Université Abdelmalek Essaadi
Faculté des Sciences et Techniques - Tanger
Département Génie Informatique

Filière :

Logiciels et Systèmes Intelligents

Rapport de Projet :

Identification des personnes par reconnaissance faciale

Réalisé par :
BAHASSOU Mohamed Amine
ECHFFANI Hodaifa
BOUAAMOIUD Abdellah

Encadré par :
Prof. AIT KBIR M'hamed

Année universitaire : 2025–2026

Table des matières

1 Introduction Générale

La vision par ordinateur est un domaine de l'intelligence artificielle qui vise à permettre aux machines de "voir" et d'interpréter le monde visuel. Parmi ses applications les plus connues, la reconnaissance faciale se distingue par son potentiel et sa complexité. Elle consiste à identifier ou à vérifier l'identité d'une personne à partir d'une image ou d'une vidéo.

Ce projet a pour objectif de développer une application robuste et évolutive de reconnaissance faciale en temps réel. En utilisant Python et des bibliothèques spécialisées comme **OpenCV** pour le traitement vidéo et **face_recognition** pour les algorithmes d'identification, nous avons mis en place un système capable de reconnaître des individus à partir d'un flux de webcam. L'architecture a été conçue pour être facilement maintenable, permettant l'ajout de nouvelles personnes sans modifier le code, simplement en enrichissant une base de données d'images.

2 Architecture et Fonctionnalités

2.1 Fonctionnalités Principales

Le système développé intègre plusieurs fonctionnalités clés qui en font une solution complète et performante :

— **Base de Données Dynamique :** Le système charge automatiquement toutes les identités à partir d'une structure de dossiers simple. Chaque dossier représente une personne et contient ses photos.

— **Pipeline de Reconnaissance Complet :** Le processus suit les trois étapes fondamentales de la reconnaissance faciale :

1. **Détection** des visages dans l'image.
2. **Extraction de Caractéristiques** pour créer une signature numérique unique du visage (un "encoding").
3. **Comparaison et Identification** en calculant la similarité entre les visages détectés et ceux de la base de données.

— **Performances Optimisées :** Pour assurer une analyse fluide en temps réel, le traitement est accéléré en ne traitant qu'une trame sur deux et en travaillant sur des images redimensionnées.

— **Visualisation Claire :** Les résultats sont affichés en direct sur le flux vidéo. Un cadre est dessiné autour de chaque visage, avec le nom de la personne identifiée ou la mention "Inconnu".

2.2 Structure du Projet

Pour fonctionner, l'application requiert une organisation spécifique des fichiers. Un dossier racine nommé `database` contient des sous-dossiers portant le nom de chaque personne à identifier.

```
votre_projet/
|-- reconnaissance_faciale.py
|-- database/
|   |-- Barack_Obama/
|   |   |-- obama1.jpg
|   |   `-- obama2.png
|   |-- Joe_Biden/
|   |   `-- biden1.jpeg
|   `-- Autre_Personne/
|       `-- photo.jpg
```

3 Le Pipeline de Reconnaissance Faciale

L'application repose sur un pipeline en deux phases : une phase d'apprentissage (chargement de la base de données) et une phase d'identification en temps réel.

3.1 Phase 1 : Apprentissage et Encodage

Au démarrage, le script parcourt le dossier `database`. Pour chaque image trouvée, il réalise les opérations suivantes :

1. **Chargement de l'image** en mémoire.
2. **Détection du visage** présent dans l'image.
3. **Calcul de l'encodage facial** (*face encoding*). Il s'agit d'une transformation du visage en un vecteur de 128 valeurs numériques. Ce vecteur est une signature unique qui représente les caractéristiques biométriques du visage. Le modèle sous-jacent est un réseau de neurones profond (Deep Learning) pré-entraîné sur des millions de visages.

Tous les encodages et les noms correspondants sont stockés dans deux listes en mémoire, prêts à être utilisés pour la comparaison.

3.2 Phase 2 : Identification en Temps Réel

Une fois la base de données chargée, la boucle principale s'active et traite le flux de la webcam.

Étape A : Détection des Visages Pour chaque trame, la fonction `face_recognition.face_locations()` analyse l'image et retourne les coordonnées de tous les visages détectés.

Étape B : Extraction des Caractéristiques Un encodage facial (vecteur de 128 dimensions) est calculé pour chaque visage détecté dans la trame.

Étape C : Comparaison et Identification C'est l'étape cruciale. L'encodage du visage inconnu est comparé à tous les encodages stockés en mémoire.

- La comparaison est effectuée en calculant la **distance euclidienne** entre les vecteurs. Une distance faible signifie une grande similarité.
- Le nom associé à l'encodage connu ayant la **plus petite distance** est retenu comme le meilleur candidat.
- Un **seuil de tolérance** (ici, 0.6) est appliqué. Si la plus petite distance est supérieure à ce seuil, le système considère qu'il n'y a pas de correspondance satisfaisante, et le visage est étiqueté "Inconnu".

4 Implémentation et Analyse du Code

Le script est structuré en plusieurs parties logiques. Nous allons les analyser en détail.

4.1 Importations et Dépendances

Le script commence par importer les bibliothèques nécessaires.

- `face_recognition` : La bibliothèque principale pour la détection et l'encodage.
- `cv2` (OpenCV) : Utilisée pour la capture et la manipulation des flux vidéo.
- `numpy` : Pour les opérations mathématiques, notamment le calcul du minimum des distances.
- `os` et `glob` : Pour la navigation dans les dossiers et la recherche de fichiers images.

4.2 Chargement de la Base de Données

La fonction `load_known_faces` est responsable de la phase d'apprentissage. Elle parcourt la structure de dossiers, charge chaque image, en extrait l'encodage facial et stocke le tout dans deux listes.

```
def load_known_faces(database_path='database'):
    known_face_encodings = []
    known_face_names = []

    print("Chargement de la base de donnees de visages...")

    # Parcourir chaque sous-dossier (chaque personne)
    for person_name in os.listdir(database_path):
        person_dir = os.path.join(database_path, person_name)
        if not os.path.isdir(person_dir):
            continue

        # Trouver toutes les images dans le dossier de la personne
        image_paths = glob.glob(os.path.join(person_dir, '*.[jJpP][pPnN][gG]*')
    )

        # Encoder chaque image de la personne
        for image_path in image_paths:
            try:
                image = face_recognition.load_image_file(image_path)
                face_encoding_list = face_recognition.face_encodings(image)

                if face_encoding_list:
                    face_encoding = face_encoding_list[0]
                    known_face_encodings.append(face_encoding)
                    known_face_names.append(person_name)
            except Exception as e:
                print(f"ERREUR lors du chargement de {image_path}: {e}")

    print("Chargement termine.")
    return known_face_encodings, known_face_names
```

Listing 1 – Fonction de chargement et d'encodage des visages

4.3 Boucle Principale de Traitement

Cette boucle infinie est le cœur du traitement en temps réel.

```
# Obtenir une reference a la webcam
video_capture = cv2.VideoCapture(0)
process_this_frame = True

while True:
    ret, frame = video_capture.read() # Capturer une trame
    if not ret: break

    # Ne traiter qu'une trame sur deux pour optimiser
    if process_this_frame:
        # Redimensionner et convertir l'image en RGB
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        rgb_small_frame = cv2.cvtColor(small_frame, cv2.COLOR_BGR2RGB)

        # Etape A et B : Detection et encodage
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
    face_locations)

```

```
        face_names = []
        for face_encoding in face_encodings:
            # Etape C : Comparaison
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = "Inconnu"

            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)

    process_this_frame = not process_this_frame

    # ... (partie affichage) ...
```

Listing 2 – Boucle principale pour le traitement en temps réel

Les optimisations clés ici sont le redimensionnement de l'image via `cv2.resize` et le traitement d'une trame sur deux, ce qui réduit considérablement la charge de calcul. La conversion de BGR (format OpenCV) vers RGB est une étape cruciale, car la bibliothèque `face_recognition` attend ce format.

4.4 Affichage des Résultats

La dernière partie de la boucle utilise les fonctions de dessin d'OpenCV pour afficher les résultats de manière visuelle sur la trame vidéo.

```
    # Afficher les resultats sur la trame video
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Redimensionner les coordonnees du visage
        top *= 4; right *= 4; bottom *= 4; left *= 4

        # Definir la couleur du cadre
        box_color = (0, 0, 255) if name == "Inconnu" else (0, 255, 0)

        # Dessiner un rectangle autour du visage
        cv2.rectangle(frame, (left, top), (right, bottom), box_color, 2)

        # Dessiner une etiquette avec le nom
        cv2.rectangle(frame, (left, bottom - 35), (right, bottom), box_color, cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255, 255, 255), 1)

    # Afficher l'image resultante
    cv2.imshow('Video', frame)

    # Appuyer sur 'q' pour quitter
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
```

Listing 3 – Affichage des résultats sur la vidéo

Notez que les coordonnées des visages, calculées sur l'image réduite, sont multipliées par 4 pour correspondre à la taille de la trame originale avant l'affichage.

5 Guide d'Utilisation

5.1 Prérequis et Installation

Pour exécuter ce projet, Python 3 doit être installé, ainsi que les bibliothèques suivantes :

— `face_recognition`

— `opencv-python`

— `numpy`

— `dlib`

L'installation peut être effectuée avec une seule commande pip :

```
pip install face_recognition opencv-python numpy dlib
```

Listing 4 – Commande d'installation des dépendances

5.2 Exécution du Programme

1. Cloner ou télécharger le projet.
2. Créer un dossier `database` à la racine.
3. Remplir la base de données en créant des sous-dossiers pour chaque personne et en y plaçant leurs photos.
4. Lancer le script depuis un terminal :

```
python reconnaissance_faciale.py

```

Listing 5 – Lancement du script

5. Une fenêtre s'ouvrira, affichant le flux de la webcam avec les identifications.
6. Pour quitter, appuyer sur la touche **'q'**.

6 Conclusion

Ce projet a permis de mettre en œuvre avec succès un système de reconnaissance faciale fonctionnel, performant et évolutif. En combinant la puissance de traitement d'images d'OpenCV et les algorithmes de deep learning de la bibliothèque `face_recognition`, nous avons créé une application qui illustre parfaitement les concepts clés de la vision par ordinateur.

L'architecture modulaire, basée sur une base de données de fichiers, rend le système particulièrement flexible. Les optimisations appliquées, telles que le redimensionnement des images et le traitement intermittent des trames, sont essentielles pour atteindre une performance acceptable en temps réel sur du matériel standard.

Ce travail constitue une base solide qui pourrait être étendue avec des fonctionnalités plus avancées, comme le suivi de personnes sur plusieurs trames, l'analyse des émotions ou l'intégration avec d'autres systèmes de sécurité.