INSTITUT SUPERIEUR INFORMATIQUE ISI
المعهد العالي للإعلامية

PRESENTÉ À :
MR ALI CHAABANI

SYSTEME DE RECONNAISANCE FACIALE

ELABORÉ PAR :
GUEMBRI LINA
GABSI ZAINEB

1 ING 01
ANNEE UNIVERSITAIRE : 2022-20231

UTM
جامعة تونس المنار
UNIVERSITE DE TUNIS EL MANAR

Table des matières

Table des figures

Chapitre 1

Introduction

L'utilisation de la reconnaissance faciale est de plus en plus courante dans notre vie quotidienne. Cette technologie permet l'identification et la vérification de l'identité d'une personne en comparant les caractéristiques faciales avec une base de données. Cela a des applications pratiques dans des domaines tels que la sécurité, la surveillance et le contrôle d'accès. Avec l'augmentation de la popularité des systèmes de reconnaissance faciale, il est devenu important de développer des solutions efficaces et peu coûteuses pour les rendre accessibles à tous. C'est là qu'entre en jeu le Raspberry Pi, un ordinateur monocarte peu coûteux et facilement programmable. Ce rapport présente la conception, la mise en place et l'évaluation d'un système de reconnaissance faciale basé sur Raspberry Pi. En utilisant cette plateforme, nous avons pu créer un système de reconnaissance faciale performant et peu coûteux qui peut être utilisé dans une variété de situations pratiques.

Chapitre 2

Presentation du projet

2.1 Description du projet

Le projet est un système de reconnaissance faciale qui permet d'identifier les personnes enregistrées dans le système en temps réel à partir d'une vidéo capturée par une caméra PiCamera. L'interface utilisateur est réalisée avec la bibliothèque tkinter. Les encodages faciaux des personnes enregistrées sont stockés dans un tableau numpy, ainsi que leur nom. Le système peut également ajouter de nouvelles personnes à la base de données à partir d'une image capturée par la caméra. L'authentification pour l'ajout de nouvelles personnes nécessite un mot de passe prédéfini. Le système permet également de supprimer des personnes de la base de données. La suppression d'une personne nécessite de saisir son nom et est également protégée par un mot de passe.

2.2 Fonctionnalités

Voici quelques-unes des fonctionnalités clés de notre système :

— Enregistrement des utilisateurs : Le système permet l'enregistrement des utilisateurs en stockant leurs informations personnelles, telles que leur nom et photo.

— Capture d'images : Le système est capable de capturer des images à partir de caméras.
— Détection de visages : Le système utilise des algorithmes de détection de visages pour identifier les visages dans les images capturées.
— Reconnaissance des visages : Le système peut reconnaître les visages enregistrés dans la base de données et identifier les personnes correspondantes.
— Gestion de la base de données : Le système permet la gestion de la base de données des visages enregistrés, y compris l'ajout, la suppression et la modification de données.
— Gestion des autorisations : Le système peut être configuré pour autoriser ou refuser l'accès en fonction des visages détectés.

Chapitre 3

Materiels utilisés

Pour réaliser notre projet, nous aurons besoin de quelques matériaux.

3.1 Carte Raspberrypi 4

La carte Raspberry Pi 4 est un ordinateur monocarte de petite taille conçu pour les applications embarquées et les projets éducatifs. Elle est équipée d'un processeur quad-core Broadcom ARM Cortex-A72 cadencé jusqu'à 1,5 GHz, d'un GPU vidéo Broadcom VideoCore VI, d'un choix de mémoire vive (RAM) allant de 1 à 8 Go et d'un choix de capacité de stockage allant de 16 à 64 Go.

La carte Raspberry Pi 4 dispose également de deux ports micro-HDMI pour la sortie vidéo jusqu'à 4K, d'un port Gigabit Ethernet, de ports USB 2.0 et USB 3.0, d'un port GPIO (General Purpose Input/Output) à 40 broches pour connecter des périphériques externes, ainsi que d'un port d'alimentation USB-C. Elle est compatible avec les systèmes d'exploitation Linux tels que Raspbian, Ubuntu, et de nombreux autres.

Grâce à sa puissance de calcul et à sa connectivité, la carte Raspberry Pi 4 est idéale pour les projets de robotique, d'Internet des objets (IoT), de domotique, de centres

multimédias, de serveurs et bien plus encore. Elle est également abordable et facile à utiliser, ce qui la rend accessible aux amateurs et aux professionnels.

FIGURE 3.1.1 – Carte raspberrypi 4

3.2 Picamera

La PiCamera est une caméra conçue spécifiquement pour être utilisée avec les ordinateurs monocarte Raspberry Pi. Elle peut être connectée directement au port CSI (Camera Serial Interface) de la carte Raspberry Pi pour capturer des images et des vidéos de haute qualité.

La caméra Raspberry Pi est disponible en deux versions : la version standard et la version NoIR (sans filtre infrarouge), qui permet de capturer des images en lumière visible et en infrarouge.

La caméra Raspberry Pi offre une résolution allant jusqu'à 12 mégapixels et une fré-

quence d'images maximale de 90 images par seconde (fps) pour la vidéo à 640x480 pixels. Elle dispose également de fonctionnalités telles que la mise au point automatique, la balance des blancs automatique et la correction des couleurs.

La PiCamera peut être utilisée pour une grande variété de projets, tels que la surveillance vidéo, la photographie de la nature, la robotique et bien plus encore. Elle est également très populaire dans les projets de bricolage et les projets éducatifs en raison de sa facilité d'utilisation et de sa compatibilité avec la carte Raspberry Pi.

FIGURE 3.2.1 – Picamera

3.3 Carte SD 16G

La carte SD de 16 Go est une petite carte mémoire amovible de stockage de données utilisée dans divers appareils électroniques tels que les smartphones, les tablettes, les appareils photo, les caméras, les lecteurs MP3 et les consoles de jeux portables. Elle peut stocker jusqu'à 16 gigaoctets de données, y compris des photos, des vidéos, de la musique et d'autres fichiers. Les cartes SD sont pratiques car elles peuvent être facilement échangées entre différents appareils et permettent aux utilisateurs d'augmenter la capacité de stockage de leurs appareils portables. Dans notre projet elle est utilisée pour contenir l'OS Raspien ,le code et les données .

FIGURE 3.3.1 – Catre SD

3.4 Cable Ethernet

Le câble Ethernet est un câble de communication filaire utilisé pour connecter des appareils réseau tels que des ordinateurs, des routeurs, des modems, des switchs, etc. Il permet de transférer des données à haut débit entre ces appareils en utilisant une connexion filaire stable et fiable.

En ce qui concerne la carte Raspberry Pi, le câble Ethernet peut être utilisé pour connecter la carte à un réseau local ou à Internet en utilisant une connexion filaire. Cela permet à la carte Raspberry Pi de communiquer avec d'autres appareils connectés au même réseau et d'accéder à des ressources en ligne telles que des bases de données, des sites Web, etc. De plus, l'utilisation d'une connexion Ethernet peut être plus stable et plus rapide que l'utilisation de connexions sans fil telles que le Wi-Fi.

FIGURE 3.4.1 – Cable Ethernet

Chapitre 4

Technologies et Langages

Le système de reconnaissance faciale est une technologie de plus en plus utilisée dans diverses applications telles que la sécurité, la surveillance, la vérification d'identité, la recherche criminelle, etc. Cette technologie utilise des algorithmes de traitement d'image pour identifier et comparer les caractéristiques faciales d'un individu avec une base de données de visages préenregistrés.

En termes de technologies, le système de reconnaissance faciale peut être mis en place en utilisant une variété de technologies telles que les caméras, les capteurs, les algorithmes de traitement d'image, les bases de données, etc.

4.1 Raspbian

Dans ce rapport, nous avons utilisé l'OS Raspbian pour alimenter notre système de reconnaissance faciale. En effet , Raspbian est un système d'exploitation open source basé sur Debian , spécialement conçu pour les Raspberry Pi. Il est développé par la Fondation Raspberry Pi et est optimisé pour fonctionner sur des ordinateurs monocartes à faible consommation d'énergie. Raspbian est gratuit et open source, et il

est livré avec une grande variété d'outils et de logiciels pré-installés pour les projets éducatifs, les projets d'Internet des objets, les serveurs web et les médias.

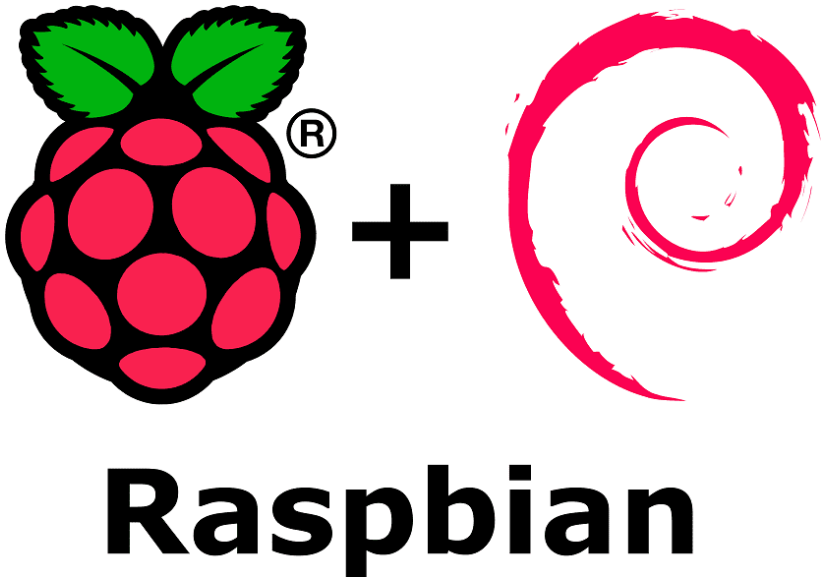

FIGURE 4.1.1 – Logo du systeme d'exploitation Raspbian

En outre, nous avons utilisé le langage de programmation Python, qui est un langage populaire pour la programmation de l'apprentissage automatique et de la vision par ordinateur. Nous avons utilisé les bibliothèques OpenCV et FaceRecognition de Python pour la reconnaissance faciale.

FIGURE 4.1.2 – Logo du langage Python

4.1.1 Opencv

OpenCV est une bibliothèque de vision par ordinateur open-source largement utilisée pour la manipulation d'images et la reconnaissance de formes . Dans notre cas , la bibliothèque OpenCV (importée avec "import cv2") pour traiter le flux vidéo provenant de la caméra.

FIGURE 4.1.3 – Logo de la bibliothéque Opencv

4.1.2 Face Recognition

La bibliothèque face_recognition est une bibliothèque qqui permet la reconnaissance faciale en utilisant des algorithmes d'apprentissage automatique Dans notre projet , elle est utilisée pour effectuer la reconnaissance faciale en temps réel à partir d'une caméra. Elle est utilisée pour détecter les visages dans une image en utilisant la méthode face_locations(), et pour encoder ces visages en utilisant la méthode face_encodings(). Les visages sont ensuite comparés avec les visages connus à l'aide de la méthode compare_faces(). Si un match est trouvé, le nom de la personne correspondante est affiché sur l'image en utilisant la méthode putText() de la bibliothèque OpenCV. La méthode load_image_file() de la bibliothèque est également utilisée pour charger l'image de la personne à ajouter dans le système.

4.2 ISIS Proteus

ISIS Proteus est un logiciel de simulation électronique développé par la société Labcenter Electronics. Il permet de concevoir, simuler et tester des circuits électroniques analogiques, numériques et mixtes. Le logiciel offre une interface conviviale qui permet de créer des schémas électroniques et de simuler leur comportement en temps réel. ISIS Proteus comprend également des outils de conception de PCB (Printed Circuit Board) qui permettent de créer des modèles de circuits imprimés en 2D et 3D. Le logiciel est largement utilisé par les ingénieurs électroniques et les étudiants en électronique pour la conception et la simulation de circuits électroniques. C'est pour cela qu'on a exploiter Proteus pour réaliser une simulation de notre systeme avant de commencer le travail avec les matériaux .

FIGURE 4.2.1 – Logo de ISIS Proteus

Chapitre 5

Simulation et Cablage

5.1 Simulation

La simulation avec ISIS Proteus est un outil très utile pour les ingénieurs électroniques avant de commencer la réalisation d'un système électronique. La simulation permet de tester et de valider le fonctionnement théorique du système avant sa mise en œuvre physique. En utilisant ISIS Proteus, l'ingénieur peut concevoir et simuler des circuits électroniques pour vérifier leur fonctionnement, identifier les problèmes potentiels et les corriger avant de passer à la phase de réalisation. La simulation peut également permettre d'optimiser la conception du circuit pour améliorer ses performances et réduire les coûts. Ainsi, l'utilisation de la simulation avec ISIS Proteus peut réduire considérablement les coûts et les délais de développement, tout en améliorant la qualité et la fiabilité du produit final.

Pour notre projet , La simulation nous a permit de tester le code avec un bouton pour déclencher le systeme , une Picamera pour capturer les images , une carte Raspberrypi et un ecran LCD pour l'affichage .

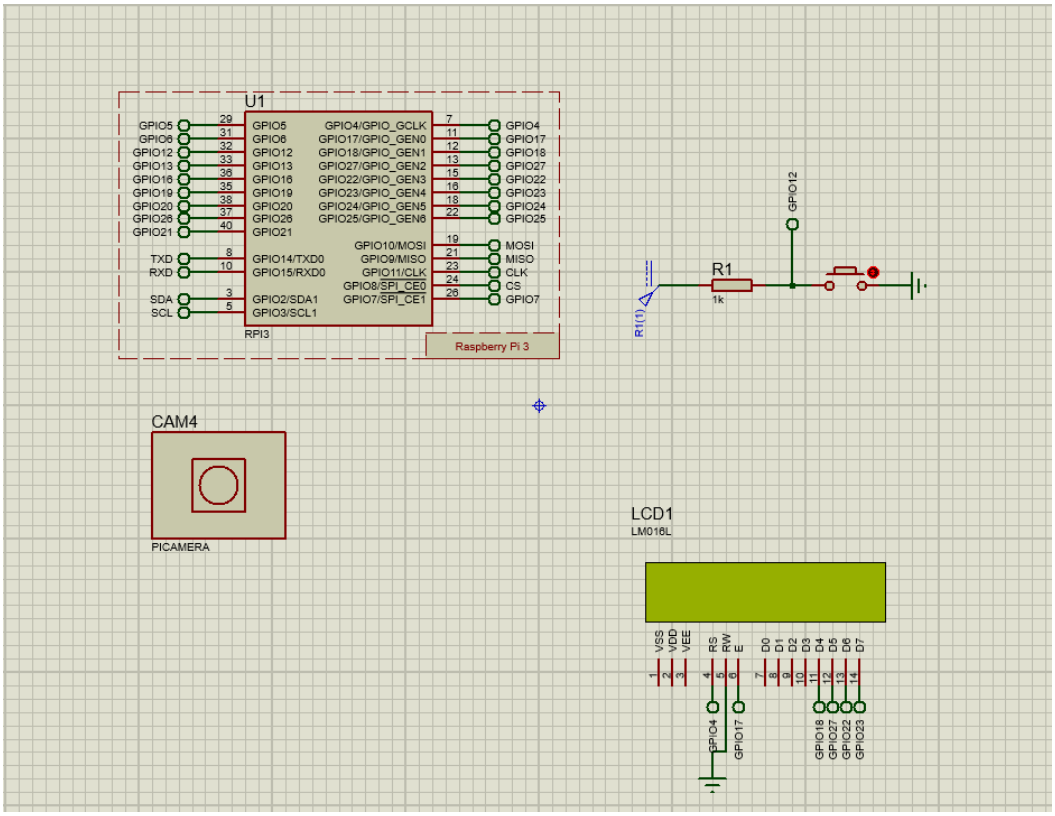

FIGURE 5.1.1 – Simulation du Systeme avec ISIS proteus

5.2 Cablage

5.2.1 Cablage entre la camera et la carte Raspberrypi

Comme premiere etape pour le cablage , on devait relier la camera à la catre Raspberry . Pour assurer cette tache on a suivit les etapes suivantes :

— Éteidre la Raspberry Pi et débrancher la de toute source d'alimentation.

— Insértion de la caméra Pi dans le connecteur CSI de la Raspberry Pi.

— S'assurezque la caméra Pi est solidement fixée en place.

— Connection du câble de la caméra Pi à la Raspberry Pi en poussant le connecteur bleu dans le connecteur CSI de la Raspberry Pi.

— Démarrage de la Raspberry Pi et vérification que la caméra Pi est détectée en

exécutant une commande dans le terminal. (sudo raspistill -o test.jpg)

FIGURE 5.2.1 – Cablage d'une Picamera avec Raspberrypi 4

5.2.2 Cablage entre la carte Raspberry et l'ordinateur

Pour accéder à une carte Raspberry Pi , il existe plusieurs vois : headless Setup sans cable ou à travers un cable Ethernet . La deuxieme vois nous permet d'accéder à la carte avec n'importe quelle connexion WIFI c'est pour cela qu'on a opter pour cette option . pour ce faire on a suivit ces etapes :

— Connection du câble Ethernet à la Raspberry Pi et à l'ordinateur.
— Démarrage de la Raspberry Pi.
— ouverture du terminal : "ifconfig" pour afficher la configuration réseau de l'ordinateur.
— Recherche de la section Ethernet et ecriture de l'adresse IP de l'ordinateur.
— Sur la Raspberry Pi, ouvrture d'un terminal

— commande : "sudo raspi-config" pour ouvrir l'utilitaire de configuration.

— Sélection "Interface Options" puis "SSH" et activation de SSH. le terminal de l'ordinateur : "ssh pi@adresse_ip_de_la_Raspberry_Pi" en remplaçant "adresse_ip_de_la_Raspber par l'adresse IP de la carte Raspberry Pi.

— Accée par mot de passe de la Raspberry Pi si nécessaire.

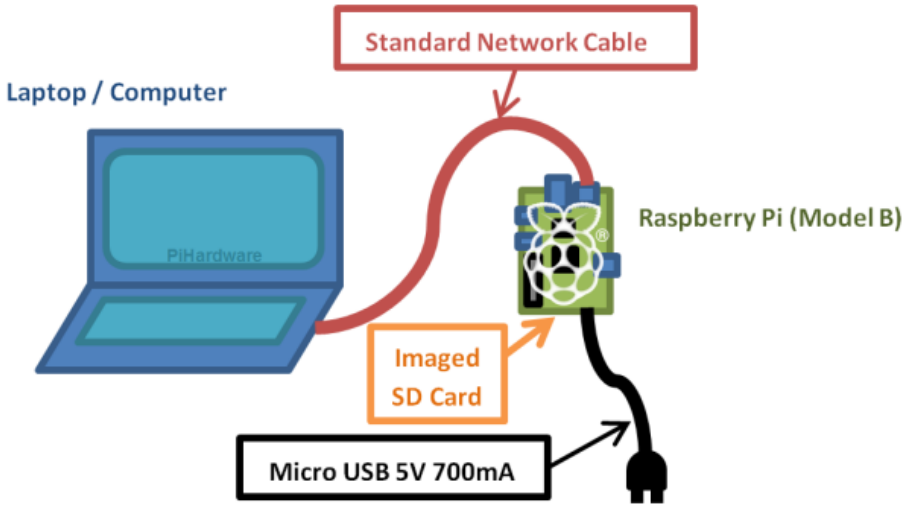

FIGURE 5.2.2 – Cablage de la carte Raspberrypi avec l'ordinteur

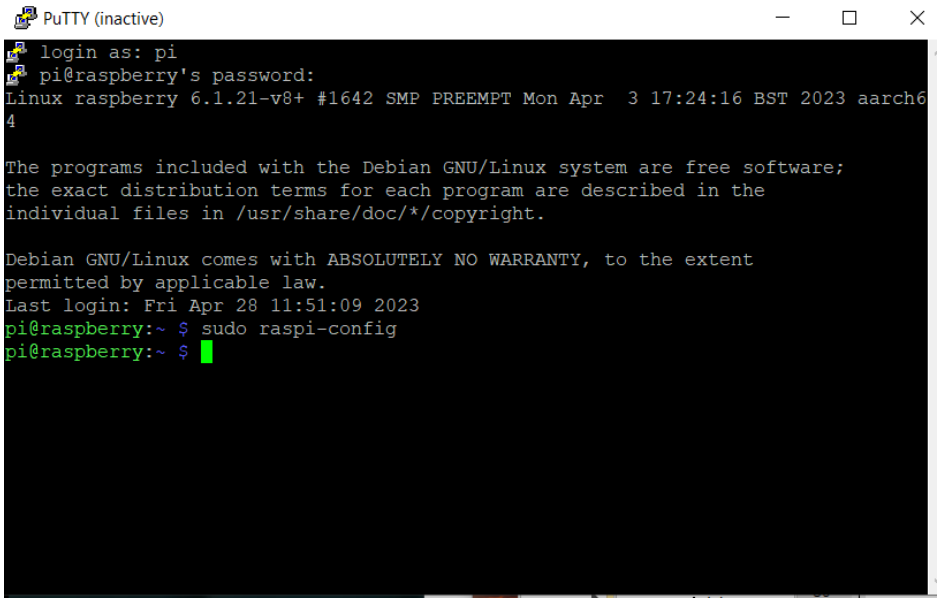

FIGURE 5.2.3 – Connexion à la carte Raspberrypi

Chapitre 6

Réalisation

Cette partie va être consacrée à la mise en œuvre concrète des éléments planifiés de notre projet et à la réalisation du code.

6.1 Assemblage du matériels

L'assemblage du matériel était assez simple. Nous avons commencé par insérer la carte SD contenant le système d'exploitation Raspbian dans la carte Raspberry Pi 4. Ensuite, nous avons connecté la caméra Pi à la fente prévue à cet effet sur la carte Raspberry Pi. Nous avons également connecté un câble Ethernet de la carte Raspberry Pi à notre ordinateur pour pouvoir accéder à la carte via une connexion réseau. Pour accéder à l'interface utilisateur de Raspbian, nous avons utilisé le logiciel Putty pour établir une connexion SSH et le logiciel VNC viewer pour afficher le contenu de la carte Raspberry Pi sur notre ordinateur.

Une fois l'assemblage de base terminé, nous avons commencé à configurer le système d'exploitation. Nous avons d'abord effectué une mise à jour du système d'exploitation en utilisant la commande sudo apt-get update. Ensuite, nous avons installé les biblio-

thèques OpenCV et Face Recognition nécessaires à l'exécution de notre programme de reconnaissance faciale. Nous avons également configuré la caméra Pi en utilisant la commande sudo raspi-config pour activer la caméra et définir la résolution appropriée.

6.2 Code du projet

Nous avons écrit le code du projet en utilisant Python 3. Nous avons utilisé la bibliothèque OpenCV pour la reconnaissance faciale. Nous avons également utilisé la bibliothèque Face Recognition pour la reconnaissance des visages. Le code est divisé en deux parties principales : la première partie gère la détection de mouvement et la seconde partie gère la reconnaissance faciale. La détection de mouvement est utilisée pour détecter la présence d'une personne devant la caméra et activer la reconnaissance faciale. Si une personne est détectée, le système compare la capture d'image avec les visages stockés dans la base de données. Si une correspondance est trouvée, le système affiche le nom de la personne sur l'écran.

Enfin, nous avons testé le système en utilisant des images de visages connus et inconnus pour vérifier la précision de la reconnaissance faciale. Nous avons également effectué des tests pour nous assurer que le système était réactif et fonctionnait en temps réel.

```
from PIL import Image, ImageTk
import time
import picamera
from tkinter import simpledialog, messagebox
import os
VIDEO_WIDTH = 640
VIDEO_HEIGHT = 480
VIDEO_FPS = 30
PASSWORD = "8050"
class FaceRecognitionGUI:
    def __init__(self, root):
        self.root = root
        self.root.title("Face Recognition System")
        self.video_label = tk.Label(self.root)
        self.video_label.pack()
        self.start_stop_button = tk.Button(self.root, text="Start", command=self.toggle_recognition,
                                           bg="green", fg="white", font=("Arial", 16))
        self.start_stop_button.pack(pady=10)
        self.add_person_button = tk.Button(self.root, text="Add Person", command=self.authenticate,
                                           bg="blue", fg="white", font=("Arial", 16))
        self.add_person_button.pack(pady=10)
        self.camera = picamera.PiCamera()
        self.camera.resolution = (VIDEO_WIDTH, VIDEO_HEIGHT)
        self.camera.framerate = VIDEO_FPS
        self.frame = None
        self.recognize_faces = False
        self.thread = threading.Thread(target=self.capture_frames, args=())
        self.thread.daemon = True
        self.known_encodings = np.array([])
        self.known_names = []
        self.delete_person_button = tk.Button(self.root, text="Delete Person", command=lambda:
                                              self.delete_person(simpledialog.askstring("Name", "Enter the name of the p
                                              bg="red", fg="white", font=("Arial", 16))
        self.delete_person_button.pack(pady=10)
        self.load_known_encodings_and_names()
```

FIGURE 6.2.1 – Initialisation des variables

La partie de code ci-dessous contient trois fonctions : La fonction `start()` démarre le thread et la boucle principale de la fenêtre graphique.

La fonction `capture-frames()` capture les images de la caméra et utilise la bibliothèque `face-recognition` pour détecter les visages dans chaque image. Elle compare ensuite les visages détectés avec les visages connus et affiche le nom de chaque personne connue détectée. Elle met également à jour l'affichage de la fenêtre graphique avec la nouvelle image traitée.

La fonction `toggle-recognition()` permet de démarrer ou d'arrêter la reconnaissance faciale. Lorsque la reconnaissance faciale est active, elle met à jour l'affichage des images en temps réel et affiche les noms des personnes connues détectées. Lorsqu'elle est désactivée, elle arrête de traiter les images et met à jour le bouton de démarrage/arrêt pour indiquer l'état actuel.

```
    def start(self):
        self.thread.start()
        self.root.mainloop()

    def capture_frames(self):
        while True:
            if self.recognize_faces:
                frame = np.empty((VIDEO_HEIGHT * VIDEO_WIDTH * 3,), dtype=np.uint8)
                self.camera.capture(frame, 'bgr', use_video_port=True)
                frame = frame.reshape((VIDEO_HEIGHT, VIDEO_WIDTH, 3))
                face_locations = face_recognition.face_locations(frame)
                face_encodings = face_recognition.face_encodings(frame, face_locations)
                for face_location, face_encoding in zip(face_locations, face_encodings):
                    matches = face_recognition.compare_faces(self.known_encodings, face_encoding)
                    name = "Unknown Person"
                    if True in matches:
                        match_index = matches.index(True)
                        name = self.known_names[match_index]
                    top, right, bottom, left = face_location
                    color = (0, 255, 0) if name != "Unknown Person" else (0, 0, 255)
                    cv2.rectangle(frame, (left, top), (right, bottom), color, 2)
                    cv2.putText(frame, name, (left, top - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2
                cv2image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
                pil_image = Image.fromarray(cv2image)
                tk_image = ImageTk.PhotoImage(image=pil_image)
                self.video_label.configure(image=tk_image)
                self.video_label.image = tk_image
            else:
                time.sleep(0.1)

    def toggle_recognition(self):
        if self.recognize_faces:
            self.recognize_faces = False
            self.start_stop_button.configure(text="Start", bg="green")
        else:
            self.recognize_faces = True
            self.start_stop_button.configure(text="Stop", bg="red")
```

FIGURE 6.2.2 – Fonctions de reconnaissance faciale en temps réel

Cette partie de code contient trois fonctions liées à la gestion des personnes connues dans le système de reconnaissance faciale.

La fonction `authenticate()` demande à l'utilisateur de saisir un mot de passe et si le mot de passe est correct, elle appelle la fonction add-person pour ajouter une nouvelle personne au système.

La fonction `delete-person()` permet de supprimer une personne connue du système en supprimant son encodage de visage et son nom de la liste des personnes connues.

La fonction `add-person()` permet à l'utilisateur d'ajouter une nouvelle personne au système. Elle demande à l'utilisateur de saisir le nom de la personne, puis elle affiche un message demandant à l'utilisateur de regarder directement la caméra et de prendre une photo en appuyant sur OK. Si un seul visage est détecté dans la photo, l'encodage du visage est ajouté à la liste des encodages connus et le nom de la personne est ajouté à la liste des noms connus. Sinon, un message d'erreur est affiché demandant à l'utilisateur de réessayer.

```
    def authenticate(self):
        password = simpledialog.askstring("Password", "Enter the password:", parent=self.root, show="*")
        if password == PASSWORD:
            self.add_person()
    def delete_person(self, name):
        if name in self.known_names:
            index = np.where(self.known_names == name)[0][0]
            self.known_encodings = np.delete(self.known_encodings, index, axis=0)
            self.known_names = np.delete(self.known_names, index)
            self.save_known_encodings_and_names()
            messagebox.showinfo("Success", f"{name} has been removed from the system.")
        else:
            messagebox.showerror("Error", f"{name} not found in the system.")
    def add_person(self):
        name = simpledialog.askstring("Name", "What's the person's name?")
        if name is not None:
            message = "Look directly at the camera and press OK to take a picture."
            messagebox.showinfo("Take Picture", message)
            self.camera.start_preview()
            time.sleep(2)
            self.camera.capture("temp.jpg")
            self.camera.stop_preview()
            image = face_recognition.load_image_file("temp.jpg")
            face_locations = face_recognition.face_locations(image)
            if len(face_locations) == 0:
                messagebox.showerror("Error", "No face detected. Please try again.")
            elif len(face_locations) > 1:
                messagebox.showerror("Error", "Multiple faces detected. Please try again.")
            else:
                face_encoding = face_recognition.face_encodings(image, face_locations)[0]
                self.known_encodings = np.append(self.known_encodings, [face_encoding], axis=0)
                self.known_names = np.append(self.known_names, name)

                self.save_known_encodings_and_names()
                messagebox.showinfo("Success", f"{name} has been added to the system.")
```

FIGURE 6.2.3 – Authentification et gestion de la base de données des visages

Cette partie du code contient deux fonctions, qui permettent de charger et de sauvegarder les encodages des visages et les noms des personnes connues dans le système de reconnaissance faciale. Les encodages et les noms sont stockés dans des fichiers .npy.

La fonction `load-known-encodings-and-names()` charge les encodages et les noms à partir des fichiers .npy s'ils existent. Si les fichiers n'existent pas, la fonction ne fait rien.

La fonction `save-known-encodings-and-names()` sauvegarde les encodages et les noms dans les fichiers .npy.

Enfin, le bloc `if name == "main":` crée une instance de la classe FaceRecognitionGUI et appelle sa méthode start() pour démarrer l'application.

```
    def load_known_encodings_and_names(self):
        try:
            with open("encodings.npy", "rb") as f:
                self.known_encodings = np.load(f, allow_pickle=True)
            with open("names.npy", "rb") as f:
                self.known_names = np.load(f, allow_pickle=True)
        except FileNotFoundError:
            pass

    def save_known_encodings_and_names(self):
        with open("encodings.npy", "wb") as f:
            np.save(f, np.array(self.known_encodings))
        with open("names.npy", "wb") as f:
            np.save(f, np.array(self.known_names))


if __name__ == "__main__":
    face_recognition_gui = FaceRecognitionGUI(tk.Tk())
    face_recognition_gui.start()
```

FIGURE 6.2.4 – Sauvegarde et chargement des encodages et des noms connus

Chapitre 7

Conclusions

Pour conclure, ce travail est fait dans le contexte d'un mini projet de la matière « Système de microprocesseur ». Le système de reconnaissance faciale avec Raspberry Pi est un projet ambitieux qui a été réalisé en utilisant les bibliothèques OpenCV et face-recognition, nous avons pu développer un système capable de détecter, reconnaître et suivre les visages en temps réel. En plus une interface graphique utilisateur (GUI) a été crée pour faciliter son utilisation.

Ce projet nous a permis de comprendre en profondeur le fonctionnement des algorithmes de reconnaissance faciale et des techniques de traitement d'image, ainsi que les différentes technologies utilisées pour implémenter ces algorithmes. Nous avons également appris à travailler avec le Raspberry Pi, à interagir avec des périphériques tels que la caméra et à utiliser des bibliothèques tierces pour des applications de reconnaissance faciale.

En fin de compte, ce projet peut être utilisé dans de nombreux domaines, tels que la sécurité, la surveillance, la gestion de la présence ou même pour des applications de marketing personnalisées. Nous sommes convaincus que ce projet peut servir de base solide pour des projets futurs dans le domaine de la reconnaissance faciale et de

l'intelligence artificielle en général.