Face Detector and Counter – A Python-Based Project

## About the Project

This project focuses on developing a basic face detection and counting system using Python and OpenCV. The objective is to create a simple yet functional application that captures real-time video from a webcam, detects human faces using a pre-trained model, and displays the number of faces detected on the screen. The project serves as an introductory exploration into computer vision for students and beginners.

## Purpose

The motivation behind this project was to understand the fundamentals of computer vision and object detection. Face detection is a widely used feature in security, social media applications, and human-computer interaction. Implementing a basic face counter helps grasp the underlying concepts in a practical and hands-on manner.

## Tools and Technologies Used

* Programming Language: Python
* Library: OpenCV (Open Source Computer Vision Library)
* Hardware: Standard laptop webcam

## Functional Overview

1. Captures live video stream from the webcam.

2. Detects human faces in each frame using Haar Cascade Classifier.

3. Highlights detected faces with bounding boxes.

4. Displays the count of detected faces in real time.

## Implementation Steps

**Step 1**: Install OpenCV

pip install opencv-python

**Step 2**: Develop the Python Script

python

import cv2

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

video = cv2.VideoCapture(0)

while True:

ret, frame = video.read()

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(gray)

face\_count = len(faces)

for (x, y, w, h) in faces:

cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

cv2.putText(frame, f"Faces: {face\_count}", (10, 30), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

cv2.imshow("Face Detector and Counter", frame)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

video.release()

cv2.destroyAllWindows()

## Observations

**Working Aspects**

* Successfully detects and counts faces in real time.
* Handles multiple face detections per frame.
* Provides immediate visual feedback.

**Challenges Faced**

* Accuracy is affected under poor lighting conditions.
* Fast movement or side profiles may reduce detection reliability.
* Haar Cascade is not the most robust detection method for all angles.

## Key Learnings

* Basic understanding of computer vision workflows.
* Hands-on experience with OpenCV and Haar cascades.
* Gained familiarity with processing video frames and rendering outputs.

## Future Enhancements

* Replace Haar Cascade with deep learning-based detectors like YOLO or SSD for improved accuracy.
* Add facial feature recognition (eyes, smiles).
* Implement data logging for face counts.

## Conclusion

This project was a valuable introduction to face detection using Python. It reinforced core programming concepts and introduced essential ideas in computer vision. With further exploration, this system can be extended into more sophisticated applications for surveillance, attendance, or interactive systems.

**Report Submitted by Naveen Karthik M**