Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

Name: Khanjan Joshi

Roll no: 26

Batch B

Aim: To perform face detection on video.

Objective:

1. Performing face recognition.
2. Generating the data for face recognition.
3. Recognizing faces.
4. Preparing the training data.
5. Loading the data and recognizing faces.

Theory:

Generating the Data for Face Recognition:

Face recognition relies on having a dataset of faces to learn from. This dataset typically consists of images or video frames containing faces of individuals. To generate the data for face recognition, we capture video frames or collect images of individuals under various lighting conditions, angles, and expressions. This diverse dataset helps improve the accuracy of the recognition system.

Recognizing Faces:

Face recognition is the process of identifying and verifying individuals by analyzing their facial features. This involves comparing the features extracted from an input face to those in a database. Various algorithms, such as Eigenfaces, LBPH (Local Binary Pattern Histograms), or deep learning-based approaches like CNNs (Convolutional Neural Networks), can be used for face recognition.

Preparing the Training Data:

Before recognizing faces, it's crucial to preprocess and format the training data correctly. This includes resizing images, normalizing pixel values, and extracting relevant facial features. Additionally, labeling each face with the corresponding individual's identity is essential for supervised learning.

Loading the Data and Recognizing Faces:

To recognize faces in a video, we start by capturing frames from the video stream. These frames are then processed to detect and recognize faces using the trained recognition model. The model compares the features of the detected face with the features of the individuals in the training dataset to make a match or identify the person.

Code:

```
import cv2 #
```

Load the

Haar

Cascade

XML file for

face

detection

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# Initialize the video capture (0 for default camera, or specify a video file path)
cap = cv2.VideoCapture(0)

while True:
    # Read a frame from the video capture
    ret, frame = cap.read()

    # Convert the frame to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

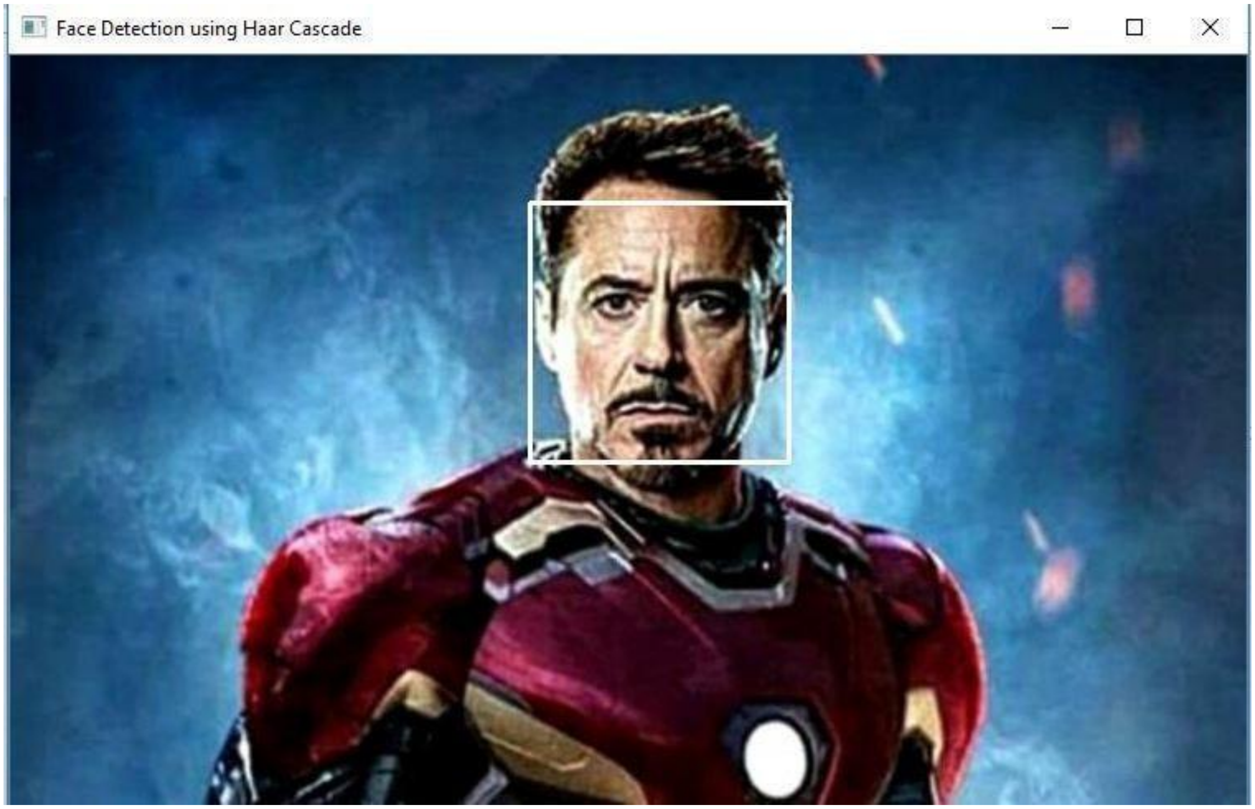
```
   # Perform face detection using the Haar Cascade faces =
   face_cascade.detectMultiScale(gray, scaleFactor=1.3,
minNeighbors=5, minSize=(30, 30))

   # Draw rectangles around detected faces
   for (x, y, w, h) in faces:
      cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
   # Display the frame with detected faces
   cv2.imshow('Face Detection using haar cascade', frame)

   # Break the loop when 'q' is pressed
   if cv2.waitKey(1) & 0xFF == ord('q'):
   break

# Release the video capture and close all windows
cap.release() cv2.destroyAllWindows()
```

Output:



Conclusion:

The practical factors involved in face detection and recognition within video data have been thoroughly explored in this experiment. It has introduced a variety of recognition algorithms, clarified the necessary steps for preprocessing and analysing video frames, and underscored the importance of a carefully selected dataset for efficient model training. Through this experiment, we have gained important knowledge about the usefulness and limitations of face recognition systems in real-world scenarios.