# Face Recognition System with OpenCV

This documentation provides a detailed guide on creating a face recognition system using OpenCV. The approach involves storing data in a folder, processing this data into a pickle model, and then using this model to recognize faces in real-time. The system is designed to run on a Raspberry Pi 4B with 8GB of RAM, targeting a frame rate of 12 FPS.

## Prerequisites

### Hardware

* Raspberry Pi 4B with 8GB of RAM
* Camera (compatible with Raspberry Pi)
* Power supply for Raspberry Pi

### Software

* Raspbian OS (or any compatible OS for Raspberry Pi)
* Python 3.x
* OpenCV
* dlib
* face\_recognition
* numpy
* pickle

## Steps to Create the Face Recognition System

### 1. Setting Up the Environment

First, ensure that your Raspberry Pi is up to date:

sudo apt update  
sudo apt upgrade

Next, install the necessary libraries:

sudo apt install python3 python3-pip  
pip3 install opencv-python opencv-python-headless dlib face\_recognition numpy

### 2. Preparing the Dataset

Store all your images in a folder, for example dataset/. Each image should be named according to the person in the picture, e.g., person1.jpg, person2.jpg, etc.

### 3. Extracting Data and Creating the Pickle Model

Create a script to process the images and generate the pickle model:

import os  
import face\_recognition  
import pickle  
  
# Path to the dataset  
dataset\_path = 'dataset/'  
  
# Initialize lists  
known\_encodings = []  
known\_names = []  
  
# Loop over the images in the dataset  
for filename in os.listdir(dataset\_path):  
 if filename.endswith('.jpg') or filename.endswith('.png'):  
 # Load the image  
 image = face\_recognition.load\_image\_file(os.path.join(dataset\_path, filename))  
  
 # Get the face encoding  
 encodings = face\_recognition.face\_encodings(image)  
  
 if encodings:  
 # Save the encoding and the name  
 known\_encodings.append(encodings[0])  
 known\_names.append(os.path.splitext(filename)[0])  
  
# Save the encodings and names to a pickle file  
with open('encodings.pkl', 'wb') as f:  
 pickle.dump((known\_encodings, known\_names), f)

### 4. Real-time Face Recognition

Create a script to perform real-time face recognition:

import cv2  
import face\_recognition  
import pickle  
import numpy as np  
  
# Load the known faces and names  
with open('encodings.pkl', 'rb') as f:  
 known\_encodings, known\_names = pickle.load(f)  
  
# Initialize the camera  
cap = cv2.VideoCapture(0)  
  
# Target frames per second  
fps = 12  
  
while True:  
 # Capture frame-by-frame  
 ret, frame = cap.read()  
  
 if not ret:  
 break  
  
 # Resize frame for faster processing  
 small\_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)  
 rgb\_small\_frame = small\_frame[:, :, ::-1]  
  
 # Find all face locations and face encodings in the frame  
 face\_locations = face\_recognition.face\_locations(rgb\_small\_frame)  
 face\_encodings = face\_recognition.face\_encodings(rgb\_small\_frame, face\_locations)  
  
 for (top, right, bottom, left), face\_encoding in zip(face\_locations, face\_encodings):  
 matches = face\_recognition.compare\_faces(known\_encodings, face\_encoding)  
 name = "Unknown"  
  
 face\_distances = face\_recognition.face\_distance(known\_encodings, face\_encoding)  
 best\_match\_index = np.argmin(face\_distances)  
 if matches[best\_match\_index]:  
 name = known\_names[best\_match\_index]  
  
 top \*= 4  
 right \*= 4  
 bottom \*= 4  
 left \*= 4  
  
 if name == "Unknown":  
 color = (0, 0, 255)  
 else:  
 color = (0, 255, 0)  
  
 # Draw a rectangle around the face  
 cv2.rectangle(frame, (left, top), (right, bottom), color, 2)  
  
 # Draw a label with a name below the face  
 cv2.rectangle(frame, (left, bottom - 35), (right, bottom), color, cv2.FILLED)  
 font = cv2.FONT\_HERSHEY\_DUPLEX  
 cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)  
  
 # Display the resulting frame  
 cv2.imshow('Face Recognition', frame)  
  
 # Break the loop on 'q' key press  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
  
 # Sleep to target the FPS  
 cv2.waitKey(int(1000 / fps))  
  
# Release the camera and close windows  
cap.release()  
cv2.destroyAllWindows()

### 5. Running the System

To start the face recognition system, simply run the real-time face recognition script:

python3 face\_recognition\_realtime.py

### Conclusion

This guide provides a comprehensive overview of setting up a face recognition system using OpenCV on a Raspberry Pi 4B. The system loads pre-processed face data from a pickle model and performs real-time face recognition, marking recognized faces with green rectangles and unknown faces with red rectangles. The target frame rate is 12 FPS, ensuring smooth performance on the specified hardware.