Facial Recognition Guide for Raspberry Pi 4 (4GB) + Pi Camera v2

---

1. Hardware Requirements

- **Raspberry Pi 4 (4GB recommended)**
- **Pi Camera v2**
- **MicroSD Card (16GB minimum)**
- **Power Supply**
- **Monitor, Mouse, Keyboard (for setup)**
- **Cooling Solution (recommended for long-term use)**
- **Optional:** GPIO hardware (relay, LED, or servo for access control)

---

2. Install Raspberry Pi OS

1. **Use Raspberry Pi Imager** to flash **Raspberry Pi OS (64-bit)** onto a microSD card.
2. **Boot the Pi**, complete the setup, and **connect to WiFi**.
3. **Enable the Pi Camera**:

   ```
   sudo raspi-config
   ```

   - **Interface Options > Camera > Enable**
   - Exit and reboot:

     ```
     sudo reboot
     ```

---

3. Set Up a Virtual Environment

Using a **virtual environment** isolates dependencies, preventing conflicts with system packages.

Create and Activate the Virtual Environment

```
python3 -m venv --system-site-packages face_rec
source face_rec/bin/activate
```

---

4. Install Required Libraries

Update system and install dependencies:

```
sudo apt update && sudo apt full-upgrade -y
pip install opencv-python imutils face-recognition
sudo apt install cmake -y
```

Download Haarcascade Model for Face Detection

```
wget https://github.com/opencv/opencv/raw/master/data/haarcascades/haarcascade_frontalface_default.xml
```

---

5. Capture and Store Face Images

The following script **captures images for each user and organizes them for training**.

Create a Face Capture Script

```
nano image_capture.py
```

Paste the following:

```
import cv2
import os
from datetime import datetime
from picamera2 import Picamera2
import time

PERSON_NAME = input("Enter name to register: ").strip()

def create_folder(name):
    dataset_folder = "dataset"
    os.makedirs(dataset_folder, exist_ok=True)

    person_folder = os.path.join(dataset_folder, name)
    os.makedirs(person_folder, exist_ok=True)

    return person_folder

def capture_photos(name):
    folder = create_folder(name)

    picam2 = Picamera2()
    picam2.configure(picam2.create_preview_configuration(main={"format": 'XRGB8888', "size": (640, 480)}))
    picam2.start()

    time.sleep(2)

    photo_count = 0
    print(f"Taking photos for {name}. Press SPACE to capture, 'q' to quit.")

    while True:
        frame = picam2.capture_array()
        cv2.imshow('Capture', frame)

        key = cv2.waitKey(1) & 0xFF

        if key == ord(' '):
            photo_count += 1
            filename = f"{name}_{datetime.now().strftime('%Y%m%d_%H%M%S')}.jpg"
            filepath = os.path.join(folder, filename)
            cv2.imwrite(filepath, frame)
            print(f"Photo {photo_count} saved: {filepath}")

        elif key == ord('q'):
            break

    cv2.destroyAllWindows()
    picam2.stop()
    print(f"Photo capture completed. {photo_count} photos saved for {name}.")

if __name__ == "__main__":
    capture_photos(PERSON_NAME)
```

Run the Script

```
python3 image_capture.py
```

- Enter the **user's name**.
- Press **SPACE** to capture images.
- Press **'q'** to exit.
- Repeat for multiple users.

---

6. Train the Facial Recognition Model

This script **trains a model** using stored images.

Create a Training Script

```
nano model_training.py
```

Paste:

```
import os
import face_recognition
import pickle
import cv2
from imutils import paths

print("[INFO] Processing faces...")
imagePaths = list(paths.list_images("dataset"))
knownEncodings = []
knownNames = []

for (i, imagePath) in enumerate(imagePaths):
    print(f"[INFO] Processing image {i + 1}/{len(imagePaths)}")
    name = imagePath.split(os.path.sep)[-2]

    image = cv2.imread(imagePath)
    rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    boxes = face_recognition.face_locations(rgb, model="hog")
    encodings = face_recognition.face_encodings(rgb, boxes)

    for encoding in encodings:
        knownEncodings.append(encoding)
        knownNames.append(name)

print("[INFO] Serializing encodings...")
data = {"encodings": knownEncodings, "names": knownNames}
with open("encodings.pickle", "wb") as f:
    f.write(pickle.dumps(data))

print("[INFO] Training complete. Encodings saved.")
```

Run the Training Script

```
python3 model_training.py
```

- This generates `encodings.pickle`, storing trained faces.

---

7. Test Facial Recognition

This script runs **real-time recognition** using Pi Camera v2.

Create a Recognition Script

```
nano facial_recognition.py
```

Paste:

```
import face_recognition
import cv2
import numpy as np
from picamera2 import Picamera2
import pickle
import time

print("[INFO] Loading encodings...")
```

```
with open("encodings.pickle", "rb") as f:
    data = pickle.loads(f.read())

known_face_encodings = data["encodings"]
known_face_names = data["names"]

picam2 = Picamera2()
picam2.configure(picam2.create_preview_configuration(main={"format": 'XRGB8888', "size": (640, 480)}))
picam2.start()

cv_scaler = 4

while True:
    frame = picam2.capture_array()

    small_frame = cv2.resize(frame, (0, 0), fx=(1/cv_scaler), fy=(1/cv_scaler))
    rgb_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)

    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    face_names = []
    for face_encoding in face_encodings:
        matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
        name = "Unknown"

        face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

        face_names.append(name)

    for (top, right, bottom, left), name in zip(face_locations, face_names):
        top *= cv_scaler
        right *= cv_scaler
        bottom *= cv_scaler
        left *= cv_scaler

        cv2.rectangle(frame, (left, top), (right, bottom), (0, 255, 0), 2)
        cv2.putText(frame, name, (left + 6, top - 6), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2)

    cv2.imshow("Facial Recognition", frame)

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

cv2.destroyAllWindows()
picam2.stop()
```

Run the Recognition Test

```
python3 facial_recognition.py
```

- Recognized faces are labeled.
- Press **'q'** to exit.

Next Steps

- **Integrate MQTT for smart notifications.**
- **Add motion detection using a PIR sensor or Ultrasonic ranger.**
- **Control relays or servos for access control.**