COMPUTER VISION - PROJECT

TITLE OF THE PROJECT : FACE RECOGNITION

Team Members

Oppangi Poojita	B21CS055
Ankana Chowdhury	B21CS010
Keshika Patwari	B21CS039
Eslavath Meenakshi	B21CS028

Prerequisites :

→ The rquired installation need to be performed in google colab.

→ installations required

```
pip install opencv-python
```

→ Before running the .py file, we need to upload the contents of the 'Uploads' folder in the Files section of google colab.

→ Due to the large size of ".ipynb" file and "Uploads" folder , we are not able to upload on github, So here are "colab" and "drive" links respctively:

Uploads link :

https://drive.google.com/file/d/1JLQ8FwVnHV9Ag4L45EudXsU00K50o9nJ/view?usp=sharing

ipynb colab link:

https://colab.research.google.com/drive/1gf4MfZEirmu9zANwu96SjitjyGuG0kJ6?usp=sharing

PROBLEM STATEMENT

Our project aims to use computer vision to detect and identify faces accurately in both images and videos. We want to develop algorithms that can find faces in pictures, including those facing forward and sideways, and in videos, allowing us to spot faces in real-time. It's crucial that our algorithms can tell the difference between faces and other objects in a scene, especially when there's a lot going on in the background.

We were also focused on making sure our face detection is as precise and quick as possible. We want to improve how fast our algorithms can process images and videos without sacrificing the accuracy of the detection. Additionally, we want to provide clear visual representations of the detected faces, like adding labels or annotations, to help understand and analyse the results better.

METHODOLOGY

1. Face Detection using Haar Cascades :

→ We utilise pre-trained Haar cascades, which are classifiers commonly used for detecting objects in images.

→ Specifically, we employ Haar cascades trained for detecting frontal and profile faces.

→ These cascades are XML files containing information about the features of the object being detected.

2. Image Processing for Grayscale Conversion :

→ Before detecting faces, we convert images to grayscale.

→ Grayscale images have only one channel instead of three (RGB), making them computationally simpler to process.

→ This conversion helps improve processing efficiency while retaining essential information for face detection.

3. Overlap Detection to Avoid Duplicate Detections :

→ We implement overlap detection to ensure that detected frontal and profile faces do not overlap, preventing duplicate detections.

→ This involves checking for intersections between the bounding boxes of detected faces.

→ If an overlap is detected, we discard one of the overlapping detections to avoid redundancy.

→ Implement algorithms for age and gender detection in addition to face recognition.

4. Video Processing for Real-time Detection :

→ Extending face detection to videos involves processing frames one by one.

→ For each frame in the video stream, we apply the face detection algorithm to identify faces.

→ This allows us to achieve real-time face detection in videos, providing instantaneous results as the video plays.

5. Visualisation of Detected Faces :

→ Once faces are detected, we visualise them on images and videos using rectangles and text annotations.

→ Rectangles are drawn around the detected faces to highlight their locations within the images or video frames.

- Text annotations, such as labels indicating the type of face (frontal or profile) and a numbering system for multiple faces, provide additional information for analysis.

6. Performance Optimization for Speed and Accuracy :

→ We aim to optimise the performance of our face detection algorithms for better speed and accuracy.

→ This may involve fine-tuning parameters such as the scale factor and minimum neighbours in the Haar cascade classifiers.

→ Additionally, we explore potential optimizations to reduce computational overhead and improve overall efficiency without compromising detection accuracy.

7. Face Matching in Group Images:

→ Face matching in group images involves comparing a known individual's facial features with multiple faces in a group image.

→ The process includes detecting faces in the group image and comparing them with the known individual's features using similarity metrics.

→ Histogram-based methods or deep learning-based approaches may be employed for feature comparison.

→ If a match is found above a certain threshold, the known individual is identified within the group image.

8. Thresholding and Confidence Calculation:

→ Thresholding is a critical step in face recognition and matching to determine the similarity threshold above which a match is considered valid.

→ Confidence calculation involves quantifying the certainty of a match based on the similarity between facial features.

→ Techniques such as histogram comparison or deep metric learning may be used to calculate confidence scores.

→ The threshold and confidence scores help in determining the accuracy and reliability of face recognition and matching results.

9. Integration and Scalability:

→ Integrating face recognition and matching capabilities into the existing framework requires seamless integration with the face detection and visualisation components.

→ The system should be scalable to handle a large number of known individuals and group images efficiently.

→ Considerations for computational resources, such as memory and processing power, are essential for ensuring scalability and performance.

RESULTS

Result - 1

The code displays images with rectangles drawn around detected faces, distinguishing between frontal and profile faces.

Observations:

→ Face detection successfully identifies both frontal and profile faces in images.

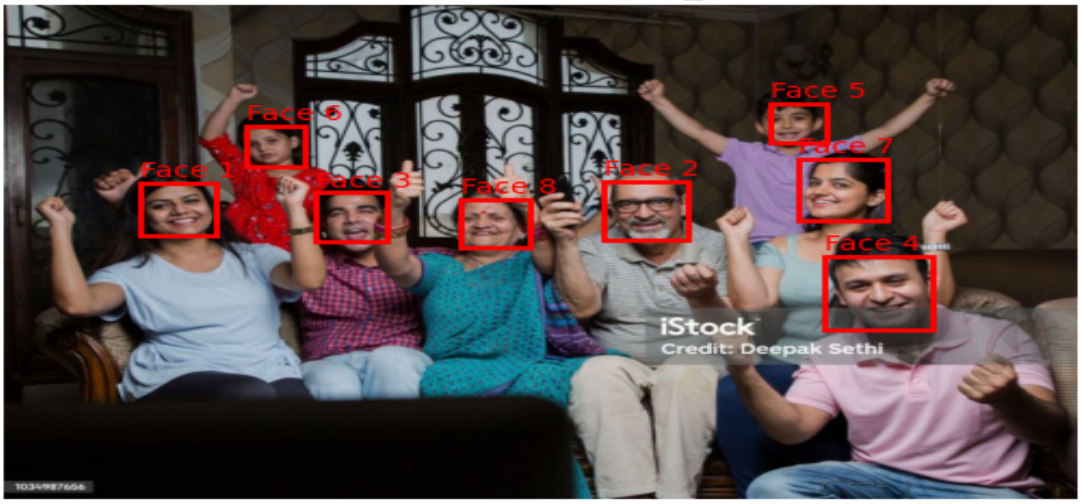
→ Detected faces are highlighted with rectangles and labelled appropriately.

→ The distinction between frontal and profile faces is clear in the visual representation.

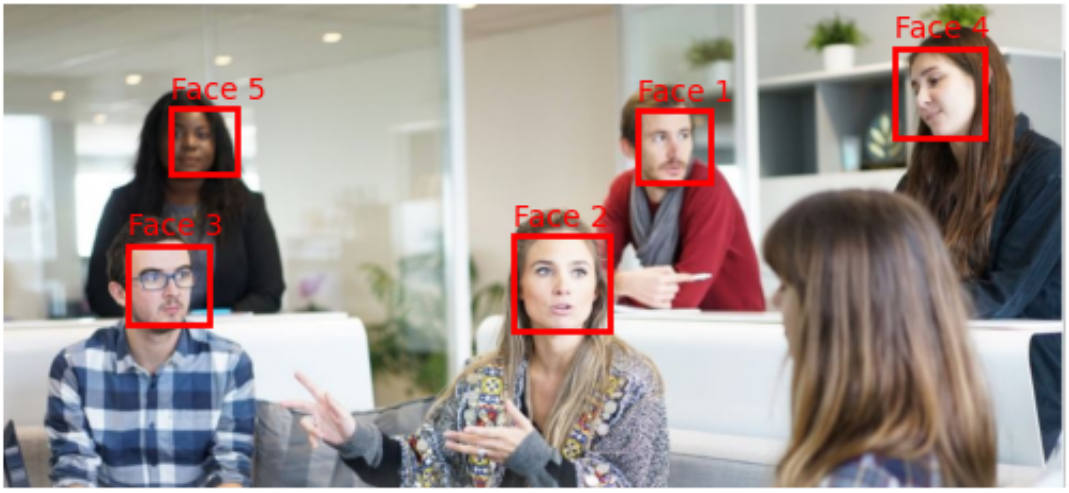
→ Provides a clear indication of the processing time for each image.

Conclusion: The code effectively detects and distinguishes between frontal and profile faces in images.

Detected faces in group_ppl.JPG



Detected faces in face_img.JPG



Detected faces in street_scene.JPG

Result - 2

The code displays original images, grayscale images, and images with detected faces side by side.

Observations:

→ Provides a comprehensive view of the original image, grayscale image, and detected faces.

→ Utilises colour-coded rectangles to differentiate between frontal and profile faces.

→ Offers a detailed analysis of face detection in images, including different processing stages.

→ Displays the aspect ratio of each detected face.

Conclusion: This code segment offers a detailed breakdown of face detection in images, providing insights into individual face detection and overall scene analysis.

Result - 3

The code displays original images, grayscale images, individual detected faces, and images with all detected faces highlighted.

Observations:

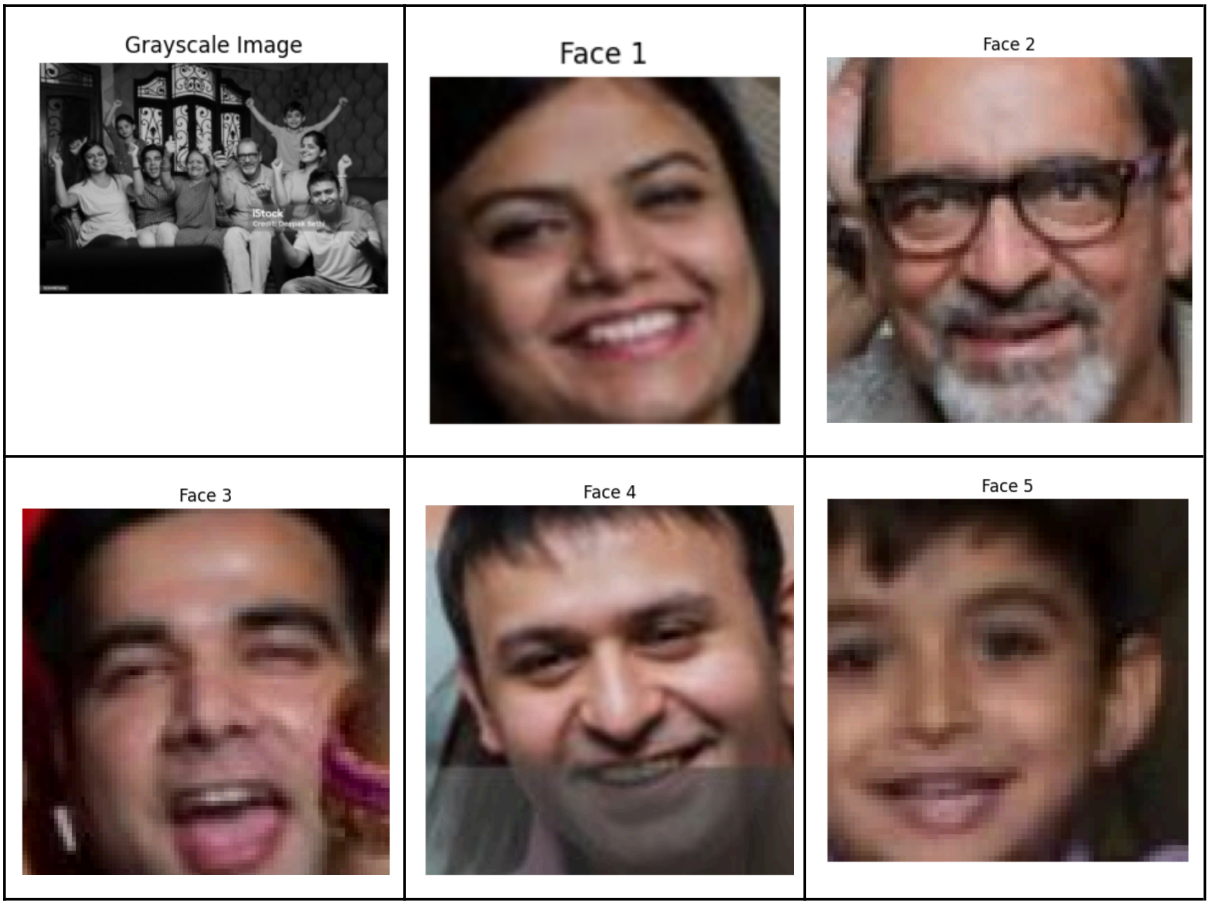
→ Shows individual detected faces separately alongside the original and grayscale images.

→ Highlights all detected faces in the final image for a comprehensive view.

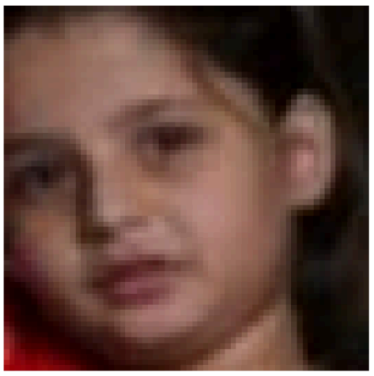
→ Provides detailed insights into face detection in images, including individual face analysis.

→ Displays the coordinates of each detected face.

Conclusion: Offers a comprehensive analysis of face detection in images, providing detailed insights into individual face detection and overall scene analysis.

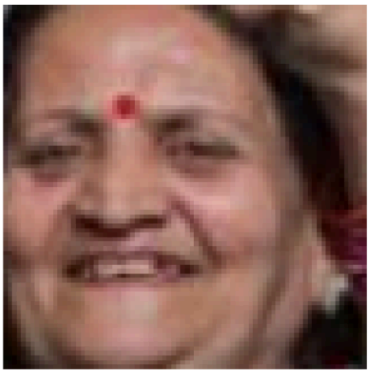


Face 6



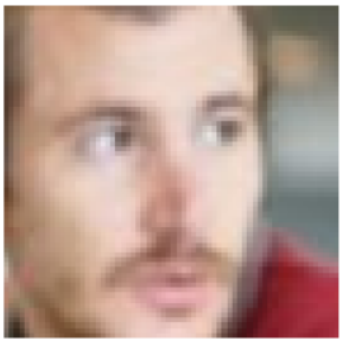
Face 7

Face 8

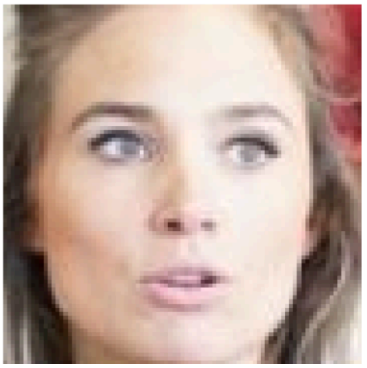


Grayscale Image

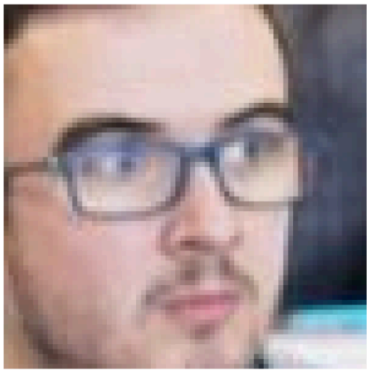
Face 1



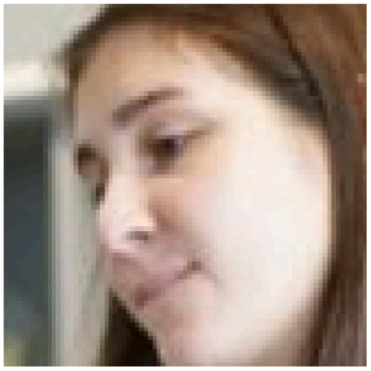
Face 2



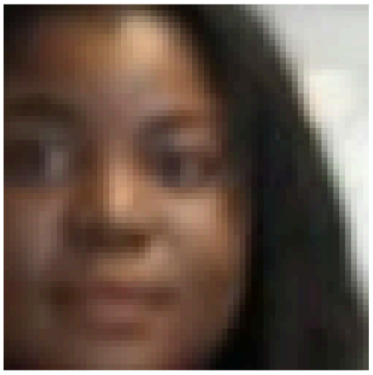
Face 3



Face 4



Face 5



Result - 4

The code detects faces in videos and estimates age and gender.

Observations:

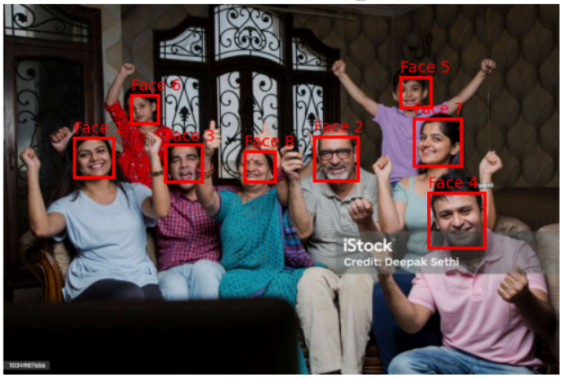
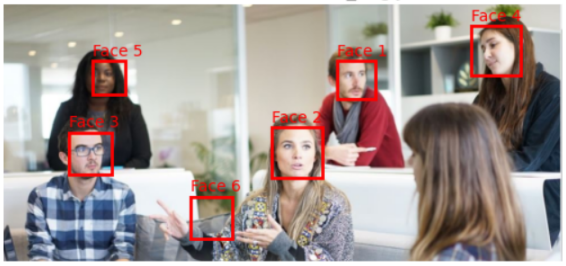
→ Applies face detection to video streams, drawing rectangles around detected faces in real-time.

→ Estimates age and gender for each detected face.

→ Enables live processing of videos for real-time face detection and demographic analysis.

→ Displays the confidence level for each estimated age and gender.

Conclusion: Enables real-time face detection and demographic analysis in videos, suitable for applications requiring live processing.

Detected faces in group_ppl.JPG 	Face 1 - Age: Adult, Gender: Female Face 2 - Age: Adult, Gender: Female Face 3 - Age: Adult, Gender: Female Face 4 - Age: Adult, Gender: Female Face 5 - Age: Young Adult, Gender: Female Face 6 - Age: Young Adult, Gender: Female Face 7 - Age: Adult, Gender: Female Face 8 - Age: Young Adult, Gender: Female
Detected faces in face_img.JPG 	Face 1 - Age: Young Adult, Gender: Female Face 2 - Age: Adult, Gender: Female Face 3 - Age: Young Adult, Gender: Female Face 4 - Age: Adult, Gender: Female Face 5 - Age: Young Adult, Gender: Female Face 6 - Age: Young Adult, Gender: Female
Detected faces in street_scene.JPG 	Face 1 - Age: Child, Gender: Female Face 2 - Age: Young Adult, Gender: Female Face 3 - Age: Young Adult, Gender: Female

Result - 5

The code saves a video with rectangles drawn around detected faces.

Observations:

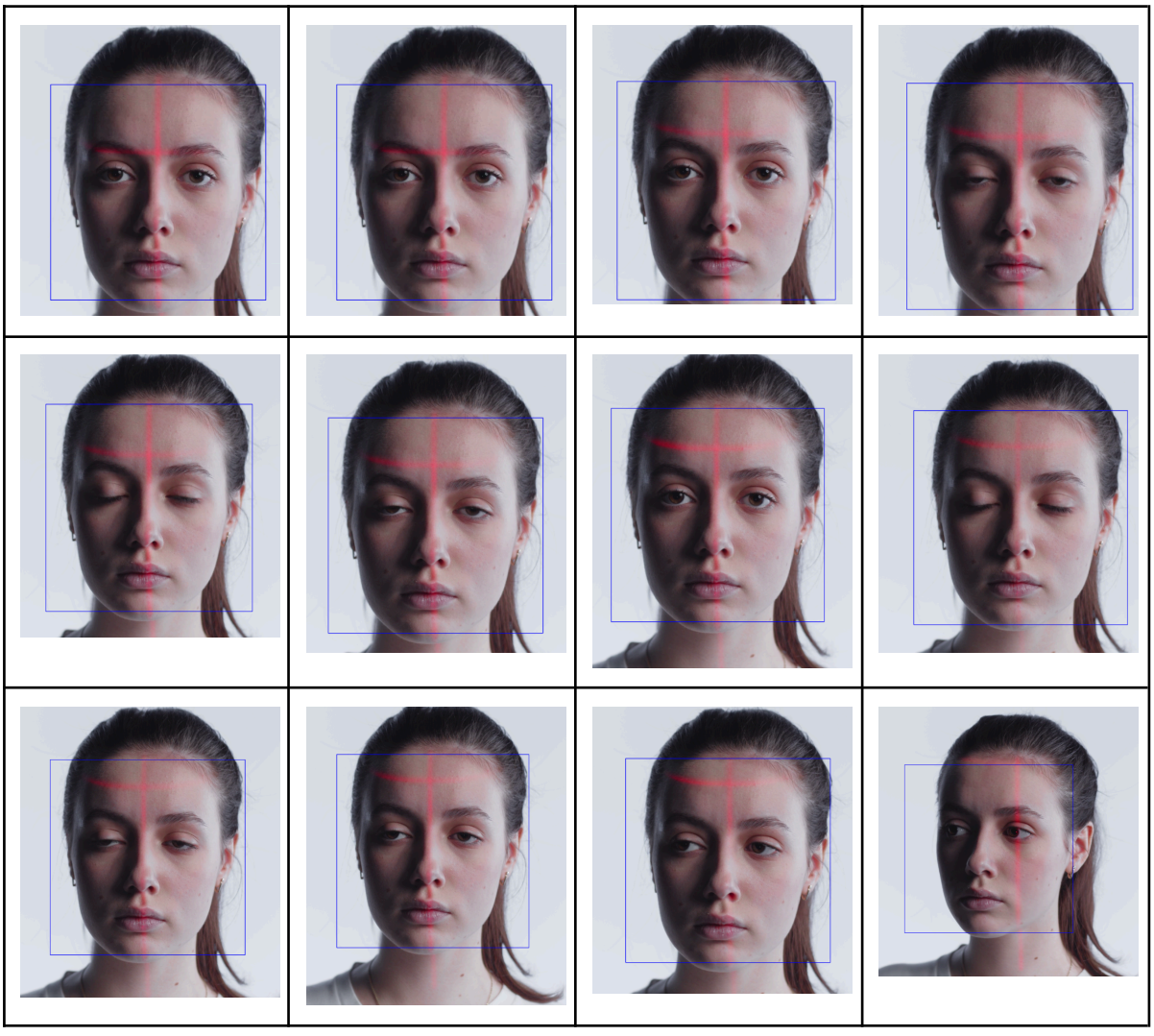
→ Detects faces in a video and saves the processed video with rectangles around detected faces.

→ Provides a practical implementation of face detection in videos, producing a processed video as output.

→ Displays the frame rate of the processed video.

Conclusion: Offers a practical implementation of face detection in videos, providing a processed video as output.

There a lot of images produced which can't be pasted here so, picked a few from them.



Result - 6

The code processes multiple videos in a folder and saves processed videos in an output folder.

Observations:

→ Iterates over all videos in a folder, processes each video for face detection, and saves the processed videos in an output folder.

→ Provides a scalable solution for batch processing multiple videos, suitable for large-scale video processing tasks.

→ Displays the number of videos processed and the processing time for each video.

Conclusion: Provides a scalable solution for batch processing multiple videos, useful for large-scale video processing tasks.

```
Processing video: /content/videos/sample5.mp4
End of video.
Output video saved successfully: /content/output_videos/sample5.mp4
Processing video: /content/videos/sample2.mp4
End of video.
Output video saved successfully: /content/output_videos/sample2.mp4
```

Result - 7

The code recognizes a known face in a group image.

Observations:

→ Loads a known face image and a group image for comparison.

→ Utilises Haar cascades for face detection and histogram comparison for recognition.

→ Draw rectangles around matched faces in the group image.

→ Displays the processing time for face recognition.

Conclusion: Provides a method for identifying a known face within a group image, enabling efficient face recognition tasks.

1)


```
A matching face is found in the group image.
```

2)


```
No matching face is found in the group image.
```

Result - 8

The code performs face recognition on an unknown image using known face images.

Observations:

→ Utilises OpenCV for face detection and recognition.

→ Loads known face images and their corresponding names from a specified folder.

→ Detects faces in the unknown image using Haar cascades.

→ Trains a Fisher Face recognizer on the known faces.

→ Predicts the label (identity) and confidence level for each detected face in the unknown image.

→ Normalizes confidence values to a range of [0, 100].

→ Maps predict labels to corresponding names and print the recognized face along with confidence.

→ Displays the processing time for face recognition.

Conclusion:

Offers a simple yet effective approach for recognizing faces in an unknown image using known face references. The code provides insights into the recognition process, including predicted labels and confidence levels for each detected face.

```
Predicted Label: 3
Predicted Confidence: 2012.2251854756819
Normalized Confidence: 79.87774814524317
Recognized face: john
```

OBSERVATIONS AND CONCLUSION

Observations:

→ The project effectively utilises Haar cascades for face detection, demonstrating proficiency in computer vision techniques. These cascades enable the detection of both frontal and profile faces, showcasing a comprehensive approach to face detection.

→ Across different codes, various aspects of face detection are explored, including different methods for processing images and videos. This demonstrates versatility in addressing different scenarios and requirements.
→ The project showcases efficient video processing capabilities, capable of handling real-time face detection as well as batch processing of multiple videos.

Conclusion:

The project exhibits a strong grasp of computer vision fundamentals, particularly in face detection and video processing. By effectively leveraging Haar cascades, the project demonstrates proficiency in detecting faces in various orientations and scenarios. The exploration of different face detection techniques across multiple codes underscores the project's versatility and adaptability to different applications. Moving forward, potential avenues for improvement could involve further optimization of algorithms for enhanced performance and accuracy in face detection tasks.

REFERENCES

Research papers -

Paul Viola, Michael Jones - Rapid Object Detection using a Boosted Cascade of Simple Feature

Navneet Dalal and Bill Triggs - Histograms of Oriented Gradients for Human Detection