Title: Face Lock Application

Introduction:

The Face Lock application is a Python-based software designed to ensure secure access control through facial recognition. It utilizes advanced machine learning techniques to detect and recognize faces, providing a seamless and reliable authentication process. The system captures a reference image of the user’s face during setup, creating a unique facial profile.

Once configured, the application uses a webcam to authenticate users in real-time, comparing the live video feed against the stored reference image. This eliminates the need for traditional passwords, enhancing both security and user convenience. The facial recognition technology employed ensures high accuracy and quick response times, making it suitable for various personal and professional use cases.

Built with a focus on usability, the Face Lock application integrates easily into existing systems. It offers robust protection against unauthorized access while maintaining a user-friendly interface. By leveraging the power of Python and cutting-edge machine learning algorithms, this project delivers a modern and secure solution for access control, catering to the growing demand for innovative security technologies.

Abstraction:

Objectives:

- To create a secure, user-friendly application for authentication.
- To utilize facial recognition technology for unlocking systems.
- To demonstrate the implementation of computer vision and machine learning techniques.

Features:

- Face Registration:
    - Captures and stores the user's face as a reference image.
- Authentication:
    - Authenticates the user by comparing live webcam input with the stored reference image.
- Real-Time Feedback:
    - Displays "Face Recognized" and "Unlocked" for authenticated users.
    - Displays "Locked" for non-matching faces.
- Interactive GUI:
    - Provides a simple, webcam-based interface.

Technologies Used

- Programming Language:
    - Python 3.x
- Libraries:
    - **OpenCV:** For image and video processing.
    - **face_recognition:** For face detection and encoding.
    - **NumPy:** For numerical operations.
- Hardware:
    - Webcam (built-in or external).

System Design Workflow

1. **Capture Reference Image:**
    - The user’s face is captured using the webcam and stored as a reference.
2. **Face Encoding:**
    - The reference image is converted into a numerical encoding using the face_recognition library.
3. **Authentication Loop:**
    - The live feed from the webcam is analyzed in real-time.
    - Detected faces are compared with the stored reference encoding.
4. **Match or Mismatch:**
    - If the faces match within a defined tolerance, the system displays "Face Recognized" and "Unlocked."
    - For unmatched faces, the system displays "Locked."

Implementation:

Prerequisites ::

- Install Python 3.x from Python.org.
- Install required libraries:

bash

Copy code

```
pip install opencv-python
pip install face_recognition

pip install numpy
```

Code Structure ::

The application is implemented in a single Python script with the following functions:

- capture_reference_image(): Captures the user's face for reference.
- main(): Handles the authentication process.

Error Handling

The application is designed to handle the following scenarios:

- **Webcam Access Error:**
  - Displays an error if the webcam cannot be accessed.
- **No Face Detected:**
  - Prompts the user to recapture the reference image if no face is detected.
- **Authentication Failures:**
  - Clearly displays "Locked" for non-matching faces.

Challenges:

- **Lighting Conditions:**
  - Variations in lighting can affect face detection accuracy.
- **Camera Quality:**
  - Low-resolution cameras may reduce recognition performance.
- **Multiple Faces:**
  - If multiple faces appear in the frame, only the closest match is used for authentication.

Future Enhancements:

Multi-User Support:

- Add functionality to register and authenticate multiple users.

Mask Detection:

- Enhance the system to recognize users even while wearing masks.

Mobile Integration:

- Extend the application to mobile devices using frameworks like Flutter or Kivy.

Integration with Systems:

- Use the application for system login or door-lock mechanisms.

Applications Where Facial Recognition Technology is Used

1. Security and Access Control :
   Facial recognition is widely used in security systems to enhance access control mechanisms. Organizations use this technology to grant or deny entry into secure areas, such as offices, laboratories, or data centers, ensuring only authorized personnel can access sensitive locations. This replaces traditional keys, cards, or PINs, providing a more secure and efficient system.

2. Smartphone Authentication :
   Most modern smartphones incorporate facial recognition for user authentication. This feature allows users to unlock their devices, authorize payments, and access private data with just a glance. It combines convenience with security, reducing reliance on passwords and fingerprints.

3. Surveillance and Law Enforcement :
   Governments and law enforcement agencies use facial recognition in surveillance systems to identify individuals in public spaces. It assists in locating suspects, monitoring criminal activities, and even tracking missing persons. Airports and border controls also employ this technology for identity verification.

4. Retail and Marketing :
   Retailers use facial recognition to enhance customer experiences. Cameras in stores can recognize returning customers and offer personalized recommendations. This technology also helps in tracking customer behavior, optimizing store layouts, and improving service efficiency.

5. Healthcare and Patient Management :
   In healthcare, facial recognition is used for patient identification, ensuring accurate medical records and preventing fraud. It also aids in monitoring patients' emotional states, especially in mental health treatment, by analyzing facial expressions.

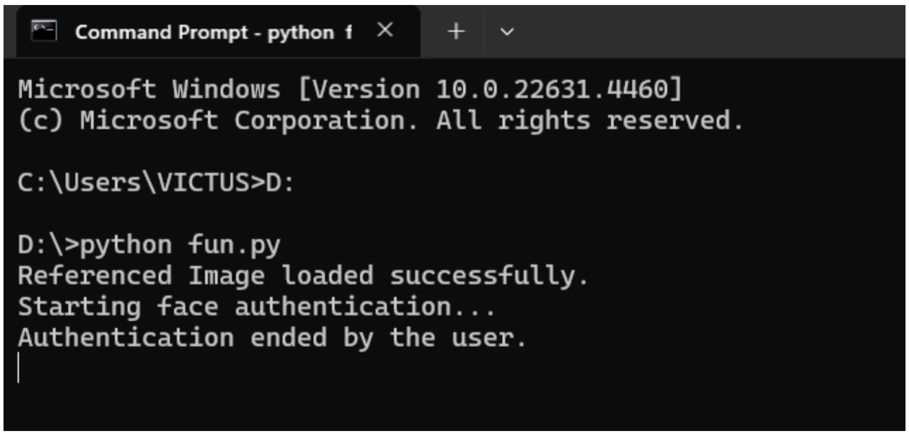
Code

```
import cv2
import face_recognition
import numpy as np
import os

# Paths to store the reference encoding and image
REFERENCE_ENCODING_FILE = "reference_encoding.npy"
REFERENCE_IMAGE_FILE = "D:\papa.jpg"

def capture_reference_image():
    """Capture a reference image for authentication."""
    print("Please look at the camera to capture your face for reference.")
    cap = cv2.VideoCapture(0)

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            print("Error: Unable to access the webcam.")
            break

    cv2.imshow("Reference Capture - Press 's' to Save or 'q' to Quit", frame)

        key = cv2.waitKey(1) & 0xFF
        if key == ord('s'):  # Save reference image
            cap.release()
            cv2.destroyAllWindows()
            return frame
        elif key == ord('q'):  # Quit without saving
            cap.release()
            cv2.destroyAllWindows()
            return None

    cap.release()
    cv2.destroyAllWindows()
    return None


def save_reference_data(encoding, image):
    """Save the reference face encoding and image to files."""
    np.save(REFERENCE_ENCODING_FILE, encoding)  # Save the encoding as a NumPy file
    cv2.imwrite(REFERENCE_IMAGE_FILE, image)  # Save the reference image
    print("Reference face encoding and image saved successfully.")
```

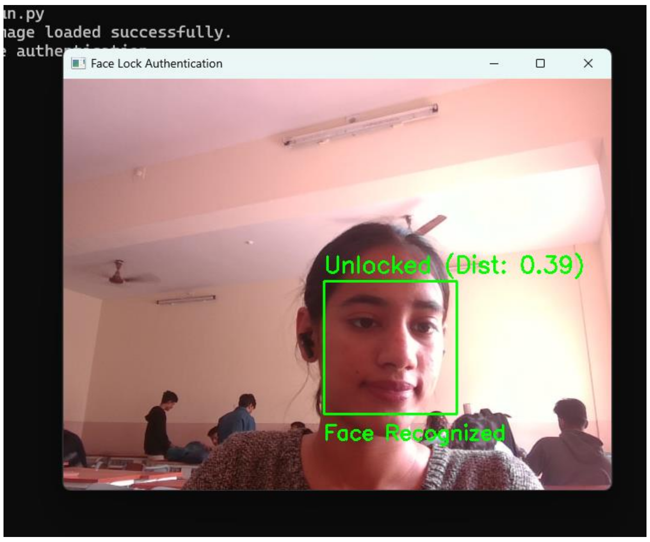
```
def load_reference_data():
    """Load the reference face encoding and image from files."""
    if os.path.exists(REFERENCE_ENCODING_FILE) and
os.path.exists(REFERENCE_IMAGE_FILE):
        encoding = np.load(REFERENCE_ENCODING_FILE)  # Load the
NumPy file
        image = cv2.imread(REFERENCE_IMAGE_FILE)  # Load the
reference image
        return encoding, image
    return None, None


def main():
    try:
        # Step 1: Load or capture reference data
        reference_encoding, reference_image = load_reference_data()

        if reference_encoding is None or reference_image is None:
            print("No reference face found. Capturing a new
reference face.")
            reference_image = capture_reference_image()
            if reference_image is None:
                print("No reference image captured. Exiting.")
                return

            # Encode the reference face
            reference_image_rgb = cv2.cvtColor(reference_image,
cv2.COLOR_BGR2RGB)
            reference_face_encodings =
face_recognition.face_encodings(reference_image_rgb)

            if len(reference_face_encodings) == 0:
                print("Error: No face detected in the reference
image.")
                return

            reference_encoding = reference_face_encodings[0]
            save_reference_data(reference_encoding, reference_image)
        else:
            print("Referenced Image loaded successfully.")

        # Step 2: Start webcam for live authentication
        print("Starting face authentication...")
        cap = cv2.VideoCapture(0)

        while cap.isOpened():
            ret, frame = cap.read()
            if not ret:
                print("Error: Unable to access the webcam.")
                break

            # Convert the frame to RGB and detect faces
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            face_locations =
face_recognition.face_locations(frame_rgb)
 face_encodings = face_recognition.face_encodings(frame_rgb,
face_locations)

  for face_encoding, face_location in zip(face_encodings,
face_locations):
                # Compare the detected face with the reference face
 matches = face_recognition.compare_faces([reference_encoding],
face_encoding, tolerance=0.5)
                face_distance =
face_recognition.face_distance([reference_encoding],
face_encoding)[0]

                top, right, bottom, left = face_location

                if matches[0]:
                    # Add "Face Recognized" message
          cv2.rectangle(frame, (left, top), (right, bottom), (0,
255, 0), 2)
                    cv2.putText(
                        frame, "Face Recognized",
  (left, bottom + 30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 0),
2)
                    cv2.putText(
                        frame, f"Unlocked (Dist:
{face_distance:.2f})",
   (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 255, 0), 2)
                else:
      cv2.rectangle(frame, (left, top), (right, bottom), (0, 0,
255), 2)
                    cv2.putText(
                        frame, "Locked",
         (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (0, 0,
255), 2
                    )

            cv2.imshow("Face Lock Authentication", frame)

            # Press 'q' to quit
            if cv2.waitKey(1) & 0xFF == ord('q'):
                print("Authentication ended by the user.")
                break
```

```python
        cap.release()
        cv2.destroyAllWindows()

    except Exception as e:
        print(f"An error occurred: {e}")


if __name__ == "__main__":
    main()
```

Result



```
Microsoft Windows [Version 10.0.22631.4460]
(c) Microsoft Corporation. All rights reserved.

C:\Users\VICTUS>D:

D:\>python fun.py
Referenced Image loaded successfully.
Starting face authentication...
Authentication ended by the user.
```



```
n.py
age loaded successfully.
 authe
```

References

1. OpenCV Documentation
2. face_recognition Library
3. Python.org - Official Python Documentation
4. Numpy library