CE-712: Digital Image Processing of Remotely Sensed Data

Project Report

Name:-	**Vachika Maripi** **Saurabh Kumar**	**Roll No:-**	**140040081** **16d100018**
Department	**Civil Engg.** **Mechanical Engg.**	**Program:-**	**B.Tech.** **Dual Degree**

Face Recognition For The Department

Objective: To locate individuals in a given frame and label them comparing with a given database.

Data Used: No data required for training purpose, as we have used python's inbuilt face_recognition library. For the deployment of model, sample images of individuals who need to be identified is to be provided and added in a database.

Methodology:

Training Section:

As we have studied in our course, image is just a combination of numbers in matrix form.

Black and white image can be represented as a single matrix of dimension equal to pixel capacity of the source with which it is captured(say 24 megapixel, implies total number of pixels in the image is 24 million. So the product of number of rows and columns in the matrix with which it is represented equals mega capacity of source).

For coloured images, we have three channels, one each for red, blue and green, as represented below:

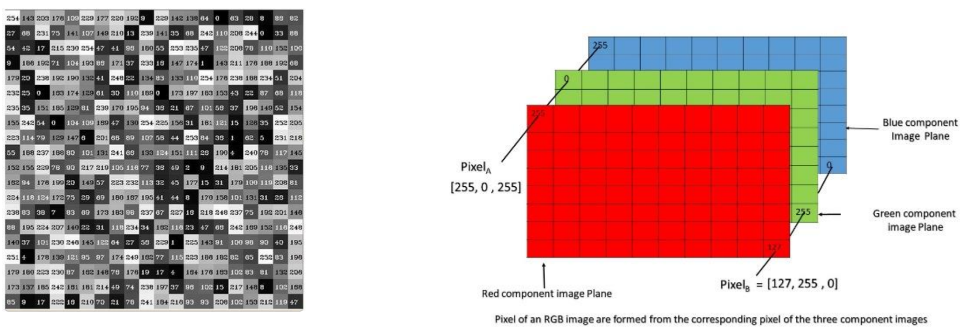

255
0
Pixel$_A$
[255, 0, 255]
Blue component Image Plane
0
Green component image Plane
255
127
Red component image Plane
Pixel$_B$ = [127, 255, 0]
Pixel of an RGB image are formed from the corresponding pixel of the three component images

Fig: Grey-scale and RD+GB representation of image

In complex scenarios such as face recognition, we need to come up with metrics which can locate instances of face in given frame and then encode the identified faces into some metric which can be used for comparision between faces and compare similar encodings of new faces with the face encodings that the system already have. So the task of face recognition splits primarily into three broader objectives:

1) **Locating faces in a given frame**

Firstly the image is imported by providing the location of the image. Then the picture is transformed from RGB to Grayscale because it is easy to detect faces in the grayscale.

Fig: RGB to grey scale

After that, the image manipulation used, in which the resizing, cropping, blurring and sharpening of the images is done if needed. The next step is image segmentation, which is used for contour detection or segments the multiple objects in a single image so that the classifier can quickly detect the objects and faces in the picture.

The next step is to use Haar-Like features algorithm, which is proposed by Voila and Jones for face detection. This algorithm used for finding the location of the human faces in a frame or image. All human faces shares some universal properties of the human face like the eyes region is darker than its neighbour pixels and nose region is brighter than eye region.

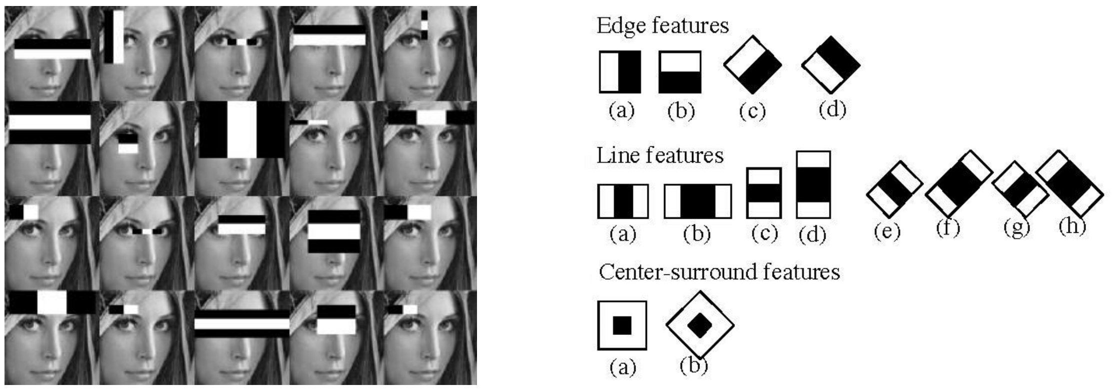


Fig: Haar-like features for face detection

The haar-like algorithm is also used for feature selection or feature extraction for an object in an image, with the help of edge detection, line detection, centre detection for detecting eyes, nose, mouth, etc. in the picture. It is used to select the essential features in an image and extract these features for face detection.
The next step is to give the coordinates of x, y, w, h which makes a rectangle box in the picture to show the location of the face or we can say that to show the region of interest in the image. After this, it can make a rectangle box in the area of interest where it detects the face. There are also many other detection techniques that are used together for detection such as smile detection, eye detection, blink detection, etc.

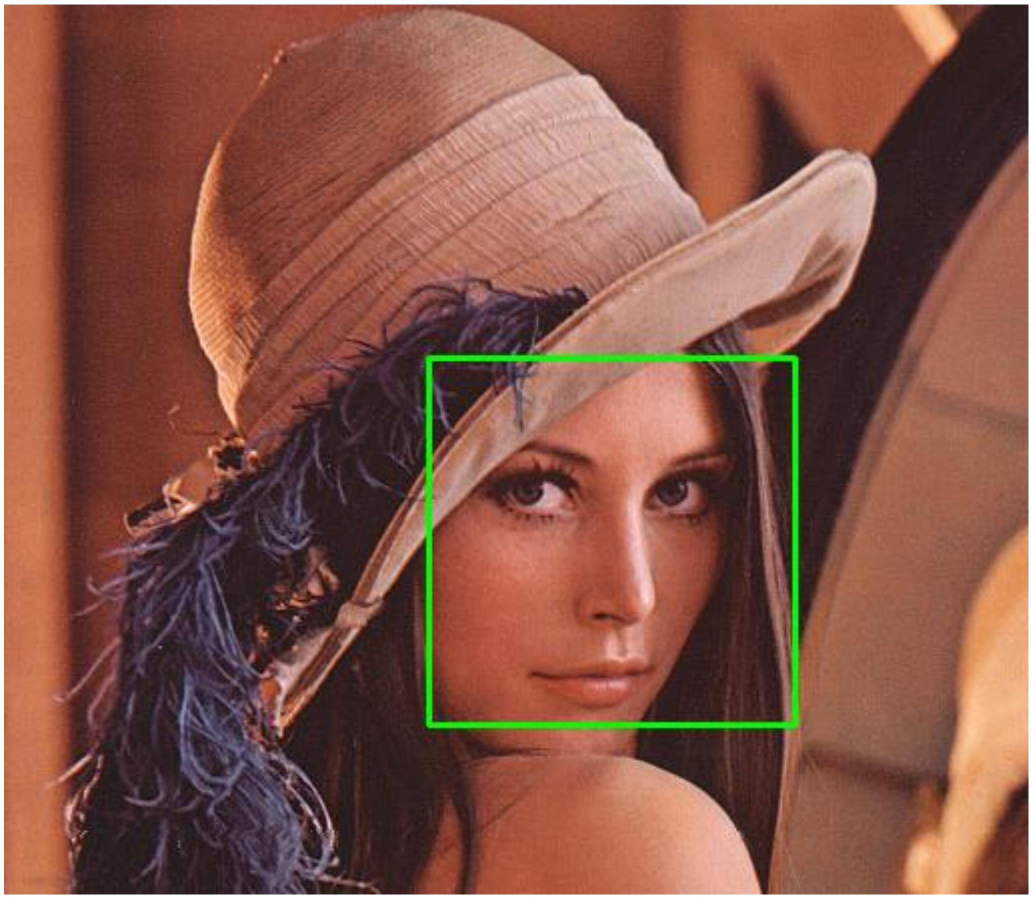


Fig: Face Located

2) **Encoding the identified faces**

So our next job left is to encode the identified faces into some metrics so that they can be compared as and when required.
The encodings can be of various features like:

- Height/width of the face.
- Height and width may not be reliable since the image could be rescaled to a smaller face. However, even after rescaling, what remains unchanged are the ratios – the ratio of height of the face to the width of the face won't change.
- Color of the face.

- Width of other parts of the face like lips, nose, etc.

Essentially, given an image, we can map out various features and convert it into a feature vector like:

Height of face (cm)	Width of face (cm)	Average color of face (RGB)	Width of lips (cm)	Height of nose (cm)
23.1	15.8	(255, 224, 189)	5.2	4.4

So, our image is now a vector that could be represented as (23.1, 15.8, 255, 224, 189, 5.2, 4.4). Of course there could be countless other features that could be derived from the image (for instance, hair color, facial hair, spectacles, etc). However, for the example, let us consider just these 5 simple features.
Now, once we have encoded each image into a feature vector, the problem becomes much simpler.

3) **Comparing the encoded faces**
Once we have encodings of all the faces(one if only one is present), then various algorithms can be used to discover the individual who has the nearest measurement to our sample picture in our database of recognized individuals. We need an algorithm to compare set of two different matrix vectors and then decide how closely they are related with each other. If the correlation between them is above a certain specified threshold then we can say that the individual is know and label it with certain pre-decided variables/names else classify and label them as unknown.

So with the above steps of locating, encoding and classifying faces in a given frame, we have achieved our objective of face recognition. Implementing the face recognition model from scratch would have required lot of training samples and would have been difficult to complete within the given timeframe. So, we have used python's inbuilt face_recognition library for the face recognition purpose.

Model usage details:

Code Link:

- Jupyter notebook format: https://drive.google.com/file/d/1o-baGXR0ypPInTQCtCpBRxL-SjfhcMRt/view?usp=sharing
- Python(.py file): https://drive.google.com/file/d/1WDj3LewPIkO2mZUMQ9YrTX2rRmyG1b7M/view?usp=sharing

Libraries Requirement:

- Os
- face_recognition
- CV2
- Numpy

Steps to use the code:

- Store images of known people in any directory
- Provide path of the directory in variable 'path'
- Run the model

Specific requirements:

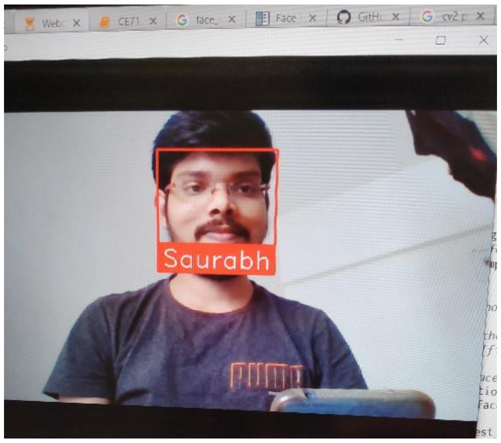
- Image name in the directory should be person's name and the file format should be either .jpg or .png or any other format of three character, else remaining all characters excluding tha last four characters of the image file name will be printed.

The results obtained from the model is shown in next section.

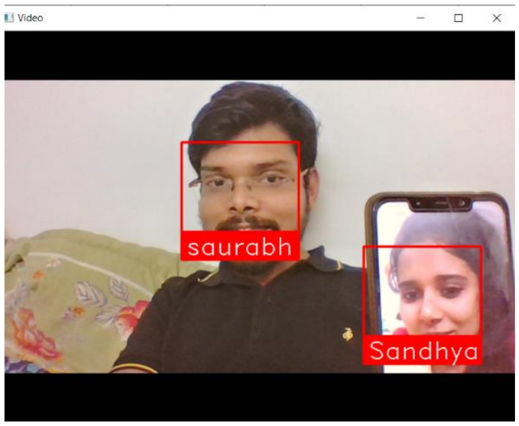
Results:

Face Recognition model in action:

The model does a good job of identifying me.



During video calling:



The model does a good job of identifying people and is robust against screen reflections and all.

Discussion:

The model can be used for security system where access can be provided to only those individuals who are registered for the facility. Also the system can be used for marking attendance given a well separated image of students attending a particular lecture. Also it can be used for identying certain individuals in public areas for preventing public from known criminals, by some team monitoring the output label of the model.

Scope of Improvemnet:

The model fails when images in the database is of young age. So, age-robustness can be introduced in the model.