Face Recognition System using OpenCV

Introduction

Face recognition technology has gained significant attention due to its diverse applications in security, access control, surveillance, and user identification. This report focuses on the implementation of a face recognition system using OpenCV, a popular computer vision library.

The system utilizes a training phase which involves extracting faces from a labelled dataset and training a model using OpenCV's LBPHFaceRecognizer algorithm. The trained model is then utilized for face recognition on new images.

1. Face Training

Faces are extracted from the training images, then labels (person names) are assigned to the faces, and face recognition model is trained using OpenCV's LBPHFaceRecognizer algorithm.

1. The code starts by importing the necessary libraries and defining the directory path to the training images (**DIR**).

```
import os
import cv2 as cv
import numpy as np
```

2. Initialize an empty list called **people** to store the names of the folders (people) inside the training directory. The code iterates over each folder (person) in the training directory and appends the folder name to the **people** list.

```
DIR = r'Photos_Train'
people = []

for i in os.listdir(DIR):
    people.append(i)
```

3. Two empty lists, **features** and **labels**, are created to store the extracted faces and their corresponding labels (person names).

```
features =[]
labels = []
```

4. The function **create_train** is defined to populate the **features** and **labels** lists. It iterates over each person folder and each image within that folder.

```
for person in people:
    path = os.path.join(DIR,person)
    label = people.index(person)

    for img in os.listdir(path):

        img_path = os.path.join(path,img)
        img_array = cv.imread(img_path)
        gray = cv.cvtColor(img_array,cv.COLOR_BGR2GRAY)

        haar_cascade = cv.CascadeClassifier('haar_face.xml')
        faces_rect = haar_cascade.detectMultiScale(gray, scaleFactor = 1.5, minNeighbors = 3)

        for (x,y,w,h) in faces_rect:
            face_roi = gray[y:y+h, x:x+h]

            features.append(face_roi)
            labels.append(label)
```

For each image, it reads the image, converts it to grayscale, and uses the Haar cascade classifier (**haar_cascade**) to detect faces.

5. Detected faces are stored in the **features** list, and their corresponding labels (person index) are stored in the **labels** list.

```
create_train(features,labels)
```

6. After iterating through all the images, the **features** and **labels** lists are converted to numpy arrays for easier handling.

```
features = np.array(features,dtype='object')
labels = np.array(labels)
```

7. The LBPH (Local Binary Patterns Histograms) face recognizer is created using **cv.face.LBPHFaceRecognizer_create()**. The face recognizer is trained using the **train** method, taking the **features** and **labels** arrays as input.

```
face_recognizer = cv.face.LBPHFaceRecognizer_create()
face_recognizer.train(features,labels)
```

8. The trained model is saved to a file named "face_trained.yml", and the **features** and **labels** arrays are also saved as numpy arrays for future use. Finally, a message "training done" is printed to indicate that the training process is completed.

```
face_recognizer.save('face_trained.yml')

np.save('features.npy', features)
np.save('labels.npy', labels)

print('training done')
```

3. Face Recognition

The code allows you to recognize faces in new images by utilizing a trained face recognition model. This is beneficial in scenarios where you want to identify individuals from a dataset of known faces. The face recognition system can be used for various applications, such as access control, surveillance, attendance systems, and personalized user experiences.

1. The code imports the necessary libraries and defines the directory path to the images to be tested (**DIR_1**).

```
DIR = r'Photos_Train'
DIR_1 = r'Photos_Check'
```

2. The **people** list, created during training, is used to store the names of the people (folders) inside the training directory.

```
people = []

for i in os.listdir(DIR):
    people.append(i)
```

3. The previously trained model is loaded using **cv.face.LBPHFaceRecognizer_create()** and **read()** method.

```
face_recognizer = cv.face.LBPHFaceRecognizer_create()
face_recognizer.read('face_trained.yml')
```

4. The code iterates over each image in the test directory. For each image, it reads the image, converts it to grayscale, and uses the Haar cascade classifier (**haar_cascade**) to detect faces.

```
haar_cascade = cv.CascadeClassifier('haar_face.xml')
```

Detected faces are then passed to the trained face recognizer using **predict()** to obtain the predicted label (person index) and confidence level. The predicted label is converted to the corresponding person name from the **people** list.

```
features = np.load('features.npy', allow_pickle=True)
labels = np.load('labels.npy')
```

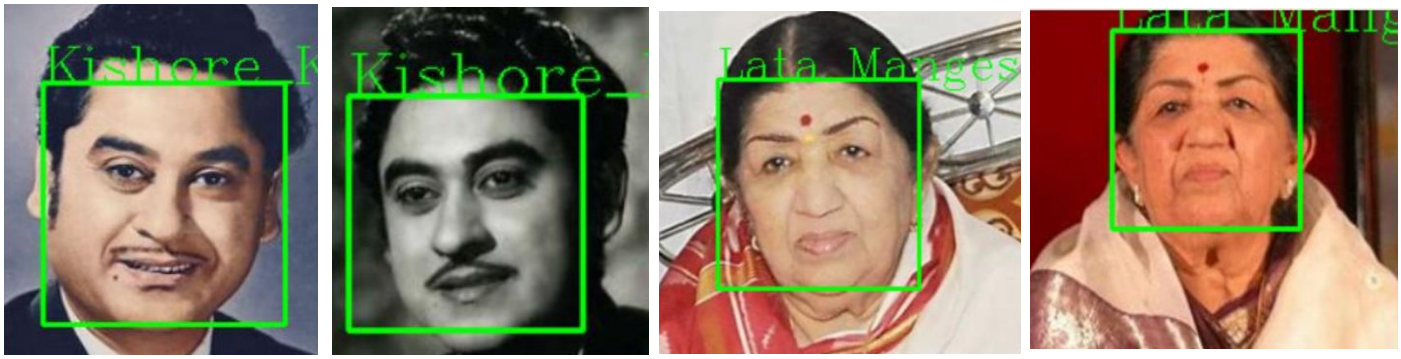
The person's name and confidence level are printed, and the recognized face is marked on the image using rectangles and text.

```
for j in os.listdir(DIR_1):
    img_path = os.path.join(DIR_1,j)
    img = cv.imread(img_path)
    gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

    face_rect = haar_cascade.detectMultiScale(gray,1.05,4)

    for(x,y,w,h) in face_rect:
        face_roi = gray[y:y+h, x:x+h]
        label, confidence = face_recognizer.predict(face_roi)
        print(people[label],confidence)
        cv.putText(img,str(people[label]),(x,y),cv.FONT_HERSHEY_COMPLEX,1.0,(0,255,0),thickness=0)
        cv.rectangle(img,(x,y),(x+w,y+h),(0,255,0),thickness=2)

    cv.imshow(j,img)    #   visualising the image
```

(function) FONT_HERSHEY_COMPLEX: Any

5. The image is displayed until a key is pressed.

```
cv.waitKey(0)
```



4. Conclusion

The face recognition system implemented using OpenCV has demonstrated its capability to accurately recognize faces in new images. The system's training phase effectively extracts facial features and trains a model capable of recognizing individuals. The face recognition phase successfully identifies known individuals from the testing dataset. However, further improvements can be made to enhance the system's accuracy and performance. Overall, the implemented system showcases the potential and significance of face recognition technology in various real-world applications.