**1. Loading and Encoding Faces**

**The FaceRecognition class loads images from a folder (images/) and extracts facial encodings.**

**Each image’s filename (without extension) is stored as the person's name.**

**The encodings are used to match detected faces in real-time.**

Key Code:

img\_encoding = face\_recognition.face\_encodings(rgb\_img)[0]

self.known\_encodings.append(img\_encoding)

self.known\_names.append(filename)

This extracts a unique face encoding from the image and stores it.

**2. Detecting Faces in a Video Stream**

**The program captures frames from the webcam.**

**It detects faces and converts the frame to RGB (as required by face\_recognition).**

**The face locations and encodings are extracted from each frame.**

Key Code:

face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)

face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)

This finds faces and generates encodings for them.

**3. Matching Faces with Stored Encodings**

**The script compares detected face encodings with stored encodings.**

**If a match is found, it retrieves the corresponding name; otherwise, it labels the face as "Unknown."**

Key Code:

matches = face\_recognition.compare\_faces(self.known\_encodings, encoding)

face\_distances = face\_recognition.face\_distance(self.known\_encodings, encoding)

best\_match = np.argmin(face\_distances)

if matches[best\_match]:

name = self.known\_names[best\_match]

The smallest distance match determines the recognized person.

**4. Drawing Bounding Boxes and Names**

**The script draws rectangles around detected faces.**

**The name of the recognized person is displayed above the rectangle.**

Key Code:

cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)

cv2.putText(frame, name, (left, top - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

The color (0, 255, 0) represents green in BGR format.

**5. Running the Webcam and Processing Frames Continuously**

**The script captures frames in a loop.**

**Pressing ESC stops the recognition process.**

Key Code:

while True:

ret, frame = cap.read()

locations, names = fr.recognize\_faces(frame)

if cv2.waitKey(1) & 0xFF == 27:

break

cap.release()

cv2.destroyAllWindows()

This continuously detects faces until the user exits.

**Conclusion**

This project efficiently:

✅ Loads known face encodings

✅ Detects faces in real-time

✅ Matches and labels faces

✅ Displays results with bounding boxes