system. The computerized mechanism has more edge than the manual system. The existing system is based on a manual system which takes a lot of time to get performance of the work. The proposed system is a web application and maintains a centralized repository of all related information. The system allows one to easily access the software and detect what he wants.

How face detection works

Face detection applications use algorithms and ML to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes -- one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face.

Advantages of face detection

As a key element in facial imaging applications, such as facial recognition and face analysis, face detection creates various advantages for users, including:

- **Improved security :** Face detection improves surveillance efforts and helps track down criminals and terrorists. Personal security is also enhanced since there is nothing for hackers to steal or change, such as passwords.

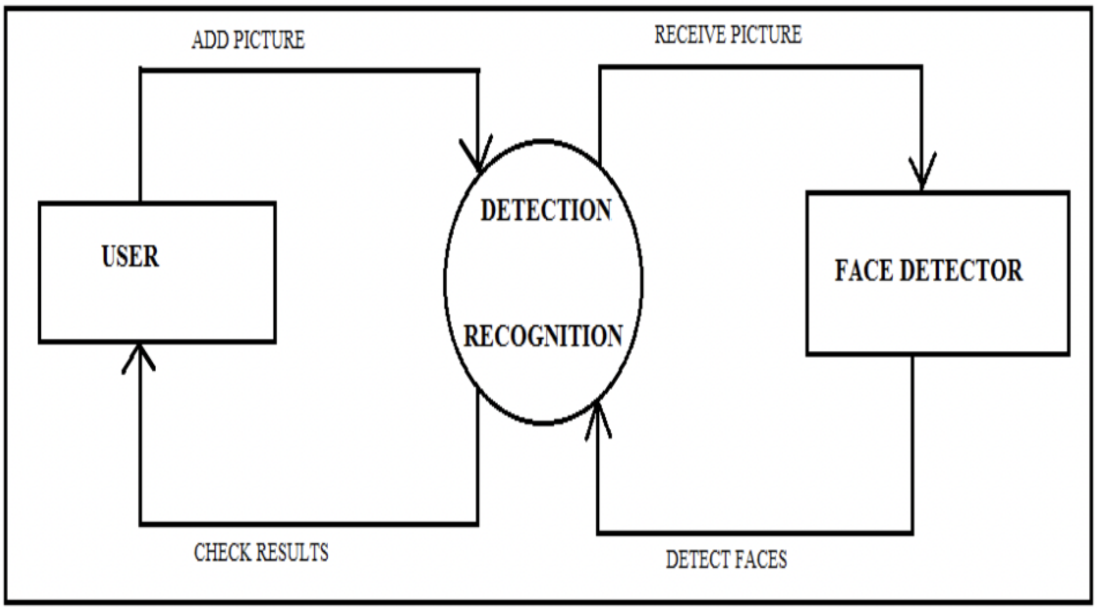
- **Easy to integrate :** Face detection and facial recognition technology is easy to integrate, and most solutions are compatible with the majority of security software.
- **Automated identification :** In the past, identification was manually performed by a person; this was inefficient and frequently inaccurate. Face detection allows the identification process to be automated, thus saving time and increasing accuracy.

Uses of face detection

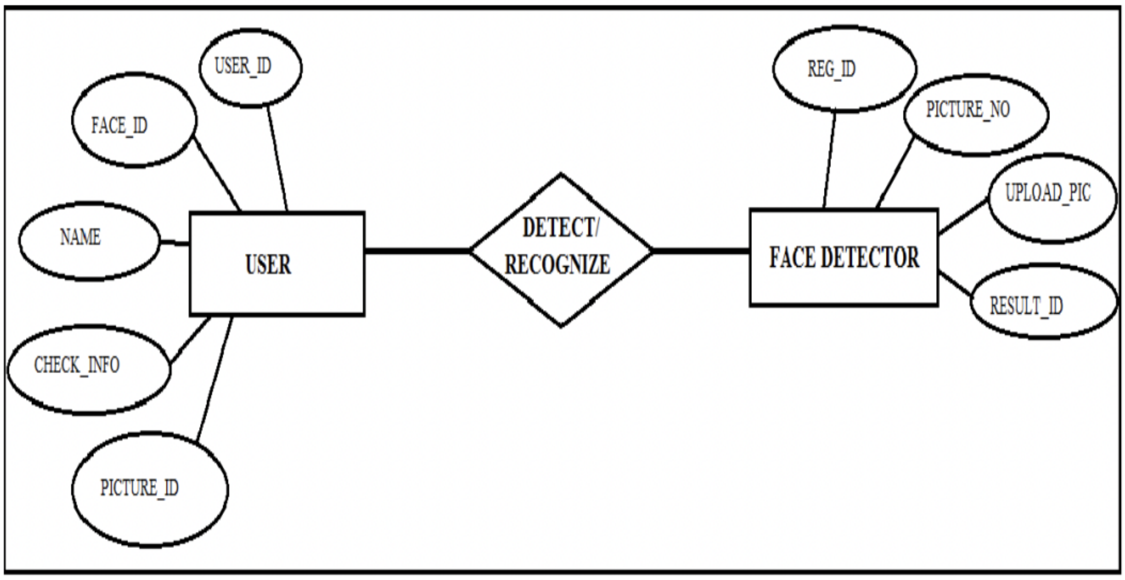
Although all facial recognition systems use face detection, not all face detection systems are used for facial recognition. Face detection can also be applied for facial motion capture, or the process of electronically converting a human's facial movements into a digital database using cameras or laser scanners. This database can be used to produce realistic computer animation for movies, games or avatars.
Face detection can also be used to auto-focus cameras or to count how many people have entered an area. The technology also has marketing applications -- for example, displaying specific advertisements when a particular face is recognized.
An additional use is drawing language inferences from visual cues, or "lip reading." This can help computers determine who is speaking, which may be helpful in security applications. Furthermore, face detection can be used to help determine which parts of an image to blur to assure privacy.
This is essentially a segmentation problem and in practical systems, most of the effort goes into solving this task. In fact the actual recognition based on features extracted from these facial landmarks is only a minor last step. There are two types of face detection problems:
1) Face detection in images and
2) Real-time face detection

Diagram

1. **Data Flow Diagram**



2. **Entity Relationship Diagram**



Program Code

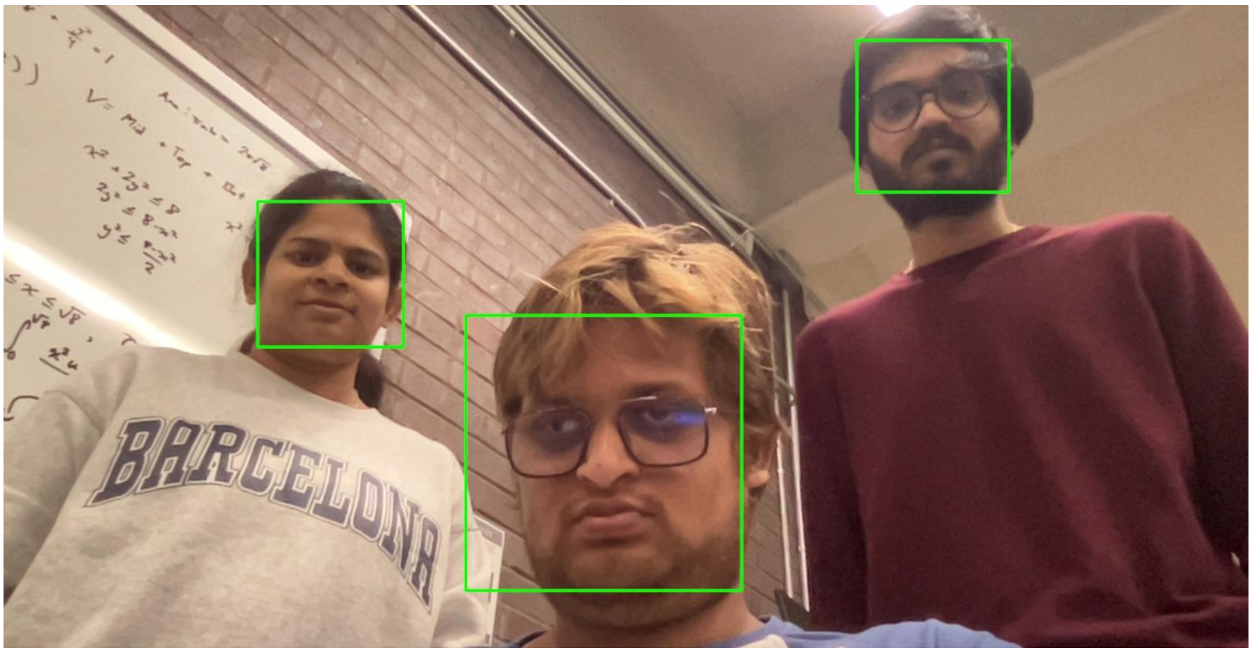
Webcam 1 :

```
import cv2
import sys
import logging as log
import datetime as dt
from time import sleep

cascPath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascPath)
log.basicConfig(filename='webcam.log',level=log.INFO)

video_capture = cv2.VideoCapture(0)
anterior = 0

while True:
    if not video_capture.isOpened():
        print('Unable to load camera.')
        sleep(5)
        pass

    # Capture frame-by-frame
    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30)
    )
    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    if anterior != len(faces):
        anterior = len(faces)
        log.info("faces: "+str(len(faces))+" at "+str(dt.datetime.now()))

    # Display the resulting frame
    cv2.imshow('Video', frame)

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

    # Display the resulting frame
    cv2.imshow('Video', frame)

# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

Webcam 1 :

```
import cv2
import sys

cascPath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascPath)

video_capture = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.cv.CV_HAAR_SCALE_IMAGE
    )

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
    # Display the resulting frame
    cv2.imshow('Video', frame)

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

# When everything is done, release the capture
video_capture.release()
cv2.destroyAllWindows()
```

Result



Conclusion

Successfully detected the human faces using a webcam.
A face detection system would certainly speed up the process of checking student attendance in comparison to other biometrics authentication methods and in the right circumstances it would be able to match their accuracy. Nowadays there are a wide variety of software, whether it is a Face API like Microsoft's or a library like OpenCV, that makes face detection accessible and reliable and is constantly improving.