PROJECT REPORT ON

COUNT NUMBER OF FACES

Submitted By

Tirumalasetty Maneesha - R180693

Kata Devaraj - R180960

Under The Guidance Of:

S Rajeswari (Guest Faculty)

August, 2023

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

R.K Valley, Vempalli(M), Kadapa(Dist) – 516330

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES RGUKT-RK VALLEY

Vempalli, Kadapa, Andhra pradesh – 516330.

CERTIFICATE OF PROJECT COMPLETION

This is to certify that I have examined the thesis entitled

"**Count Number of Faces**" submitted by **Tirumalasetty Maneesha (R180693)** under our guidance and supervision for the partial fulfillment for the degree of Bachelor of Technology in computer Science and Engineering during the academic session February 2023 –July 2023 at RGUKT RK VALLEY.

Project Guide

S Rajeswari,
Guest faculty, in dept of CSE,
RGUKT – RK VALLEY.

Head of the Department

Mr. N.Satyanandaram,
Lecturer in dept of CSE,
RGUKT – RK VALLEY.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES RGUKT-RK VALLEY

Vempalli, Kadapa, Andhra pradesh –

516330.

DECLARATION

We, **Tirumalasetty Maneesha(R180693), Kata Devaraj (R180960)** hereby declare that the project report entitled "Count number of Faces" done under guidance of

Ms. S Rajeswari is submitted in partial fulfillment for the degree of Bachelor of Technology in Computer Science and Engineering during the academic session February 2023 – July 2023 at RGUKT-RK Valley. I also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites are mentioned in the references. To the best of my knowledge, the results embodied in this dissertation work have not been submitted to any university or institute for the award of any degree or diploma.

Date: Tirumalasetty Maneesha –R180693

Place: RK Valley Kata Devaraj– R180960

ACKNOWLEDGEMENT

I would like to express my deep sense of gratitude & respect to all those people behind the screen who guided, inspired and helped me crown all my efforts with success. I wish to express my gratitude to **S Rajeswari** for her valuable guidance at all stages of study, advice, constructive suggestions, supportive attitude and continuous encouragement, without which it would not be possible to complete this project.

I would also like to extend our deepest gratitude & reverence to the Director of RGUKT, RK Valley **Prof. K. Sandyarani** and HOD of Computer Science and Engineering **Mr. N. Satyanandaram** for their constant support and encouragement.

Last but not least I express my gratitude to my parents for their constant source of encouragement and inspiration for me to keep my morals high.

With Sincere Regards,

Tirumalasetty Maneesha - R180693

Kata Devaraj – R180960

ABSTRACT

The aim of this project is to develop an advanced Python application that employs the robust capabilities of the OpenCV library for accurate face counting within images or real-time video streams. This endeavor merges the realms of computer vision and machine learning to achieve efficient and precise face detection, leading to the creation of a versatile tool with diverse applications.

The primary objective of this project is to develop a sophisticated Python application that leverages the capabilities of the OpenCV library to perform accurate and efficient face counting within both static images and real-time video streams. By integrating the realms of computer vision and machine learning, this endeavor aims to create a versatile tool capable of robust face detection, catering to various practical applications.

The project initiation involves setting up the development environment and installing essential libraries, with a particular focus on OpenCV. This ensures a seamless workflow and establishes the foundation for subsequent tasks. The heart of the project lies in the face detection mechanism, where the option to use either pre-trained Haarcascades or explore deep learning through Convolutional Neural Networks (CNNs) offers flexibility and precision. OpenCV's Haarcascades are employed as specialized classifiers capable of accurately detecting an array of objects, including faces.

Index

Introduction:

Motivation:

The motivation behind embarking on this project lies in the captivating realm of computer vision and its potential to revolutionize various aspects of modern life. The ability to harness the power of OpenCV and Python to accurately count faces within images or video streams carries profound implications across diverse fields. From enhancing security systems by efficiently monitoring crowds to enabling data-driven decision-making in social analysis, the project taps into a fundamental human instinct – recognizing faces – and transforms it into a robust technological tool.

In a world increasingly driven by visual data, the need for sophisticated face detection techniques is evident. By immersing oneself in this project, one gains the skills to decipher intricate patterns within images, translating into real-world applications such as smart surveillance, where the ability to count faces swiftly and accurately could bolster public safety measures. Moreover, this project's optional integration of face recognition adds a layer of complexity that reflects the forefront of technological advancements.

The allure of building an application that can analyze imagery and provide tangible insights fuels curiosity and innovation. Beyond its practical utility, the project promises a deep understanding of both traditional computer vision methods and contemporary machine learning techniques, thereby broadening one's skill set and paving the way for further exploration into this dynamic field. This project is an opportunity to traverse the bridge between theory and application, and as such, serves as an inspiring endeavor to push the boundaries of one's knowledge and contribute to the growing landscape of cutting-edge technology.

Objective of Project:

The primary objectives of this project encompass the development of a robust and versatile Python application using the OpenCV library for the purpose of accurate face counting in both static images and real-time video streams. These objectives can be further detailed as follows**:**

Face Detection Mastery: The project aims to achieve a deep understanding of face detection methodologies, ranging from traditional Haarcascades to modern deep learning-based Convolutional Neural Networks (CNNs). By comprehending the intricacies of these techniques, the project seeks to implement a reliable and efficient face detection mechanism. .

Flexible Input Handling: The project aims to create a user-friendly interface or command-line options that allow users to seamlessly choose between analyzing static images and dynamic video streams as input sources. The interface should provide intuitive interactions for various user preferences.

Image Preprocessing Expertise: For static images, the project targets the implementation of effective preprocessing techniques such as resizing, grayscale conversion, and noise reduction. The objective is to optimize image data for subsequent processing, enhancing the accuracy of face detection.

Optional Face Recognition Integration: The project's optional goal is to delve into the realm of face recognition using machine learning models like Eigenfaces, Fisherfaces, or deep learning-based approaches like FaceNet. This integration aims to demonstrate a broader range of applications beyond face counting.

Accurate Face Counting Algorithm: The project strives to develop a sophisticated algorithm capable of systematically processing each frame of the input source. By leveraging the chosen face detection technique, the algorithm seeks to accurately pinpoint and count faces within images or video frames.

Efficient Data Presentation: The project seeks to present the final face count in a user-friendly manner, either through printed notifications or within a graphical user interface. The objective is to provide users with clear and concise information regarding the face count.

Performance Optimization Exploration (Optional): For those interested, the project aims to explore performance optimization techniques to improve processing speed. Techniques such as frame skipping or multi-threading may be investigated to enhance the overall efficiency of the application.

Through these objectives, the project aims to equip participants with a comprehensive understanding of face detection techniques, a practical tool for face counting, and the potential to extend their knowledge into the realms of face recognition and optimization strategies.

Features:

i. Multi-Source Compatibility

ii. Accurate Face Detection

iii. Optional Face Recognition

iv. User-Friendly Interface

V.Preprocessing Techniques

vi. Visual Enhancement

vii.Dynamic Face Counting

viii.Modular Design

REQUIREMENT

ANALYSIS

Requirement Specification:

Hardware Configuration:

Ram	4GB
Hard disk	512 GB
Processor	1.0 GHz

Software Requirement:

i. Python Interpreter

ii. Pygame Library

iii. Text Editor

iv. Windows / Ubuntu OS

Functional Requirements:

1.**Input Selection**: Users can choose static images or real-time video streams for analysis.

2.**Face Detection**: Accurate face detection within frames of the chosen source.
3**.Optional Face Recognition**: Ability to recognize individuals (optional). 4**.Image Preprocessing (Optional)**: Preprocess images for optimal analysis. 5.**Face Detection Visualization (Optional)**: Display bounding boxes around detected faces.
6.**Output Display**: Show frames with face count and overlays.
7.**Performance Optimization (Optional)**: Provide optimization techniques for faster processing.
8.**User Controls**: Start, pause, and stop face counting; adjust settings. 9.**Error Handling**: Informative messages for input errors or processing issues.
10.**Modularity**: Easily extend or modify functionalities.
11.**Documentation:** Comprehensive user guides and explanations.
12.**Educational Content (Optional):** Insights into underlying algorithms and techniques.

By meeting these requirements, the application aims to offer a user-friendly, customizable, and educational tool for accurate face counting and potential recognition.

Non-Functional Requirements:

1. **Performance**:

- The system should maintain smooth real-time visualization even for larger grid sizes and complex scenarios, ensuring a responsive user experience.

2. **Usability and User Experience**:

- The graphical user interface should be intuitive and user-friendly, catering to users with varying levels of technical expertise.

3. **Compatibility and Cross-platform Support**:

- The system should work seamlessly across different operating systems (Windows, macOS, Linux) without significant variations in functionality or performance.

4. **Accessibility**:

- The system should adhere to accessibility guidelines, ensuring that users with disabilities can interact with and comprehend the visualization.

5. **Security and Data Privacy**:

- The system should not collect or store user data or personal information, ensuring user privacy and data security.

Project Plan:

For a successful software project, the following steps can be followed: • Select a project

- ▪ Identifying project's aims and objectives
- ▪ Understanding requirements and specification ▪ Methods of analysis, design and implementation ▪ Testing techniques
- ▪ Documentation.

• Project milestones and deliverables

• Budget allocation

▪ Exceeding limits within control

• Project Estimates

- ▪ Cost
- ▪ Time
- ▪ Size of code

- Duration

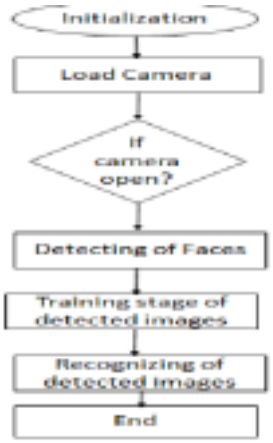
• Resource Allocation
  - Hardware
  - Software
  - Previous relevant project information

• Risk Management
  - Risk avoidance
  - Risk detection

System Architecture:

Context Diagram:

A context diagram is a visual representation that shows the system you're focusing on as a single entity, surrounded by its external interfaces. It helps to illustrate the interactions between the system and its environment without diving into the internal details. Context diagrams are commonly used in software development, systems analysis, and business process modeling to provide a high-level overview of a system's interactions with external entities.

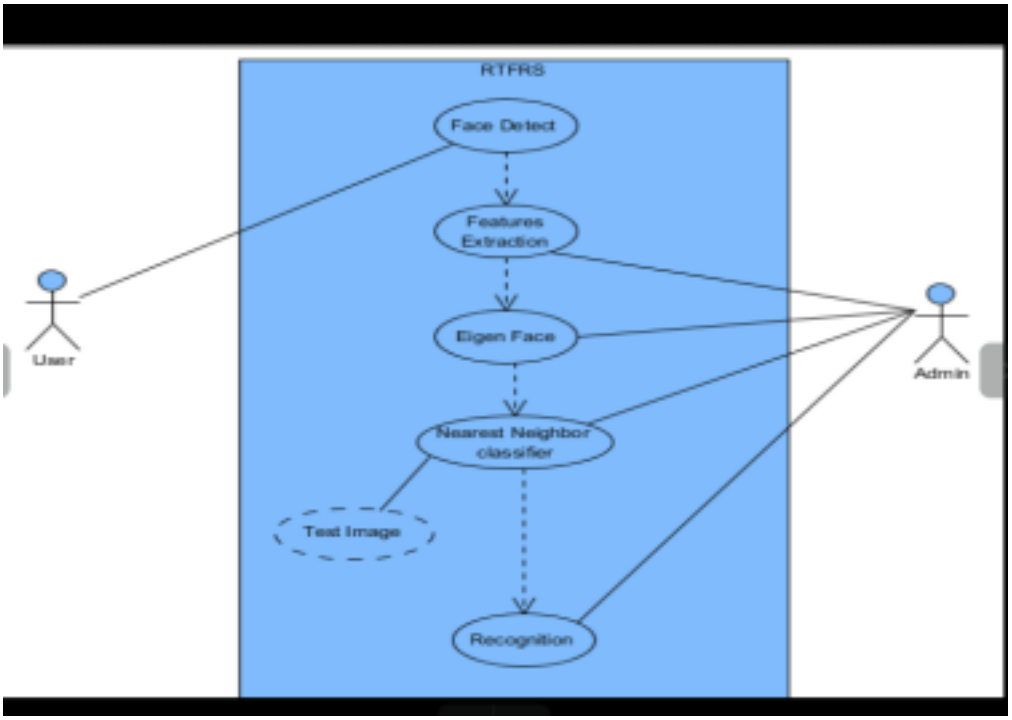


Use Case Diagram:

A use case diagram is a type of diagram used in software development and

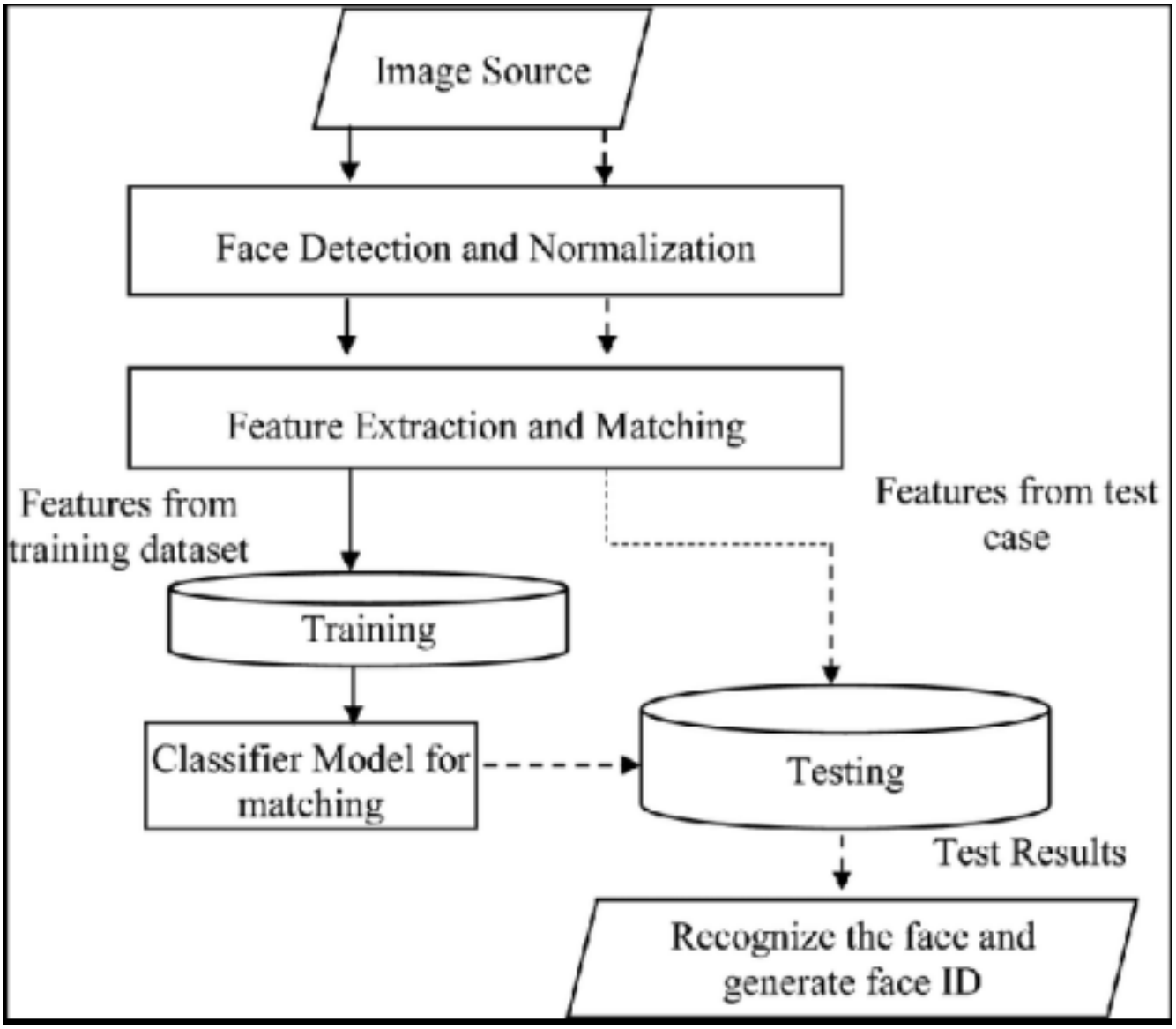
systems engineering to visualize the interactions between users (actors) and a system's functionalities (use cases). It provides a high-level view of how users interact with the system and the specific tasks the system can perform in response.

In a use case diagram, actors are represented as stick figures, and use cases are depicted as ovals. Arrows are used to show the communication between actors and use cases, indicating which functionalities the actors can access. It helps stakeholders understand the overall behavior and requirements of the system from a user's perspective. Use case diagrams are valuable tools for requirements analysis, project planning, and communication between project teams and stakeholders.



Class Diagram:

Class diagrams are the blueprints of your system or subsystem. You can use class diagrams to model the objects that make up the system, to display the relationships between the objects, and to describe what those objects do and the services that they provide. Class diagrams are useful in many stages of system design.



Count Number of Faces:

At the heart of the project lies the flexibility for users to choose between static images and real-time video streams as input sources. This adaptability ensures that the application can cater to a diverse array of scenarios, from analyzing crowd behavior to monitoring security systems.

Central to the project's functionality is its advanced face detection mechanism. By harnessing the power of pre-trained Haarcascades or even exploring the depths of deep learning through Convolutional Neural Networks (CNNs), the application accurately identifies faces within each frame of the chosen source. This process is not only robust but also serves as a foundation for precise face

counting.

For those seeking more advanced capabilities, the project integrates an optional layer of face recognition. This cutting-edge feature extends the application beyond mere counting and opens the door to recognizing individuals, adding a layer of sophistication to its analysis.

For those seeking more advanced capabilities, the project integrates an optional layer of face recognition. This cutting-edge feature extends the application beyond mere counting and opens the door to recognizing individuals, adding a layer of sophistication to its analysis.

To enhance user interaction and understanding, the project can implement face detection visualization. Through this feature, bounding boxes are superimposed around the detected faces, providing real-time visual feedback of the face detection process and making the technology more accessible to users.providing real-time visual feedback of the face detection process and making the technology more accessible to users.

The application's real-time face counting functionality dynamically tallies the number of detected faces, providing an instantaneous and ongoing count that is visually displayed alongside the frames of the input source.

To ensure that users receive the results in their preferred format, the project offers customizable output options. Users can choose between a graphical user interface that displays the visualized results or printed notifications, according to their convenience.

In recognition of the importance of performance, the project presents the opportunity for performance optimization. Techniques like frame skipping or multi-threading can be explored to enhance the responsiveness of the application, making it suitable for real-time scenarios.

l.

Pseudocode for Count Number of faces

```
# Import required libraries

import cv2


# Load the pre-trained Haarcascades face detection model

face_cascade = cv2.CascadeClassifier('path_to_haarcascade_frontalface_default.xml')


# Initialize face counter

face_count = 0
# Initialize video capture (or read image)

cap = cv2.VideoCapture('path_to_input_video.mp4') # For video stream

# img = cv2.imread('path_to_input_image.jpg') # For static image


while True: # For video stream, loop through frames

    ret, frame = cap.read()


    if not ret:

        break


    # Convert frame to grayscale for face detection
```

```
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

    # Loop through detected faces
    for (x, y, w, h) in faces:
        # Draw bounding box around face
        cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
        face_count += 1
    # Display frame with bounding boxes
    cv2.imshow('Face Counting', frame)

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

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

# Print the final face count
print("Total Faces Detected:", face_count)
```

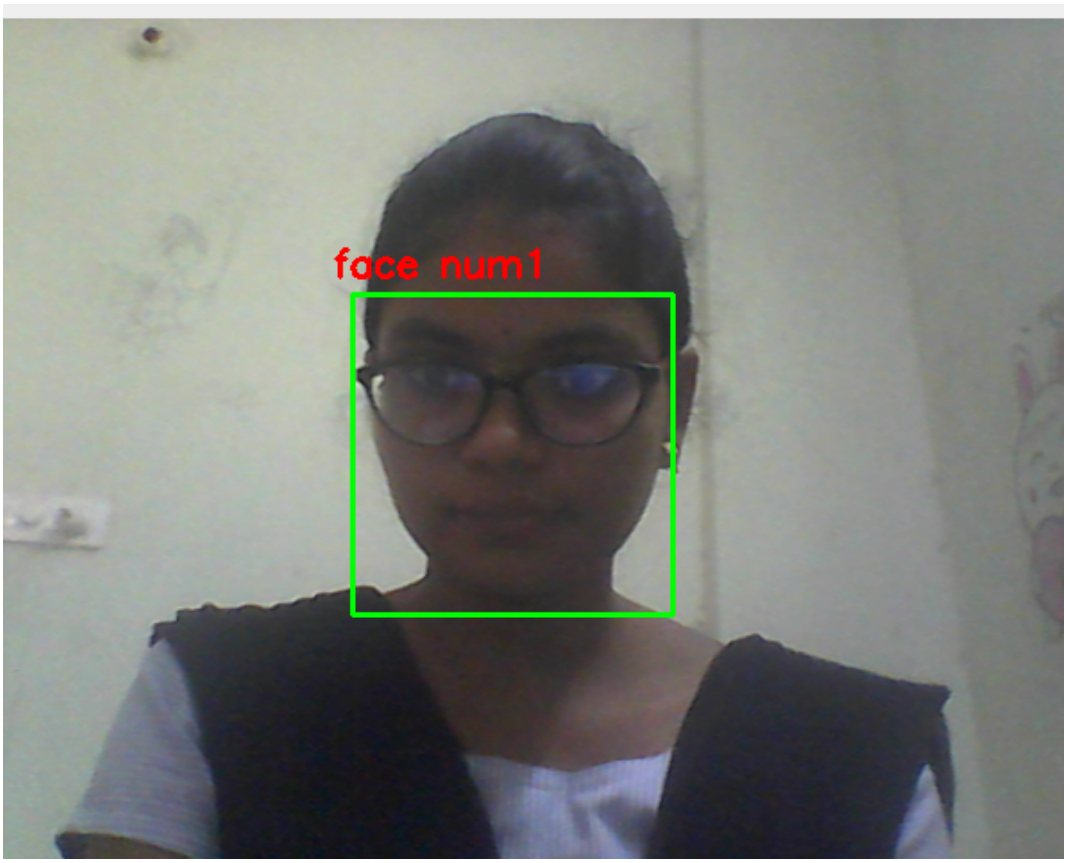
Implementation:

```
# Import required libraries

import cv2

import numpy as np
import dlib

# Connects to your computer's default camera

cap = cv2.VideoCapture(0)

# Detect the coordinates

detector =

dlib.get_frontal_face_detector() # Capture

frames continuously

while True:

# Capture frame-by-frame

ret, frame = cap.read()

frame = cv2.flip(frame, 1)

# RGB to grayscale

gray = cv2.cvtColor(frame,

cv2.COLOR_BGR2GRAY) faces = detector(gray)

# Iterator to count faces

i = 0

for face in faces:
```

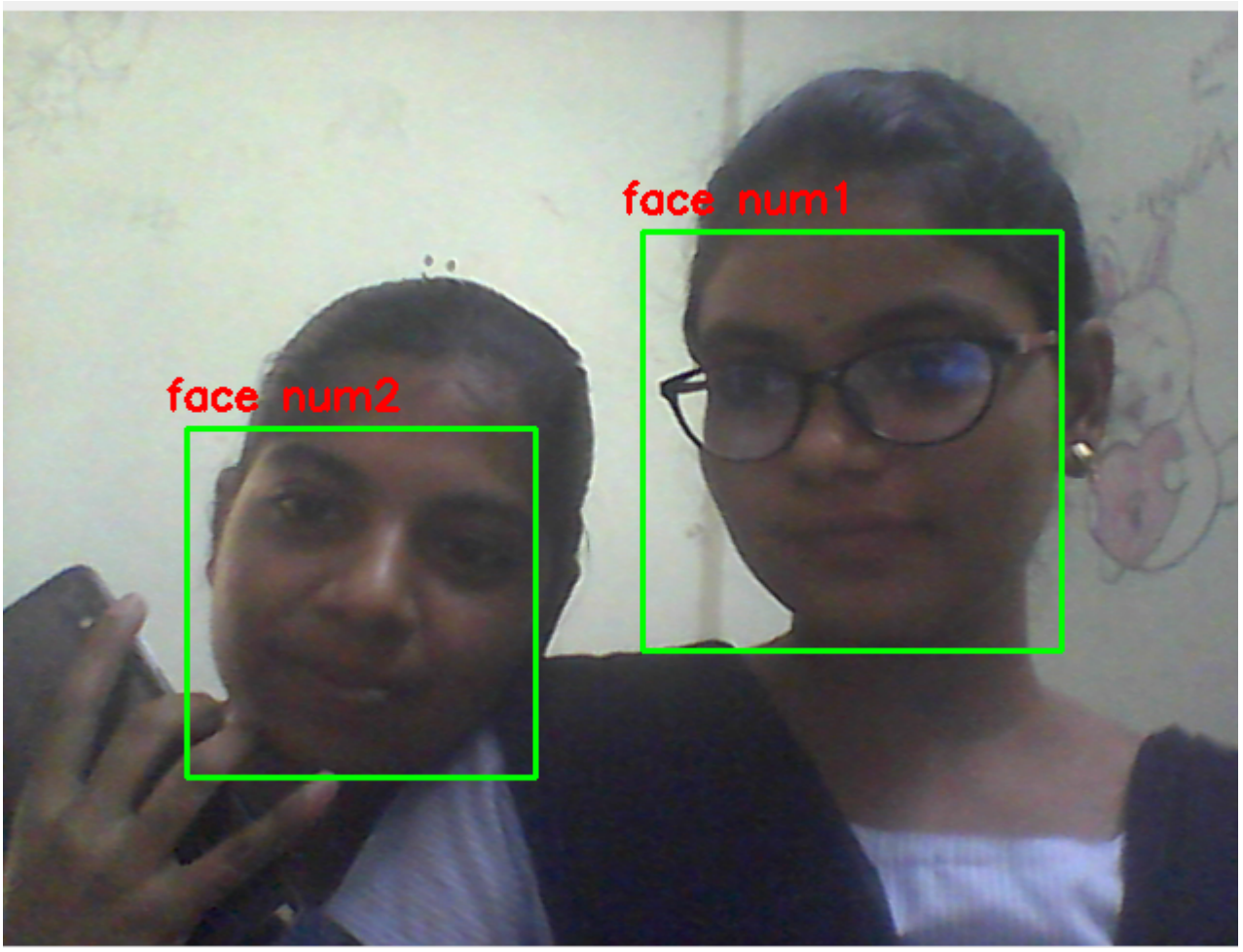
```
# Get the coordinates of faces

x, y = face.left(), face.top()

x1, y1 = face.right(), face.bottom()

cv2.rectangle(frame, (x, y), (x1, y1), (0, 255, 0),

2) # Increment iterator for each face in faces i =

i+1

# Display the box and faces

cv2.putText(frame, 'face num'+str(i), (x-10, y-10),

cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),

2) print(face, i)

# Display the resulting frame
cv2.imshow('frame', frame)

# This command let's us quit with the "q" button on a keyboard.

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

break

# Release the capture and destroy the windows

cap.release()
```


Snap Shots of Count number of Faces:

1.one person


face num1

Two persons:



Conclusion:

In a technologically advancing world, the project "**Counting Faces using Python** and OpenCV" encapsulates the fusion of computer vision expertise and practical application. With a focus on accurate face detection and counting within static images or real-time video streams, this project offers a versatile tool with far-reaching implications.

Throughout the journey, the project delved into the intricate realm of face detection techniques, leveraging the robust capabilities of the OpenCV library. By offering users the option to select between static images and dynamic video streams as input sources, the project ensures adaptability across various scenarios – from analyzing crowds to optimizing security systems.

The core of the project lies in its meticulous face detection mechanism. Employing pre-trained Haarcascades or the potential exploration of deep learning models, the application adeptly identifies faces within each frame of the chosen input source. This foundational step serves as a cornerstone for the accurate face counting process.

The optional integration of face recognition introduces a layer of sophistication, empowering the application to not only count faces but also recognize individuals. This advanced feature expands the project's utility and highlights its potential for diverse applications.
The visual enhancement feature, allowing the superimposition of bounding boxes around detected faces, offers real-time feedback and improves user engagement. The application's dynamic face counting mechanism provides users with instantaneous results, visually displayed alongside the processed frames.

Through customizable output options, users can access face count results through graphical user interfaces or printed notifications, catering to their preferences. The exploration of performance optimization techniques underscores the project's commitment to efficiency and responsiveness.

Beyond its practical utility, the project also serves as an educational gateway. It bridges the gap between theoretical concepts and tangible outcomes, offering insights into computer vision methodologies, face detection techniques, and their real-world implications. This aspect adds an educational dimension that resonates with students and technology enthusiasts alike.

The potential applications are vast and impactful. From enhancing security through efficient crowd monitoring to providing data-driven insights for social

analysis, the project aligns with the evolving landscape of technology-driven solutions.

In conclusion, "Counting Faces using Python and OpenCV" is a testament to the synergy between computer vision theories and tangible applications. By offering accurate face counting, optional recognition, and an educational gateway, this project stands as a dynamic tool that showcases the power of technology to drive real-world impact across multiple domains.

References:

1.https://geeksforgeeks.com / 2.https://chat.openai.com/

3.https://wikipedia.org/

4.https://google.com