PROJECT REPORT
ON
USING FACIAL RECOGNITION TO SPEED UP QUEUES IN LARGE HOSPITALS

BY

Akshat Lal	2018B4A70051P
Arnav Ahuja	2018B4A70619P

AT

Tamilnadu Health Systems Project
Department of Health and Family Welfare, Government of Tamil Nadu

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
JUNE 2020

A REPORT
ON
USING FACIAL RECOGNITION TO SPEED UP QUEUES IN LARGE HOSPITALS

BY

AKSHAT LAL	2018B4A70051P	**M.Sc. (Hons.) Mathematics +B.E.(Hons.) Computer Science**
ARNAV AHUJA	2018B4A70619P	**M.Sc. (Hons.) Mathematics +B.E.(Hons.) Computer Science**

Prepared in partial fulfilment of the Practice School-I Course Nos.
BITS C221/BITS C231/BITS C241

AT

Tamilnadu Health Systems Project
Department of Health and Family Welfare, Government of Tamil Nadu

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
JUNE 2020

ACKNOWLEDGEMENT

We would like to thank BITS, Pilani Practice School Division for providing this excellent opportunity to pursue our Practice School-I program at Tamil Nadu Health System Reform Program, Chennai.

We would also like to express our gratitude to all those people whose efforts have made possible our professional experience at Tamil Nadu Health System Reform Program, Chennai, a success even in this remote work from home mode during the pandemic.

We would like to express our genuine gratitude towards Dr. Vidhuthalai, Dr. Vijyalakshmi for allowing us to be part of Tamil Nadu Health System Reform Program, Chennai, an elite organization and for providing us such a great learning opportunity for facilitating the process, for assigning the project based on our interest and guiding us whenever we needed any help.

We are also grateful to our PS Faculty, Prof. Ankur Pachauri, for his dedication towards students currently in the PS and his quick response to all our queries.

Finally, we would like to thank our parents, friends, and relatives for their immense
support throughout the project.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN) Practice School Division

Station: Tamil Nadu Health System Reform Program, Chennai **Centre:** Chennai
Duration: 18/05/20 TO 27/06/20 (6 WEEKS)
Date of Start: 18 MAY 2020 **Date of Submission:** 6/06/2020

Title of the Project: Using Facial Recognition to speed up queues in large hospitals.

STUDENT DETAILS:

AKSHAT LAL	2018B4A70051P	**M.Sc. (Hons.) Mathematics +B.E.(Hons.) Computer Science**
ARNAV AHUJA	2018B4A70619P	**M.Sc. (Hons.) Mathematics +B.E.(Hons.) Computer Science**

EXPERT DETAILS
Name: Dr. Vidhuthalai
Designation: IT Expert
Name of PS Faculty: Prof. Ankur Pachauri

Key Words: Machine Learning, Facial Recognition
Project Area: Product Development

Abstract: The queues of patients in large hospitals become a very tedious process for hospitals to manage. The aim of this project is to automate this process by implementing real time facial recognition to determine the order of patients as they enter the hospital. Facial recognition is a part of machine learning that has seen a growing demand in various fields in the industry. The algorithms used today can outmatch a human's ability to recognise faces. Initially, the report discusses the development in facial recognition methodologies. Then the report goes on to discuss how we can use this technology to solve the given industrial problem. Then lastly, before concluding, the report discusses the main challenges and limitations of facial recognition.

Signature(s) of Student(s)
Date

Signature of PS Faculty
Date

TABLE OF CONTENTS

INTRODUCTION

Face recognition is the problem of identifying and verifying people in a photograph by their face.

It is a task that is trivially performed by humans, even under varying light and when faces are changed by age or obstructed with accessories and facial hair. Nevertheless, it has remained a challenging computer vision problem for decades until recently.

Deep learning methods can leverage very large datasets of faces and learn rich and compact representations of faces, allowing modern models to first perform as-well and later to outperform the face recognition capabilities of humans.

The introduction will shed a light on:

- Face recognition is a broad problem of identifying or verifying people in photographs and videos.
- Face recognition is a process comprised of detection, alignment, feature extraction, and a recognition task
- Deep learning models first approached then exceeded human performance for face recognition tasks.

Overview

The overview is divided into five parts; they are:

1. Faces in Photographs
2. Process of Automatic Face Recognition
3. Face Detection Task
4. Face Recognition Tasks
5. Deep Learning for Face Recognition

Faces in Photographs

There is often a need to automatically recognize the people in a photograph.

The reasons to want to automatically recognise a person in a photograph are uncountable.

For example:

- We may want to restrict access to a resource to one person, called face authentication.
- We may want to confirm that the person matches their ID, called face verification.
- We may want to assign a name to a face, called face identification.

Generally, we refer to this as the problem of automatic "*face recognition*" and it may apply to both still photographs or faces in streams of video.

Process of Automatic Face Recognition

Face recognition is the problem of identifying or verifying faces in a photograph.

A general statement of the problem of machine recognition of faces can be formulated as follows: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces

Face recognition is often described as a process that first involves four steps; they are: face detection, face alignment, feature extraction, and finally face recognition.

1. **Face Detection**. Locate one or more faces in the image and mark with a bounding box.
2. **Face Alignment**. Normalize the face to be consistent with the database, such as geometry and photometrics.
3. **Feature Extraction**. Extract features from the face that can be used for the recognition task.
4. **Face Recognition**. Perform matching of the face against one or more known faces in a prepared database.

A given system may have a separate module or program for each step, which was traditionally the case, or may combine some or all the steps into a single process.

A helpful overview of this process is provided in the book "Handbook of Face Recognition," provided below:

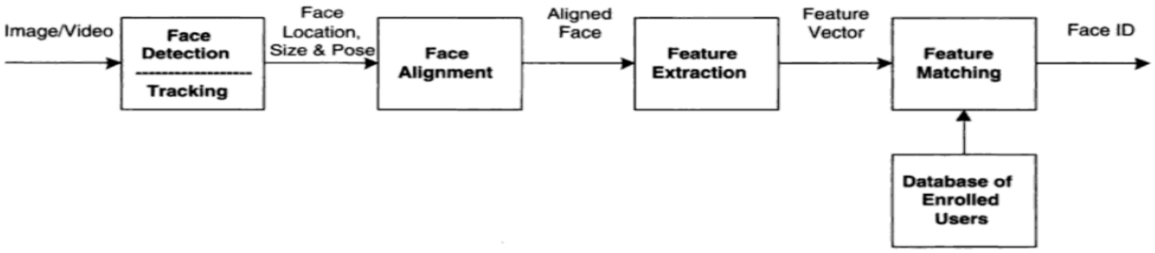


Fig. 1.2. Face recognition processing flow.

Face Detection Task

Face detection is the non-trivial first step in face recognition.

It is a problem of object recognition that requires that both the location of each face in a photograph is identified (e.g. the position) and the extent of the face is localized (e.g. with a bounding box). Object recognition itself is a challenging problem, although in this case, it is similar as there is only one type of object, e.g. faces, to be localized, although faces can vary wildly.

The human face is a dynamic object and has a high degree of variability in its appearance, which makes face detection a difficult problem in computer vision.

Further, because it is the first step in a broader face recognition system, face detection must be robust. For example, a face cannot be recognized if it cannot first be detected. That means faces must be detected with all manner of orientations, angles, light levels, hairstyles, hats, glasses, facial hair, makeup, ages, and so on.

As a visual front-end processor, a face detection system should also be able to achieve the task regardless of illumination, orientation, and camera distance

The 2001 paper titled “Face Detection: A Survey” provides a taxonomy of face detection methods that can be broadly divided into two main groups:

- Feature-Based.
- Image-Based.

The feature-based face detection uses hand-crafted filters that search for and locate faces in photographs based on a deep knowledge of the domain. They can be very fast and very effective when the filters match, although they can fail dramatically when they do not, e.g. making them somewhat fragile.

Alternately, image-based face detection is holistic and learns how to automatically locate and extract faces from the entire image. Neural networks fit into this class of methods.

Perhaps the dominant method for face detection used for many years (and was used in many cameras) was described in the 2004 paper titled “Robust Real-time Object Detection,” called the detector cascade or simply “*cascade*.”

Face Recognition Tasks

The task of face recognition is broad and can be tailored to the specific needs of a prediction problem.

There are mainly three face recognition tasks:

- **Face Matching**: Find the best match for a given face.
- **Face Similarity**: Find faces that are most like a given face.
- **Face Transformation**: Generate new faces that are like a given face.

They are summarized as follows:

Matching requires that the candidate matching face image be in some set of face images selected by the system. Similarity detection requires in addition to matching that images of faces be found which are similar to a recalled face this requires that the similarity measure used by the recognition system closely match the similarity measures used by humans Transformation applications require that new images created by the system be similar to human recollections of a face.

The two main modes for face recognition are:

- **Face Verification**. A one-to-one mapping of a given face against a known identity (e.g. *is this the person?*).
- **Face Identification**. A one-to-many mapping for a given face against a database of known faces (e.g. *who is this person?*).

A face recognition system is expected to identify faces present in images and videos automatically. It can operate in either or both of two modes: (1) face verification (or authentication), and (2) face identification (or recognition).

We can describe the problem of face recognition as a supervised predictive modelling task trained on samples with inputs and outputs.

In all tasks, the input is a photo that contains at least one face, most likely a detected face that may also have been aligned.

The output varies based on the type of prediction required for the task, for example:

- It may then be a binary class label or binary class probability in the case of a face verification task.
- It may be a categorical class label or set of probabilities for a face identification task.
- It may be a similarity metric in the case of a similarity type task.

Deep Learning for Face Recognition

Face recognition has remained an active area of research in computer vision.

Perhaps one of the more widely known and adopted “machine learning” methods for face recognition was described in the 1991 paper titled “Face Recognition Using Eigenfaces.” Their method, called simply “*Eigenfaces*,” was a milestone as it achieved impressive results and demonstrated the capability of simple holistic approaches.

Face images are projected onto a feature space (“face space”) that best encodes the variation among known face images. The face space is defined by the “eigenfaces”, which are the eigenvectors of the set of faces; they do not necessarily correspond to isolated features such as eyes, ears, and noses.

Given the breakthrough of AlexNet in 2012 for the simpler problem of image classification, there was a flurry of research and publications in 2014 and 2015 on deep learning methods for face recognition. Capabilities quickly achieved near-human-level performance, then exceeded human-level performance on a standard face recognition dataset within a three-year period, which is an astounding rate of improvement given the prior decades of effort.

There are perhaps four milestone systems on deep learning for face recognition that drove these innovations; they are: DeepFace, the DeepID series of systems, VGGFace, and FaceNet.

DeepFace is a system based on deep convolutional neural networks described by Yaniv Taigman, et al. from Facebook AI Research and Tel Aviv. It was described in the 2014 paper titled “DeepFace: Closing the Gap to Human-Level Performance in Face Verification.” It was the first major leap forward using deep learning for face recognition, achieving near human-level performance on a standard benchmark dataset.

The key challenge of face recognition is to develop effective feature representations for reducing intra-personal variations while enlarging inter-personal differences. […] The face identification task increases the inter-personal variations by drawing DeepID2 features extracted from different identities apart, while the face verification task reduces the intra-personal variations by pulling DeepID2 features extracted from the same identity together, both of which are essential to face recognition..

The DeepID systems were among the first deep learning models to achieve better-than-human performance on the task, e.g. DeepID2 achieved 99.15% on the Labeled Faces in the Wild (LFW) dataset, which is better-than-human performance of 97.53%. Subsequent systems such as FaceNet and VGGFace improved upon these results.

FaceNet, that directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. […] Our method uses a deep convolutional network trained to directly optimize the embedding itself, rather than an intermediate bottleneck layer as in previous deep learning approaches. To train, we use triplets of roughly aligned matching / non-matching face patches generated using a novel online triplet mining method

Although these may be the key early milestones in the field of deep learning for computer vision, progress has continued, with much innovation focused on loss functions to effectively train the models

MODEL FOR RECOGNITION (KEY FEATURES)

The model will be able to:

1) Carry out facial recognition in real time.
2) Identify Faces of patients and match them against a given data set.
3) Store new faces in the dataset.
4) Send text messages to mobile numbers of patients and assign them token numbers, based on verifying their entry into the hospital.

APPLICATION IN THE HEALTH INDUSTRY

Facial Recognition has far reaching uses in the industry. Here we would like to focus on using facial recognition to solve the given problem. The structure of the model will be:

1) Separating patients and non-patients.
2) Further dividing patients into registered and unregistered categories.
3) Registration of new patients and assigning token numbers to old ones, to put them in a queue.
4) Calling patients for consultation based on the token numbers assigned.

The first step will involve separating the people into two categories. The patients and non-patients will walk in separate channels. The model will be used on patients. Their faces will be identified in real time and matched with a database of faces.

In the second step, faces of patients which were not recognised will be marked as new registrations.

For the third step, they will be sent to a registration desk, where their details will be taken and mapped to their face. The new registration will be stored in the database and the database will be updated. If the face is recognised, the patient will be sent a token number on their registered phone number. After this, they will be sent to a waiting room, where a list of token numbers/ names of the patients will be displayed.

Finally, in step four, based on that list, patients will be called one by one for consultation.

MODEL TO BE USED (Haar Cascade algorithm)

The model will use a Haar Cascade algorithm to carry out facial recognition.

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features 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.

The algorithm has four stages:

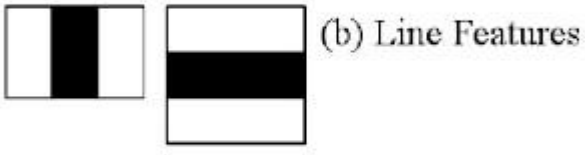
1. Haar Feature Selection
2. Creating Integral Images
3. Adaboost Training
4. Cascading Classifiers

It is well known for being able to detect faces and body parts in an image but can be trained to identify almost any object.

Let us take face detection as an example. Initially, the **algorithm needs a lot of positive images of faces and negative images** without faces to train the classifier. Then we need to extract features from it.

First step is to collect the Haar Features. A Haar feature considers adjacent rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums.

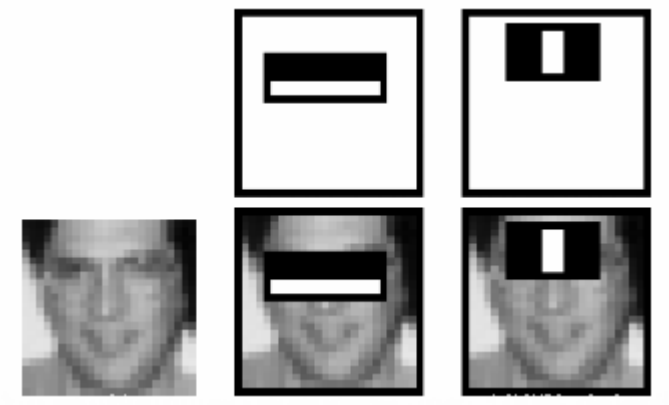

(a) Edge Features


(b) Line Features


(c) Four-rectangle features

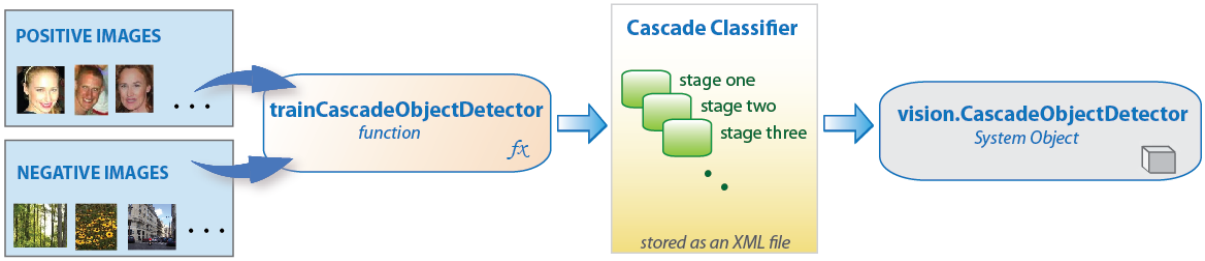
Integral Images are used to make this super-fast.

But among all these features we calculated, most of them are irrelevant. For example, consider the image below. Top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applying on cheeks or any other place is irrelevant.



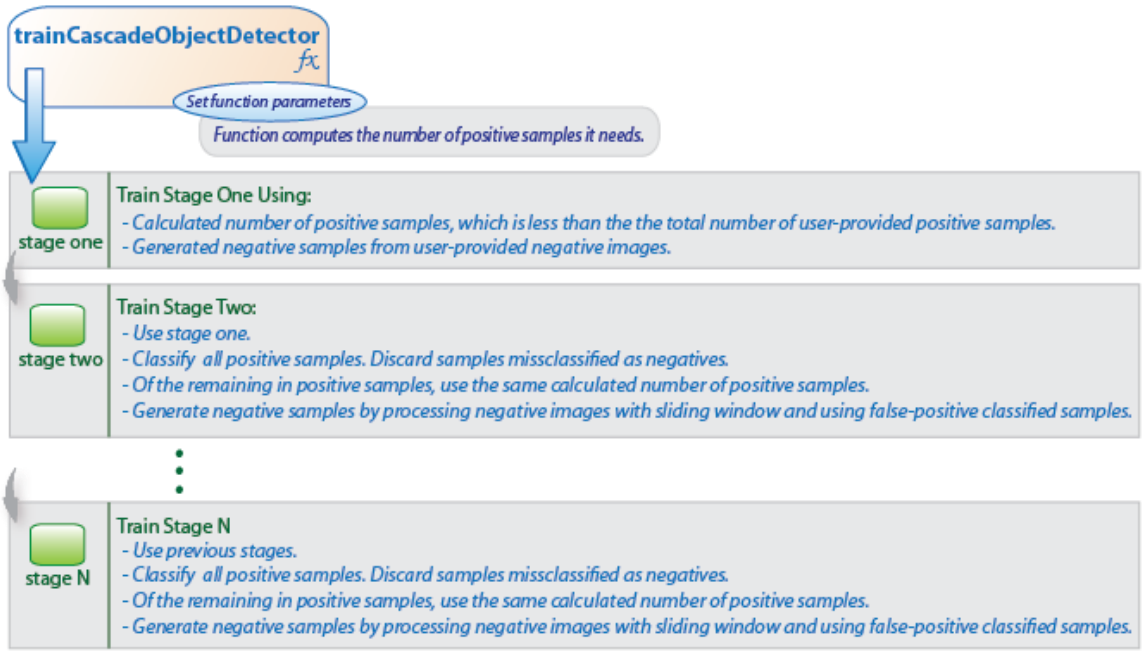
So how do we select the best features out of 160000+ features? This is accomplished using a concept called **Adaboost** which both selects the best features and trains the classifiers that use them. This algorithm constructs a “strong” classifier as a linear combination of weighted simple “weak” classifiers. The process is as follows.

During the detection phase, a window of the target size is moved over the input image, and for each subsection of the image and Haar features are calculated. You can see this in action in the video below. This difference is then compared to a learned threshold that separates non-objects from objects. Because each Haar feature is only a "weak classifier" (its detection quality is slightly better than random guessing) a large number of Haar features are necessary to describe an object with sufficient accuracy and are therefore organized into ***cascade classifiers*** to form a strong classifier.



Cascade classifier

Cascade classifier training requires a set of positive samples and a set of negative images. You must provide a set of positive images with regions of interest specified to be used as positive samples. You can use the Image Labeler to label objects of interest with bounding boxes. The Image Labeler outputs a table to use for positive samples. You also must provide a set of negative images from which the function generates negative samples automatically. To achieve acceptable detector accuracy, set the number of stages, feature type, and other function parameters.



IMPLEMENTATION

Libraries used:

1. OpenCV
2. NumPy
3. Face_Recognition
4. OS
5. DateTime

OpenCV:

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Python is a general-purpose programming language started by **Guido van Rossum** that became very popular very quickly, mainly because of its simplicity and code readability. It enables the programmer to express ideas in fewer lines of code without reducing readability.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of **NumPy**, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from NumPy arrays. This also makes it easier to integrate with other libraries that use NumPy such as SciPy and Matplotlib.

Functions used:

1. Cv2.imread: Reads the image and stores in a variable.
2. Cv2.cvtColor: Image colour correction and conversion to feed it to the algorithm.
3. Cv2.VideoCapture: Launches Webcam for real time video feed.
4. Cv2.resize: Downgrades image quality for fast recognition without compromising on the performance.
5. Cv2.rectangle: Identifies the face and draws a rectangle around the face.
6. Cv2.putText: Displays the name of the person recognised below the image.
7. Cv2.imshow: Used to display an image in a window. The window automatically fits to the image size.
8. Cv2.waitkey: Displays window until key is pressed to destroy it.

NumPy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing NumArray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors. It targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops, using NumPy.

Functions used:

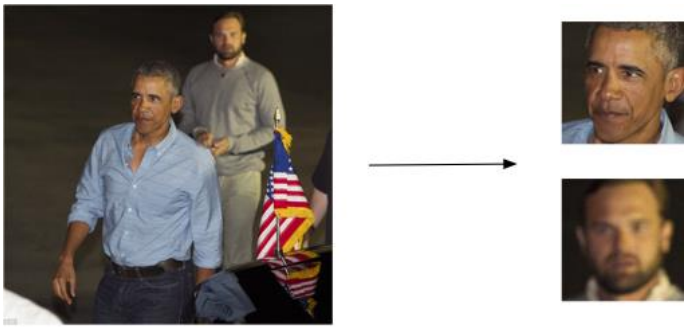
1. NumPy.argmin: Gets the argument of the registered face with minimum difference from the current face.

Face_Recognition:

- Recognize and manipulate faces from Python or from the command line with
- The world's simplest face recognition library.
- Built using dlib's state-of-the-art face recognition built with deep learning.
- The model has an accuracy of 99.38% on the labelled benchmark.
- This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line!

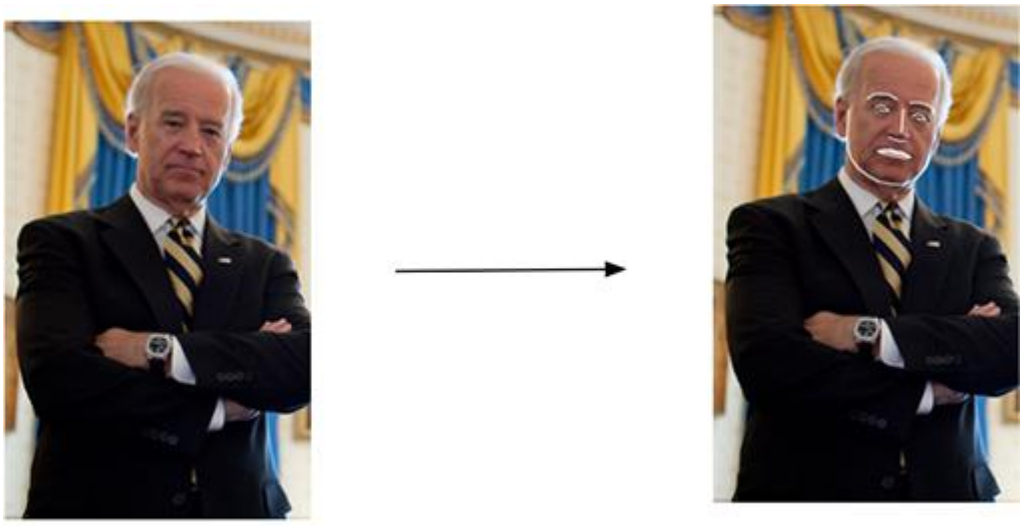
Features:

- Find all the faces that appear in a picture:



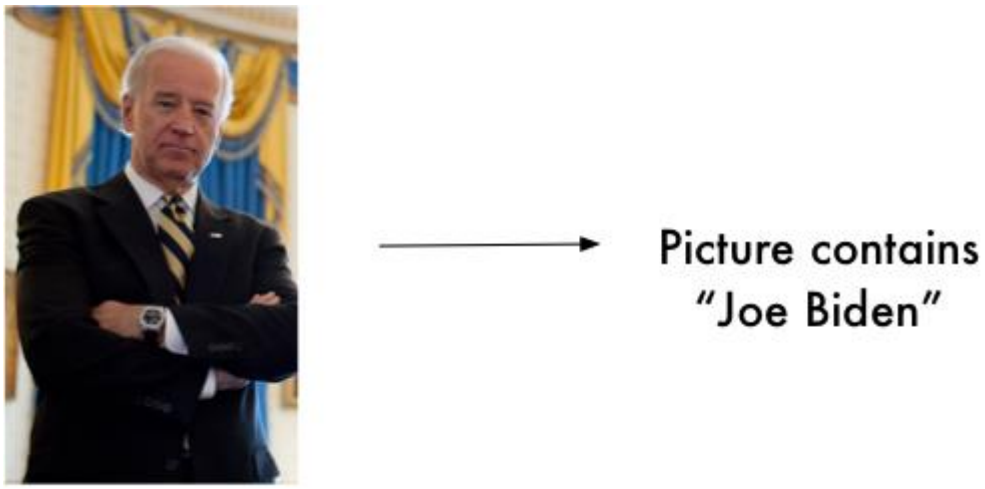
Input Output

- Find and manipulate facial features in pictures:



Input Output

- Identify faces in pictures:



Input Output

OS:

This module provides a portable way of using operating system dependent functionality. It is used for reading or writing to a file, to manipulate file paths, to read all the lines in a file in command line as well as to create temporary files and directories.

Functions used:

1. os.listdir: To get a list of the file names of registered images.
2. os.pth.splitext: To add new name to the existing list of names.

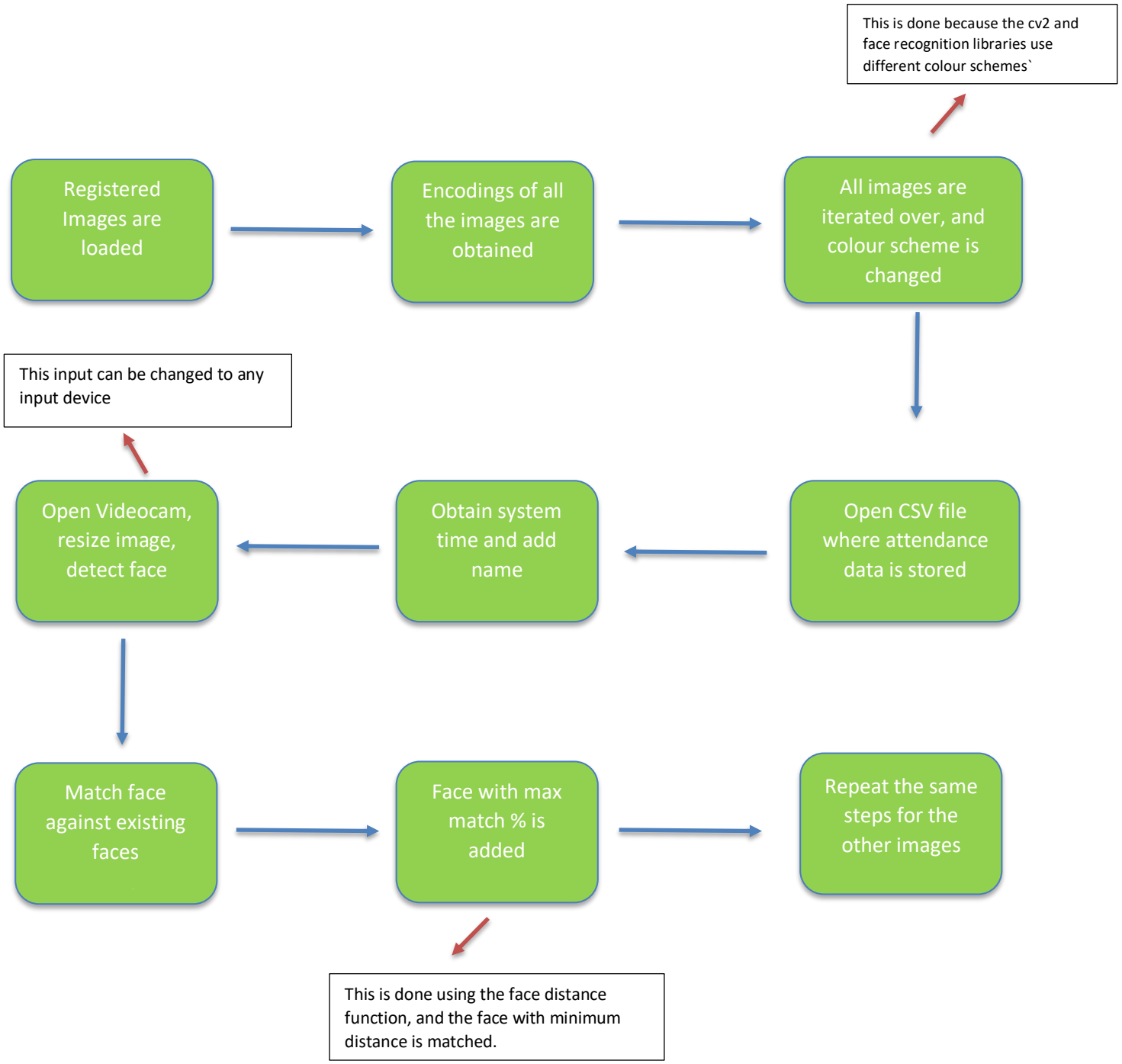
Datetime:

This package is used to obtain date and time from the system. Data is obtained in Hour-Min-Sec format. It is then added along with the face name when a user comes to mark his attendance.

Functions used:

1. datetime.now() – This function is used to return the current time of the system. This will help us to log the attendance of the employees according to the system set date and time.
2. now.strftime(): Used to obtain time of the system in the desired format. In our case we return the time in HourMinSec format.

Flow Chart representing the working of the algorithm



CONCLUSION

Facial recognition is one of the best ways to perform mass identification as it does not require the cooperation of the test subject to work. Properly designed systems installed in airports, multiplexes, and other public places can identify individuals among the crowd, without passers-by even being aware of the system. This makes it fit for our purpose of queueing up patients in large hospitals to save time. The one challenge of facial recognition is the viewing angle of the subject. Face recognition has been getting pretty good at full frontal faces and 20 degrees off, but as soon as we go towards profile, problems arise in detection. This is one of the main obstacles of face recognition in surveillance systems. Face recognition is less effective if facial expressions vary. Data privacy is the main concern when it comes to storing biometrics data in companies. Data stores about face or biometrics can be accessed by the third party if not stored properly or hacked. Hackers will be looking to replicate people's faces to trick facial recognition systems, but the technology has proved harder to hack than fingerprint or voice recognition technology in the past.

The actual effectiveness of the technology will be hard to predict without conducting a trial. However, real time facial recognition can significantly reduce the size of queues, reduce patient wait time and speed up the system, paving way for overall modernisation of the system.

As of now, the core functionality of recognising the faces against a given dataset and predicting the results in real time has been implemented. The future objectives for the model are:

1. Improve the efficiency of the model. This is possible by reducing image size or by using multiple cores of the CPU at once, each for running an instance of the program. This highly depends on the platform where the software is to be deployed.
2. A dashboard is to be created to act as UI for the software. This will allow for employees to register to the application as well as allow the authorities to access and monitor attendance data of employees.
3. A DBMS will be used for storing the images as well as the attendance register file of the employees. This will allow the file to be accessed anywhere once it is on the server as well as address data security.

APPENDICES

Python code for implementing facial recognition:

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime

# access the folder with images

path = 'Images'  # the path where the registered images are stored
mylist = os.listdir(path)  # getting a list of the filenames of registered images

# Step 0 -> register images

images = []  # a list for images
classnames = []  # a list for names

for cl in mylist:
    curimg = cv2.imread(f'{path}/{cl}')
    images.append(curimg)  # add new image to the list
    classnames.append(os.path.splitext(cl)[0])  # add new name to the list


# Step 1 ->function to get the encodings of all the images in the folder (all registered
people)
def find_encodings(images):
    encode_list = []  # an empty list for storing the encodings of all the faces
    for img in images:
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # iterate through every image and
change it from BGR TO RGB
        encode = face_recognition.face_encodings(img)[0]  # find the face and get all the
encodings
        encode_list.append(encode)  # add the found encodings to the list
    print('Images Encoded')
    return encode_list


# Step 4  -> Function for marking the attendance
def markAttendance(name):
    with open('Attendance.csv', 'r+') as f:  # open the csv file where attendance data is
stored
        namelist = []  # a list of the names already in the file
        myDataList = f.readlines()
        for line in myDataList:
            entry = line.split(',')
            namelist.append(entry[0])
        if name not in namelist:  # a check so that there is no discrepancy in the file
            now = datetime.now()
            dtString = now.strftime('%H:%M:%S')  # get the current data in HourMinSec
format in string data type
            f.writelines(f'\n{name},{dtString}')  # mark the attendance for the new face
detected


encodelistKnown = find_encodings(images)
# Step 2 recognize the face
cap = cv2.VideoCapture(0)  # get the input, in this case from webcam
```

```
while True:
    success, img = cap.read()  # get the images of each frames
    imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)  # resize the image to 0.25 of its
size for faster computations
    imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

    faceLoc = face_recognition.face_locations(
        imgS)  # detect the face with help of face recognition library and store its
location
    encodes = face_recognition.face_encodings(imgS, faceLoc)  # also get the encodings
found at the face location

    for encode, face in zip(encodes, faceLoc):
        matches = face_recognition.compare_faces(encodelistKnown,
                                                 encode)  # match the current image frame
with the already registered faces
        facedis = face_recognition.face_distance(encodelistKnown,
                                                 encode)  # get the distance(difference) of
each face from registered faces

        matchindex = np.argmin(
            facedis)  # get the argument of the registered face with minimum difference
from the current face
        if matches[matchindex]:
            name = classnames[matchindex].upper()  # convert the name to uppercase
            y1, x2, y2, x1 = face
            y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4  # re-scale the face to its
original size

            cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)  # draw a rectangle
around the face
            cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0), cv2.FILLED)
            cv2.putText(img, name, (x1 + 6, y2 - 6), cv2.FONT_HERSHEY_PLAIN, 1,
                        (0, 0, 0))  # display the name at the bottom

            markAttendance(name)  # mark the attendance of the detected face

    cv2.imshow('Webcam', img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        cap.destroyAllWindows()
        break
```

REFERENCES

- https://link.springer.com/book/10.1007/978-0-85729-932-1
- https://en.wikipedia.org/wiki/Facial_recognition_system#Advantages_and_disadvantages
- https://www.eff.org/pages/face-recognition
- https://www.thalesgroup.com/en/markets/digital-identity-and-security/government/biometrics/facial-recognition
- https://searchenterpriseai.techtarget.com/definition/facial-recognition
- https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/facial-recognition.htm
- https://www.interpol.int/en/How-we-work/Forensics/Facial-Recognition

GLOSSARY

Deep learning - Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabelled

Face Identification- The process of identifying a face with an identity stored in the database.

Face Detection- The process of detecting the presence of a face in a picture or a video.

Haar Cascade- Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video and based on the concept of features

Cascade Function- A cascade function starts from values on a coarse sequence of sampling points and produces values for successively more densely spaced sequences of sampling points in an iterative manner.

Adaboost Function- AdaBoost, short for Adaptive Boosting, is a machine learning meta-algorithm used to improve the performance of other algorithms.

Positive Image- An image which contains a face.

Negative Image- An image which does not contain a face.

Features- The features are different parts of a face like the eyes, foreheads, mouth, chin etc.