Reconnaissance Faciale sur une Raspberry Pi 3 à l'aide d'une Raspberry Pi Caméra

Amir Ahmadi et Nithusan Sivakanthan

Decembre 14 2023

Master 1 Informatique
Université Paris 8
Novembre 2023

Table des matières

Part I
Introduction

1 Contexte

Le projet du cours de robotique, a pour but d'implémenter de l'Intelligence Artificielle sur un robot, ainsi nous avons décidé d'implementer sur notre robot, de la reconnaissance faciale qui peut identifié le visage de la personne si il est dans la base de données.

2 Objectifs

L'objectifs de ceux projet est de pouvoir l'utiliser dans le cadre de la sécurité, il peut etre mis en place dans un batiment pour enregister les données d'un visage lors d'une infraction par exemple. A notre echelle on l'utilise pour verifier si une personne detecter apartient a la liste des étudiants en M1 Informatique

3 Definitions

3.1 Reconnaissance faciale

La reconnaissance faciale est une technique qui permet à partir des traits de visage : D'authentifier une personne : c'est-à-dire, vérifier qu'une personne est bien celle qu'elle prétend être.

3.2 Raspberry Pi

Une "Raspberry Pi" est un nano ordinateur de la taille d'une carte de crédit que l'on peut brancher à un écran est utilisé comme un ordinateur standard . Voici la liste des entrée et sortie de la carte:

- Sortie HDMI
- Sortie Audio
- Entrées USB
- Entrée Micro carte SD
- Entrée Ethernet
- Entrée Module caméra

3.3 XML

Le XML est l'acronyme de eXtensible Markup Language, est un langage de balisage utilisé pour décrire et structurer des données de manière lisible aussi bien par les machines que par les humains.

Part II
Recherche

4 Environnement

Nous avons chercher sur quelle langage dévelopeer notre projet, ainsi nous avons trouvé que le langage Python est le langagae de programmation le plus populaires pour faire de la reconnaissance faciale. Pour ce qui est de l'editeur de texte nous avons utilisé le logiciel installé dans l'Os Rasbian Thonny.

Pour ce qui est du Materiels, nous avons decidé que une semaine sur deux, l'un de nous garder le materiels et developpé dessus, tant dis que l'autre developpe sur ca machine personnels. Mais au bout de quelques jours nous avons commencé a developpé sur nos machines, car les performence de la Raspberry Pi de 1Go de ram, nous ralentissez beaucoup, car l'utilisation de la cameraPI additioné au

algorithme de reconnaisance faciale prennait beaucoup de ressource, le programme prennait du temps a se lancé et parfois il arrive que la Raspberry Pi crash. Ainsi sur nos machines, nous developpions sur VisualStudioCode, est pour la camera, nous utilisions notre webcam.

5 Bibliothèques

Pour pouvoir effectuer de la reconnaissance faciale, nous avons besoin d'abord de pouvoir exploiter notre camera Pi. Ainsi nous avons trouvé la bibliothèques "PiCamera" qui nous permet de reconnaitre et l'utilisé

Ensuite pour reconnaitre un visage nous avons trouver un fichier XML fournit pas "OpenCv" qui est aussi une bibliothèques pour faire du traitement d'images

Nous penssions a ajouter une base de données, car pour reconnaitre une personne il faut d'abord avoir des données de cette personne ainsi il fallait trouver un moyen d'encoder, et de stocker. Pour cela nous avons decider d'utiliser la bibliothèques "Pickle" qui va nous permettre d'encoder des données et pour stocker les données nous avons decider d'utiliser une base de données en SqlLite

6 Reconnaissance faciale

6.1 Détection du visage

Avant de faire de la reconnaisance faciale, il faut premierement reconnaitre un visage dans une image. Un visage est composé de plusieurs caracteristiques tels que des yeux, sourcils, bouche, nez.

Ainsi pour ce faire la bibliothèques "OpenCV" met a disposition sur Github des fichiers XML qui contiennent des informations sur les caractiéristiques visuelles spécifiques à un visage qui sont utilisés pour entrainer un classificateur. Ces caractéristiques sont extraites à partir d'un grand nombre d'image avec des visages et d'images sans visages. Pour pouvoir faire de la reconnaisance de visage, nous avons donc décidé d'utilisé le fichier haarcascade-frontalface-default.xml, il va nous permettre de reconnaitre un visage.

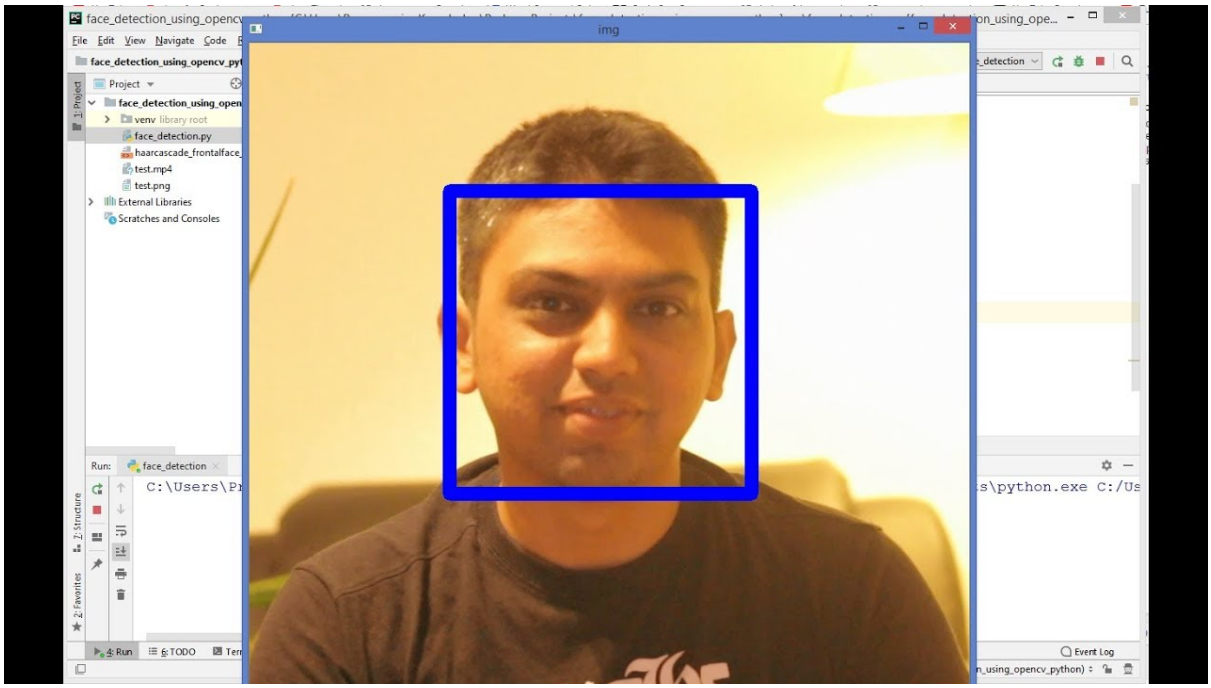


Figure 1: Image montrant les différents processus de l'algorithme

6.2 Reconnaissance d'un individu

Ensuite apres avoir reconnu un visage dans une image, il fallait reconnaitre a qui appartient le visage. Pour trouver a qui correspond le visage il fallait d'abord avoir des données d'un individu sous forme

d'images.

Ainsi apres avoir rassemblez plusieurs image differents du meme individu dans un seul dossier, il fallait entrainer un model de reconnaisance faciale avec les données de l'individu, ce qui va nous generer un fichier contenant les caracteristiques du visage de l'individu, c'est a dire, la position des yeux, du nez , de la bouche ecetera.

Enfin, avec ce fichier generer, nous avons plus qu'a compare les caracteristiques du visage detecter avec avec le fichier contenant les caracteristiques d'un individu.

7 Algorithme

7.1 Haar Cascade

- **Caractéristiques de Haar**
  - L'algorithme utilise des caractéristiques de Haar, qui sont des filtres rectangulaires qui sont appliqués sur une région de l'image pour calculer la différence entre la somme des pixels dans deux zones adjacentes. Ces caractéristiques peuvent être des rectangles adjacents, des lignes ou des blocs.
- **Création d'un Classifieur**
  - Un classifieur est construit en utilisant un ensemble d'images positives (contenant l'objet que vous souhaitez détecter, par exemple, des visages) et d'images négatives (ne contenant pas l'objet). Le classifieur est formé pour distinguer entre les deux en utilisant les caractéristiques de Haar. Classificateurs en Cascade : Le classifieur construit est ensuite organisé en une cascade de classificateurs en utilisant une technique d'apprentissage machine. Chaque étape de la cascade est un classifieur indépendant. Les classificateurs plus simples sont placés au début de la cascade, tandis que les classificateurs plus complexes sont placés vers la fin.
- **Intégration du Classifieur**
  - Une fois la cascade construite, l'algorithme utilise le classifieur intégré pour parcourir l'image en utilisant des fenêtres glissantes de différentes tailles. À chaque étape de la cascade, la fenêtre est évaluée avec un classifieur. Si la région échoue à une étape, elle est rapidement rejetée, ce qui permet d'accélérer le processus.
- **Faux Positifs et Faux Négatifs**
  - Le processus peut produire des faux positifs et des faux négatifs. Les faux positifs sont des régions incorrectement classées comme positives, tandis que les faux négatifs sont des régions positives manquées. Les poids sont ajustés pendant le processus d'apprentissage pour minimiser ces erreurs. Ajustement des Classificateurs : L'algorithme ajuste automatiquement les seuils des classificateurs pour minimiser les faux positifs tout en maintenant un taux élevé de détection.

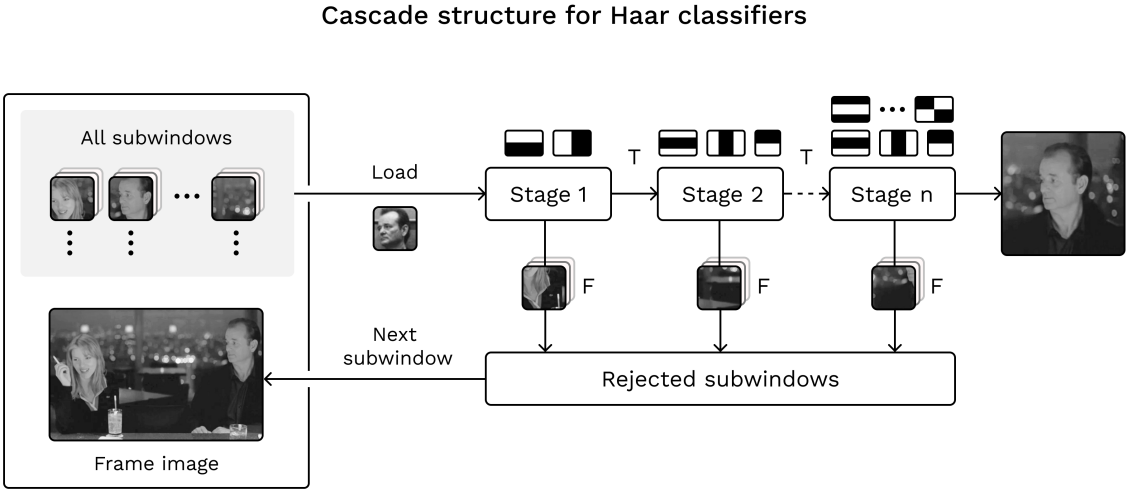


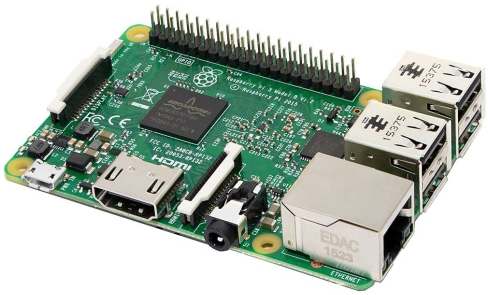
Figure 2: Image montrant la détection d'un visage

Part III
Developpement

8 Materiels

Madame Sediki nous a fournis une liste de matériels nécessaires pour réalisé notre projet, voici la liste:

- Raspberry Pi 3
- Adaptateur HDMI vers VGA
- Carte Micro SD
- Raspberry Pi Camera
- Cable d'alimentation



8.1 Installation

Pour pouvoir travaillé sur la Raspberry, nous avions d'abord besoin d'installer un Os, c'est a dire un systeme d'exploitation. Nous avions le choix entre ces deux Os :

- **Raspbian**
  - Raspbian est un système d'exploitation libre basé sur la distribution Linux Debian et optimisé pour le matériel de Raspberry Pi.
- **Ubuntu Mate**
  - Ubuntu Mate est également basé sur Debian et se révèle particulièrement utile pour avoir un ordinateur de bureau basé sur un Raspberry Pi.

Pour notre projet, nous avons décidé d'installé Raspbian, ou plutot Raspberry Pi OS qui est son nouveau nom. Pour se faire nous somme allé sur le site de Raspberry Pi, et avons téléchargé le version de Raspberry Pi Os Desktop, car nous avions besoin d'une interface pour pouvoir afficher le rendu de la camera. Nous avons ensuite utiliser Etcher pour pouvoir Flashé l'Os que l'on a telechargé sur notre carte SD. Enfin, nous inserons la carte SD dans la Raspberry Pi 3 et l'alimenton pour qu'il demarre et installe l'Os.

8.2 Configuration

Voici les differents commands pour finir la configuration de la Raspberry Pi.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

- **Réseau**
  - Nous avions besoin du réseau, pour pouvoir faire les mise a jours du systeme et aussi pour pouvoir installer des logiciels.
- **Mots de passe**

```
$ passwd
```

- **Mise à jour du système**
  - Il est important de travailler sur un systeme qui est a jours pour pouvoir travailler et executer notre projet . Voici les commandes utilisé pour mettre a jours le systeme.

```
$ sudo apt-get update
$ sudo apt-get upgrade
```

- **VNC Viewer**
  - Ce logiciel nous permet de controler la Raspberry Pi depuis nos machines, etant plus facile pour travailler.

9 Reconnaissance Faciale

Ce bout de code va nous permettre, premierement de charger le classifieur Haar Cascade, pour nous permettre de detecter un visage et ensuite de dessiner un rectangle sur le visage détecter.

9.1 Détection du visage avec la caméra Pi

Listing 1: Reconnaissance faciale

```
1
2       # On recupere le classifieur
3       face_cascade = cv2.CascadeClassifier(
4           'haarcascade_frontalface_default.xml'
5       )
6
7       # Boucle dans lequel on affiche la camera Pi on continue
8       while True:
9           for frame in cam.capture_continuous(rawCapture,
10                                              format="bgr",
11                                              use_video_port=True):
12              # On recupere la frame de la camera
13              image = frame.array
14
15              # On convertit la frame en nuance de gris
16              gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
17
18              # Avec le classifieur on detecte les visages presents da la frame
19              faces = face_cascade.detectMultiScale(gray, 1.1, 4)
20
21              # On dessine un rectangle autour des visages detecter
22              for (x, y, w, h) in faces:
23                 cv2.rectangle(image, (x, y), (x+w, y+h), (255, 0, 0), 2)
24                 cv2.putText(image,
25                             name,
26                             (x, y-10),
27                             cv2.FONT_HERSHEY_SIMPLEX,
28                             0.9,
29                             (36,255,12),
30                             2)
31              # On affiche la frame avec le rectangle sur le visage
32              cv2.imshow("Press", image)
33              rawCapture.truncate(0)
34
35              k = cv2.waitKey(30) & 0xff
36              leave = False$
37              if k == 27:
38                  print("Escape")
39                  leave = True
40                  break
41              if k == 115:
42                  photo = photo + 1
43                  print(photo)
44                  photo_name = 'db/screen/' + name + str(photo) + ".jpg"
45                  print(photo_name)
46                  cv2.imwrite(photo_name, image)
47          if leave:
48              break
49
50
51      cv2.destroyAllWindows()
```

9.2 Reconnaissance faciale

Listing 2: Fonction chapeau va reconnaitre l'individu présent sur l'image

```
1     def recognize_faces_video(im,
2     encodings_location: Path = DEFAULT_ENCODINGS_PATH,
3 ) -> None:
4     with encodings_location.open(mode="rb") as f:
5         loaded_encodings = pickle.load(f)
6
7     face_encodings = face_recognition.face_encodings(im)
8
9     for unknown_encoding in zip(face_encodings):
10        name = _recognize_face_video(unknown_encoding, loaded_encodings)
11        if not name:
12            name = "Unknown"
13        return name
```

Listing 3: Fonction qui va comparé les caracteristiques d'un individu avec la base de données encodé

```
1     def _recognize_face_video(unknown_encoding, loaded_encodings):
2     unknown_encoding = np.array(unknown_encoding)
3     boolean_matches = face_recognition.compare_faces(
4         loaded_encodings["encodings"], unknown_encoding
5     )
6     distance = face_recognition.face_distance(loaded_encodings["encodings"],
7                                               unknown_encoding)
8
9     list_distance = list(map(lambda x: round(x * 100), distance))
10    list_accuracy = list(map((lambda x: 100 - x), list_distance))
11    list_validation = list(
12        map((lambda ac: True if ac >= 50 else False),list_accuracy))
13
14    votes = Counter(
15        name
16        for match, name in zip(list_validation, loaded_encodings["names"])
17        if match
18    )
19    if votes:
20        return votes.most_common(1)[0][0]
```

On peut voir ligne 10 que l'on calcule la distance, il s'agit de la du pourcentage de ressemblence entre chaques caracteristiques des individus present l'encodage et l'individu present dans la frame de la webcam, ensuite ligne 11 je construit une liste qui remplit met false si la ressemblence est inferieur a 50 sinon true a l'index correspondendant aux données de l'individue.

Exemple: [true, false, true , false, false] equivalent à [Nithusan, Amir, Nithusan, Amir, Amir].

Ainsi, ligne 14 je compte le nombre de fois ou true apparait pour chaque individu, dans mon exemple cela va me donné votes = Nithusan: 2, Amir: 0.

Enfin je retourne l'individu qui a le plus de caracteristiques en commun de ma liste encodé.

10 Base de données

Il nous faut maintenant une base de données sur lequel l'algorithme Haar Cascade compare ces données de visage avec ceux des individues qu'on a creer, pour un base de données en SqlLite.

... Parti Amir Explication ...

10.1 Encodage des données des individus

Avant avoir creer une base de données d'individu, il nous fallait maintenant pouvoir transcrire les données des visages pressent sur chaque images dans un fichier. Ainsi nous avons creer une fonction qui parcourt une liste d'image donné, pour chaque image, on utilise la biblioteque face-recogntion pour localiser sur l'image le visage de l'individu et encoder les caracteristiques du visage de l'individu dans un fichier Pickle.

Listing 4: Fonction qui encode les caracteristiques du visage d'un indivudu

```
1
2       def encode_images_faces(name, images,
3       model: str = "hog") -> None:
4
5       encodings = []
6       names = []
7       name = name.split("-")
8       name = name[0] + "_" + name[1]
9       encodings_location: str = Path(f"db/encoding/liste_personne/{name}.plk")
10      for image_file in images:
11          face_reco_image = face_recognition.load_image_file(image_file)
12
13          face_locations = face_recognition.face_locations(face_reco_image,
14                                                           model=model)
15          face_encodings = face_recognition.face_encodings(face_reco_image,
16                                                           face_locations)
17
18          for encoding in face_encodings:
19              names.append(name)
20              encodings.append(encodings)
21
22      names_encodings = {"names": names, "encodings": encodings}
23
24      with encodings_location.open(mode="wb") as f:
25          pickle.dump(names_encodings, f)
26
27      return encodings_location
```

Part IV
Résultats

11 Analyse des résultats

Part V
Conclusion

Pour conclure, nous avons reussi a faire de la reconnaissance faciale a l'aide de la camera Pi, ce projet nous a donné un avant de gouts du fonctionnement de la reconnaissance faciale.