Haute École de Namur-Liège-Luxembourg

École d'ingénieurs
Département ingénieur industriel
de Pierrard-Virton

Année 2019- 2020

Master en Sciences de l'Ingénieur Industriel
Orientation Automatisation
Master 1

Rapport :
Systèmes Intelligents – Système de reconnaissance faciale pour surveiller l'accès à une résidence

Étudiants :
DUPONT Corentin
GOFFIN Gérôme
JOSIS Arnaud

Professeur :
Mme SLAMA Rim

Sommaire

1. Introduction 4

2. Descriptif du projet 5

3. Matériel utilisé 6

4. Planification 8

5. Reconnaissance faciale 11

5.1. Reconnaissance avec face recognition 11

5.2. Reconnaissance avec le modèle VGG 13

5.2.1. Détection & découpage du visage sur l'image 15

5.2.2. Analyse du visage 17

5.2.2.1. Structure du modèle VGG 17

5.2.2.2. Création du modèle 19

5.2.2.3. Création de la dataset 19

5.2.3. Prédiction du nom de la personne en fonction du visage 20

5.2.4. Mise en place du programme principal 21

6. Actionneurs 23

6.1. Servomoteur 23

6.2. Avertisseur sonore 24

6.3. Détection de mouvement 24

6.5. Association des programmes 25

7. Dashboard NodeRed 25

8. Communication MQTT 28

9. Lien Github 29

10. Améliorations 29

11. Conclusion 31

12. Bibliographie 32

Table des figures

Figure 1 : équipements utilisés dans le projet 5
Figure 2 : Tableau des coûts 7
Figure 3 : GANTT; tâches 1 , 2 et 3 9
Figure 4 : GANTT, tâches 4, 5 et 6 10
Figure 5 - FaceRecognition - encodage de la dataset 11
Figure 6 - FaceRecognition - prédiction d'une personne 12
Figure 7 - FaceRecognition - plusieurs personnes détectées sur l'image 13
Figure 8 - les 4 IA de la reconnaissance faciale 13
Figure 9 - agrandissement et rétrécissement de l'image 15
Figure 10 - principe de la détection et du découpage du visage 16
Figure 11 - structure du modèle VGG 17
Figure 12 - principe de la convolution 17
Figure 13 - fonction d'activation ReLu 18
Figure 14 - principe du max pooling 18
Figure 15 - formule cosine distance 20
Figure 16 - principe de fonctionnement de la fonction find_closest 20
Figure 17 - principe de fonctionnement du mainProgram 21
Figure 18 : Schéma de câblage des actionneurs 23
Figure 19 : Servomoteur utilisé 23
Figure 20 : Buzzer piézoélectrique utilisé 24
Figure 22 : Capteur PIR utilisé 24
Figure 21 : Capteur de luminosité utilisé (photorésistance) 24
Figure 22 : Algorithme du programme de gestion des actionneurs 25
Figure 23 : Dashboard 26
Figure 24 : Flow NodeRed, acquisition flux caméra 26
Figure 25 : Flow NodeRed, requête http 26
Figure 26 : Flow NodeRed pour l'implémentation des boutons et l'envoie sur les topics mqtt 27
Figure 27 - logo MQTT 28
Figure 28 - Erreur avec l'algorithme HaarCascade 30
Figure 29 - Algorithme HaarCascade sans personne sur l'image 30
Figure 30 - Algorithme predict shape (librairie dlib) 30

1. Introduction

Dans le cadre du cours de systèmes intelligents enseigné par Madame Slama, nous avons choisi de réaliser un projet utilisant les notions de Machine Learning et de Deep Learning vues au cours théorique. Des groupes de deux ou trois étudiants ont proposé des idées de projet en réalisant une liste de matériel, une planification prévisionnelle sous forme de Gantt Chart ainsi qu'une description du projet. Ces documents ont été soumis à une analyse par Madame Slama afin d'accepter la proposition du projet.

L'idée présentée dans ce rapport suivant notre proposition de projet est de réaliser un système de reconnaissance faciale, utilisé pour contrôler l'accès des personnes à leur résidence. L'algorithme de reconnaissance faciale devra être capable de prédire avec précision la personne voulant rentrer dans la maison. Cet algorithme pourra également reconnaître les personnes dans des conditions aléatoires, c'est-à-dire, lorsque celles-ci portent des lunettes, une casquette, un bonnet, une écharpe ou encore un masque, avec différentes conditions d'intensité lumineuse.

La liberté du choix du sujet du projet, ainsi que la mise en pratique des notions théoriques concernant le Machine Learning et le Deep Learning abordées pendant le cours, ont amplifié notre envie de réaliser ce projet. De plus, les recherches personnelles concernant l'utilisation de différents modèles pour le système de reconnaissance nous a permis d'en découvrir davantage sur les pratiques du Deep Learning, ce qui nous a poussé à s'impliquer dans ce travail.

Ce projet a été réalisé en plusieurs étapes dont les trois plus grandes sont la recherche, la création et l'utilisation de l'algorithme de reconnaissance faciale utilisé pour verrouiller, ou déverrouiller la porte d'entrée du domicile des résidents d'une maison. La communication entre le système de reconnaissance et les différents actionneurs utilisés dans ce projet sont un des autres points également abordés dans ce rapport.

2. Descriptif du projet

L'objectif du projet proposé est de réaliser un système de reconnaissance faciale pour ouvrir la porte d'entrée de la maison lorsqu'un des résidents s'en approche. Pour se faire, une caméra pi est placée près de la porte d'entrée, pointant vers les personnes souhaitant entrer. Les visages de tous les résidents de la maison sont préalablement enregistrés dans le système.

Le flux vidéo de la caméra est analysé par la Raspberry en temps réel. Si la personne est reconnue par le système et est autorisée à entrer, la porte est déverrouillée via l'Arduino par le biais d'une serrure électrique, modélisée par un servomoteur dans ce projet. Si cette personne n'est pas reconnue par le système ou si elle n'est pas autorisée à entrer dans la maison, une sonnette retentit pour avertir les résidents qu'un invité se trouve sur le pas de la porte. De plus, le flux vidéo de la caméra est également affiché sur un écran à l'intérieur de la maison, modélisé par un écran d'ordinateur dans le cadre de notre projet.

Dans le cas où il fait sombre et qu'une personne est détectée par le capteur de mouvement, la lumière extérieure est allumée par l'Arduino durant un certain temps. Ceci permet d'accueillir le résident ou l'invité et permet également d'éclairer un minimum le visage du résident pour assurer le bon fonctionnement du système de reconnaissance, comme la caméra utilisée ne possède pas de système infrarouge.

Le point suivant aborde les équipements utilisés dans ce projet, illustrés par l'image suivante.

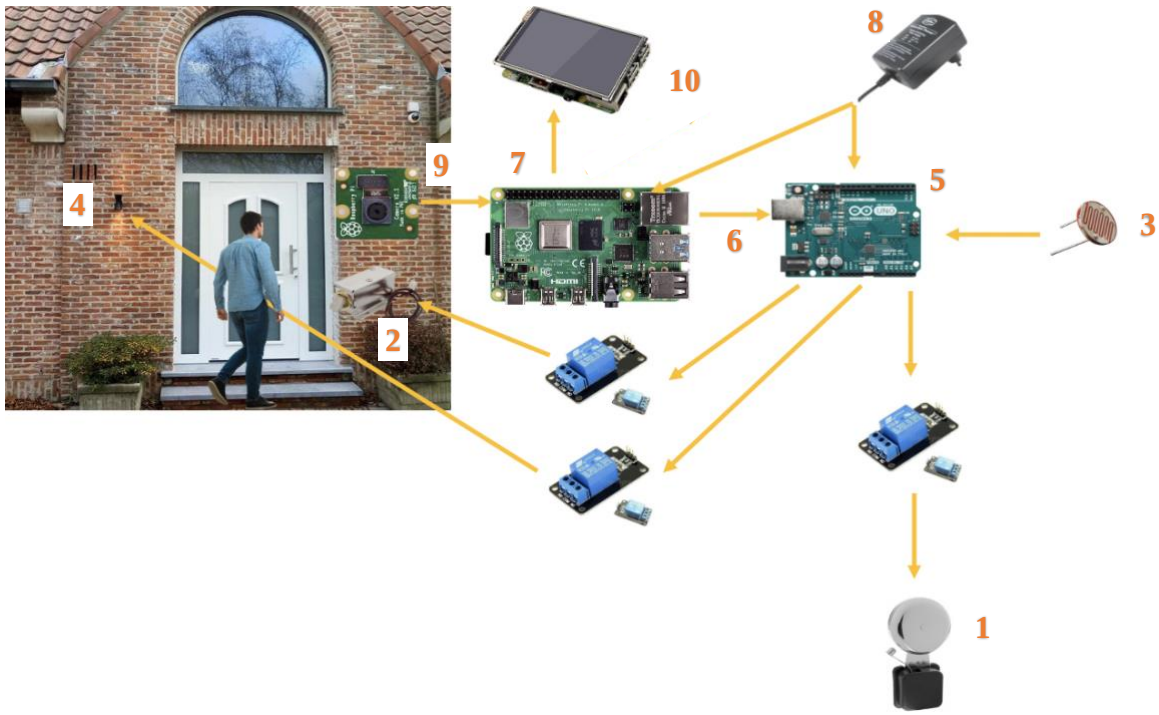


Figure 1 : équipements utilisés dans le projet

3. Matériel utilisé

Les équipements nécessaires pour la réalisation du projet ont été décomposés en deux grandes parties :

- **La partie actionneurs:**
  Cette partie est composée d' :
  - un buzzer piézoélectrique qui représente la sonnette *(1)*.
  - un servomoteur pour verrouiller ou déverrouiller la porte d'entrée *(2)*.
  - un capteur de lumière *(3)* et une LED jaune *(4)* pour simuler, sur la maquette, l'éclairage que donnerait un spot LED à l'extérieur.

  Tous ces éléments sont commandés via un Arduino Uno *(5)* fourni par l'école sous forme d'un kit (super starter kit R3 project). Un câble USB-A vers USB-B *(6)* est utilisé pour réaliser une communication série entre l'Arduino Uno et le Raspberry Pi 4 *(7)*. En plus du kit Arduino, une alimentation 9V *(8)* ainsi qu'un capteur de mouvement sont utilisés.

- **La partie concernant la reconnaissance faciale comprend tous les équipements nécessaires pour reconnaître une personne à l'entrée de la maison.**
  Cette partie est composée :
  - D'un kit Raspberry Pi 4 1Go de ram (Pi4-1G + 32G) fourni par l'école *(7)*.
  - D'une caméra pi (RPI-CAM-V2) fournie également par l'école *(9)*.
  - L'écran tactile *(10)* utilisé avec la Raspberry a été remplacé par un écran de pc, amplement suffisant pour notre utilisation.

Le tableau suivant liste de manière exhaustive l'ensemble des équipements utilisés dans ce projet.

Part		Components	Unit price	Quantity	Total price	Provider	Link
Actuator	Arduino kit provided by the school Ref of the kit : Super Starter Kit R3 project	Arduino uno	0,00 €	1	0,00 €	School	https://www.amazon.fr/Elegoo-Démarrage-dUtilisation-Débutants-Professionnels/dp/B01JD2Z5XW/ref=sr_1_2_sspa?__mk_fr_FR=ÅMÅŽÕÑ&dchild=1&keywords=kit+arduino&qid=1585646662&sr=8-2-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEzUVpRMUFURVJMU080JmVuY3J5cHRlZElkPUEwOTkxNDU1MkxlTkdOWlRaNTRDUSZlbmNyeXB0ZWRBZElkPUEwNjYwNzM4MkFIQ1o0QkxPWVVDTiZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=
	To order	Motion sensor	9,99 €	1	9,99 €	Amazon	https://www.amazon.fr/EFISH-Adaptateur-dalimentation-transformateur-Radiowecker/dp/B07D9DTWYS/ref=sr_1_9?__mk_fr_FR=ÅMÅŽÕÑ&keywords=alimentation%2Barduino&qid=1585122844&sr=8-9&th=1
		LED projector	11,99 €	1	11,99 €	Amazon	https://www.amazon.fr/Lumare-Projecteur-ultra-connecteur-Blanc/dp/B07BRW337F/ref=sr_1_28?__mk_fr_FR=ÅMÅŽÕÑ&crid=3FMF0R42XE0DI&dchild=1&keywords=spot%2Bled%2B10w&qid=1585645985&sprefix=spot%2Bled%2B10%2Caps%2C176&sr=8-28&th=1
		9v power supply (for Arduino)	10,99 €	1	10,99 €	Amazon	https://www.amazon.fr/EFISH-Adaptateur-dalimentation-transformateur-Radiowecker/dp/B07D9DTWYS/ref=sr_1_9?__mk_fr_FR=ÅMÅŽÕÑ&keywords=alimentation%2Barduino&qid=1585122844&sr=8-9&th=1
		Raspberry touchscreen	69,99 €	1	69,99 €	Amazon	https://www.amazon.fr/dp/B07LF1GYXS/?coliid=I2XR1RZ4HXKKH&colid=3H0UF0JNNWG1I&psc=1&ref_=lv_ov_lig_dp_it_im
Facial recognition	Provided by the school Ref of the raspberry kit : Pi4-1G+32G Ref of the pi camera : RPI-CAM-V2	Raspberry pi 4B 1Go kit	0,00 €	1	0,00 €	School	https://www.amazon.fr/TICTID-Alimentation-Interrupteur-Ventilateur-Dissipateur/dp/B07ZF73RB5/ref=sr_1_7?__mk_fr_FR=ÅMÅŽÕÑ&dchild=1&keywords=raspberry+pi+4&qid=1585647417&sr=8-7
		Pi camera	0,00 €	1	0,00 €	School	https://www.amazon.fr/Raspberry-Pi-1080p-Module-Caméra/dp/B01ER2SKFS/ref=sr_1_1_sspa?__mk_fr_FR=ÅMÅŽÕÑ&dchild=1&keywords=RPI+cam+v2&qid=1585647710&sr=8-1-spons&psc=1&smid=A3VFEKEXULHMZR&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyUU9OR0dDSjlSSUQ2JmVuY3J5cHRlZElkPUEwMjA2ODk4Mkk4QTc5Q1JQV1FKNiZlbmNyeXB0ZWRBZElkPUEwNzQ5Mjk5MlZPQkpFMVk5WDg4WSZ3aWRnZXROYW1lPXNwX210ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=

Total :	102,96 €

Figure 2 : Tableau des coûts

4. Planification

L'idée du projet de reconnaissance faciale pour gérer l'accès à la résidence a été la première étape de ce travail. Celle-ci a été validée par Madame Slama pour pourvoir ensuite réaliser l'analyse de ce projet correspondant à la deuxième étape. Celle-ci comprenait une description du projet proposé, une liste du matériel nécessaire pour la réalisation du projet, une recherche concernant les ressources à utiliser ainsi qu'un diagramme de Gantt présenté par les images ci-dessous. Il a été décomposé en plusieurs parties afin de planifier au mieux la réalisation de ce projet.

Les parties 3 et 4 sont les plus importantes du projet, elles concernent la conception du système de reconnaissance faciale (partie 3) et la programmation des actionneurs (partie 4). Afin d'agrémenter ce projet par une approche plus visuelle lors de la démonstration, une maquette représentant une façade de maison a été réalisée (partie 5). Les parties concernant la conception de l'algorithme de reconnaissance faciale et la programmation des actionneurs ont été divisées en palier avec des objectifs définis. Elles ont également été réalisées en parallèle afin d'optimiser au maximum le temps qui était imposé.
L'algorithme de reconnaissance faciale devait reconnaître une seule personne dans des conditions fixes (sans chapeau, sans lunettes) comme premier palier. Le second palier consistait à reconnaître 3 personnes dans les mêmes conditions fixes puis, ensuite, dans des conditions aléatoires telles que le port d'une écharpe, d'un bonnet ou encore de lunettes suivant différents niveau de luminosité. Les derniers paliers concernaient la réalisation d'une interface utilisateur afin de pouvoir ajouter et supprimer des personnes dans la base de données du système de reconnaissance.
La programmation des actionneurs a également été réalisée sous forme de paliers où les actionneurs étaient implémentés les uns après les autres dans le programme principal avant de mettre en place la communication avec le système de reconnaissance faciale.

Le point 6 constituait la rédaction de ce présent rapport ainsi que la présentation du projet.

Le prochain titre aborde la troisième partie du diagramme de Gantt à savoir la réalisation du système de reconnaissance faciale.

Diagramme de Gantt

Intitulé du projet : **Home Monitoring system**

Date : 25/03/20

	Détail des tâches	MARS		AVRIL				MAI	
		S13	S14	S15	S16	S17	S18	S19	S20
1	**Présentation de l'idée du projet**								
1.1	Réalisation d'un schéma et d'un texte pour expliquer notre idée du projet	Tous							
1.2	Présentation de l'idée à Mme Slama	Tous							
2	**Analyse du projet**								
2.1	Rédaction du document Word pour l'analyse								
2.1.1	Recherches des équipements nécessaires au projet (liste de matériel)	Gérôme et Corentin							
2.1.2	Description et diagramme du projet	Tous							
2.1.3	Réalisation du diagramme de gantt	Arnaud							
2.1.4	Recherches des ressources nécessaires								
2.1.4.1	Réalisation des 2 TPs du DeepLearning	Tous							
2.1.4.2	Recherches des ressources nécessaires pour l'algorithme de reconnaissance faciale	Gérôme et Arnaud							
2.2	Réalisation du PPT pour l'analyse	Tous							
2.3	Présentation de l'analyse du projet		Tous						
3	**Partie reconnaissance faciale**								
3.1	Reconnaître une personne dans des conditions fixes		Gérôme et Arnaud						
3.2	Reconnaître les 3 étudiants du groupe dans des conditions fixes			Gérôme et Arnaud					
3.3	Reconnaître les 3 étudiants du groupe dans des conditions aléatoires				Gérôme et Arnaud				
3.4	Reconnaître tous les étudiants de la classe dans des conditions aléatoires (option)								
3.5	Apprendre de nouvelles personnes						Gérôme et Arnaud		
3.6	Réalisation d'un interface utilisateur sur l'écran tactile (option)				Corentin				
3.7	Réalisation d'un système de reset pour la Raspberry						Arnaud		

Figure 3 : GANTT; tâches 1 , 2 et 3

Diagramme de Gantt

Intitulé du projet : **Home Monitoring system**

Date : 25/03/20

	Détail des tâches	MARS		AVRIL				MAI	
		S13	S14	S15	S16	S17	S18	S19	S20
4	**Partie actionneur**								
4.1	Programmer la serrure électrique		Corentin						
4.2	Programmer la sonnette			Corentin					
4.3	Programmer l'éclairage extérieur		Corentin						
4.4	Structurer la communication entre la Raspberry et l'Arduino			Tous					
4.5	Tester la communication entre l'Arduino et la			Corentin					
4.6	Mise en place de la communication entre l'Arduino et la Raspberry avec les équipements				Corentin				
5	**Réalisation de la maquette**								
5.1	Construction de la maquette	Corentin							
5.2	Placement des composants sur la maquette		Corentin				Corentin		
5.3	Câblage des composants sur la maquette		Corentin				Corentin		
5.4	Tests du fonctionnement du système de reconnaisance sur la maquette						Tous		
6	**Rapport et présentation**								
6.1	Rédaction du rapport					Tous			
6.2	Réalisation du PPT						Tous		
6.3	Montage de la vidéo du test réel du projet						Tous		
6.4	Remise du projet							Tous	
6.5	Remise du rapport							Tous	
6.6	Présentation								Tous

Figure 4 : GANTT, tâches 4, 5 et 6

5. Reconnaissance faciale

5.1. Reconnaissance avec face recognition

La première partie de nos recherches s'est dirigée vers la librairie « FaceRecognition » découverte dans l'article [1] de Adrian Rosebrock. Celui-ci se base sur l'utilisation d'OpenCV, de python ainsi que le Deep Learning. L'apprentissage ainsi que la reconnaissance des visages utilisent une technique appelée « deep metric learning » qui, au lieu de réaliser une classification d'image, produit un vecteur caractéristique composé de 128 valeurs de type réelles qui définissent la personne sur l'image. Le modèle utilisé dans la librairie se base sur celui du « ResNet-34 ». Ce réseau, entrainé par Davis King à l'aide d'environ 3 millions de données, peut atteindre une précision de 99,38% suivant l'article consulté.

Cette phase de test a été principalement décomposée en deux parties, une pour l'encodage et l'autre pour la reconnaissance.
La première partie concernant l'encodage contient :

- Un dossier nommé « dataset », avec les photos des résidents préalablement enregistrées dans des dossiers nommés par leur nom et prénom sous le format suivant Prénom_Nom comme par exemple Arnaud_Josis.
- Un programme nommé « encode_faces.py » qui utilise ces images en les faisant passer dans le modèle pour enregistrer les 128 valeurs caractérisant chaque personne dans un dictionnaire nommé « encodings.pickle ».

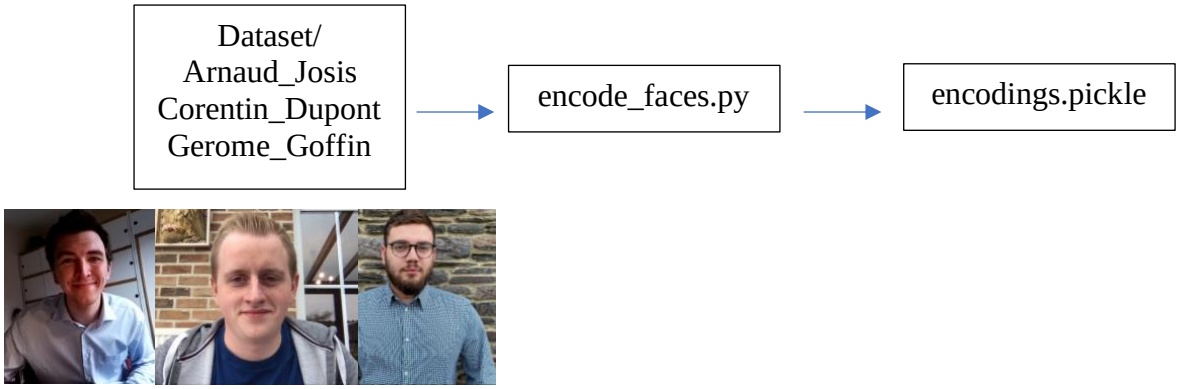


Figure 5 - FaceRecognition - encodage de la dataset

[1] https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

La deuxième partie contient :

- Un dossier nommé « examples », qui contient les images à soumettre lors de la prédiction.
- Un programme nommé « recognize_faces_image.py » qui utilise le dictionnaire « encodings.pickle » ainsi qu'une image du dossier « examples » pour réaliser la prédiction.

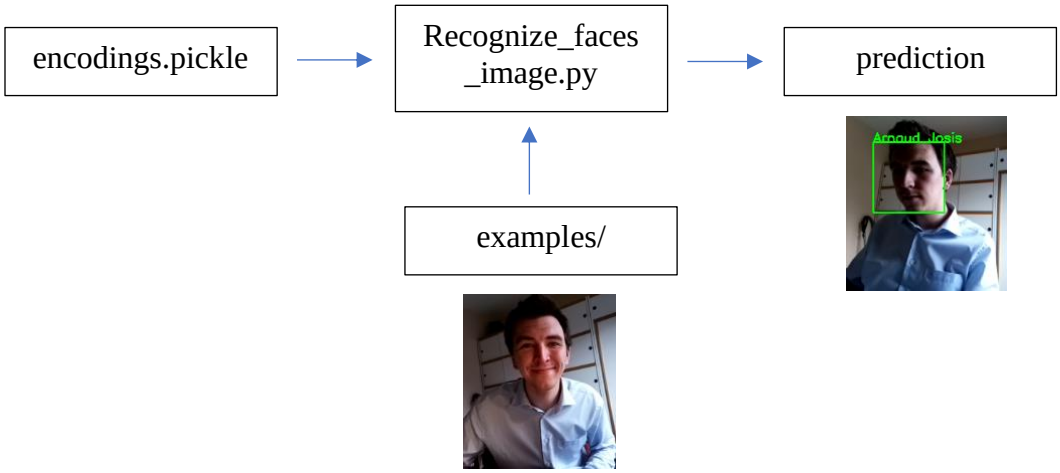


Figure 6 - FaceRecognition - prédiction d'une personne

Ce programme fait passer dans le modèle l'image en entrée, afin d'en sortir les 128 valeurs caractéristiques de la figure de la personne. Ensuite, ces valeurs sont comparées avec celles contenues dans le dictionnaire « encodings.pickle » en calculant la distance euclidienne, afin de ressortir un tableau remplit de valeurs booléennes (vrai ou faux) pour chaque image de la dataset. Si la distance est inférieure au seuil, la valeur vraie est retournée, ce qui indique une correspondance des images, sinon, si la distance est supérieure au seuil, la valeur fausse est retournée signifiant que les images ne correspondent pas. Les valeurs vraies sont comptabilisées pour chaque nom enregistré dans la dataset, et le nom contenant le plus d'occurrences est celui qui correspond à la personne sur l'image.

L'image est envoyée dans l'algorithme Haar cascade[2], qui détecte le visage, puis l'entoure d'un cadre vert et affiche le nom de la personne sur celui-ci. Dans le cas où le visage ne correspond pas à une personne de la dataset, le nom « Unknown » est affiché.

Il a également l'avantage de pouvoir reconnaitre plusieurs personnes sur une même image, comme le montrent les images suivantes. Lorsqu'il y a plusieurs personnes sur une image, l'algorithme est exécuté plusieurs fois d'affilée pour effectuer la reconnaissance (le nombre d'itérations correspond au nombre de visages détectés).

[2] https://github.com/opencv/opencv/tree/master/data/haarcascades

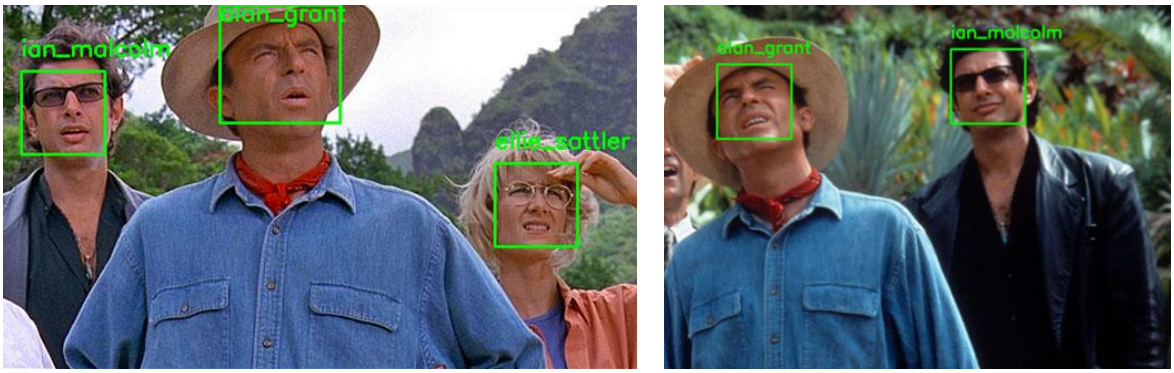


Figure 7 - FaceRecognition - plusieurs personnes détectées sur l'image

5.2. Reconnaissance avec le modèle VGG

La deuxième partie de nos recherches s'est dirigée vers la conception et l'implémentation d'un modèle souvent utilisé dans le domaine de la reconnaissance faciale à savoir le modèle VGG, suivant l'article[3] proposé par « pensée artificielle ». De plus, ce modèle est assez récent puisqu'il a été développé en 2014, contrairement à d'autres algorithmes qui datent parfois des années 2000. Comme le montre l'image ci-dessous, cet article se base sur 4 intelligences artificielles pour prédire la personne :

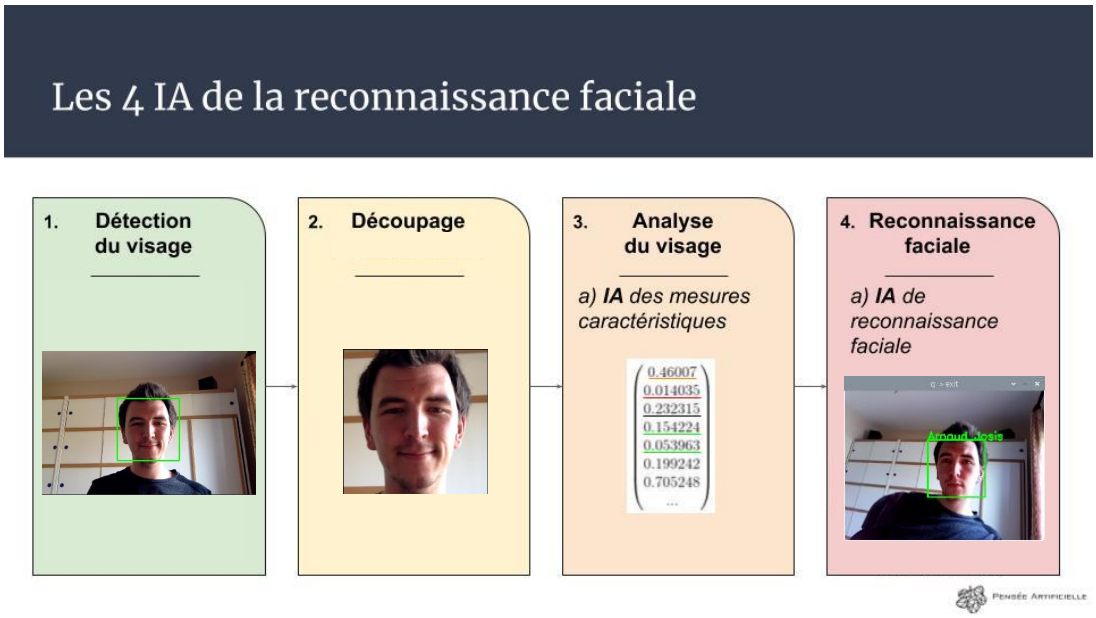
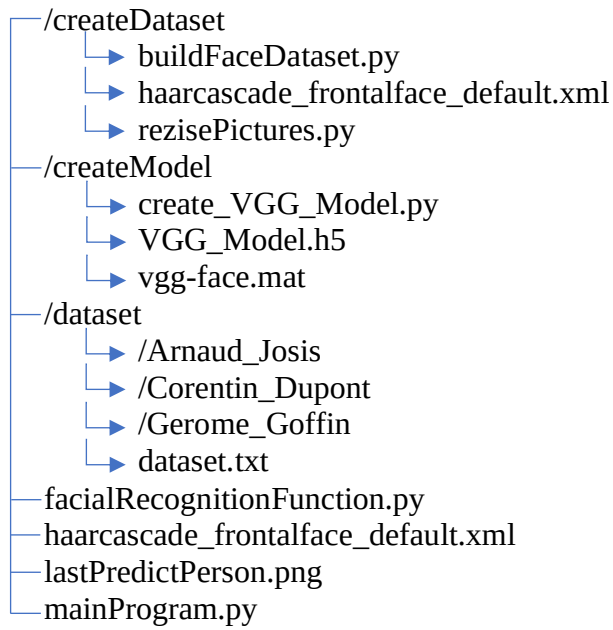


Figure 8 - les 4 IA de la reconnaissance faciale

Premièrement, l'intelligence artificielle trouve le visage sur l'image à l'aide de l'algorithme Haar Cascade[4] dans notre cas. Une fois que l'emplacement du visage est trouvé, l'image est découpée pour envoyer uniquement la tête à l'algorithme d'analyse du visage. Celui-ci est basé sur le modèle VGG et prend en entrée l'image du visage, pour ressortir 2622 valeurs caractérisant le visage de l'utilisateur. Grâce à une dernière IA, l'image est analysée et comparée aux données enregistrées afin de déterminer et afficher le nom de la personne.

3 http://penseeartificielle.fr/tp-reconnaissance-faciale/
4 https://github.com/opencv/opencv/tree/master/data/haarcascades

L’arborescence de ce projet se présente de la manière suivante :



Le dossier « createDataset » contient deux programmes permettant d’enregistrer une nouvelle personne dans le système. Le premier programme nommé « buildFaceDataset.py » permet d’enregistrer une nouvelle personne dans le système à partir de la piCaméra. Le deuxième programme nommé « rezisePictures.py » permet d’enregistrer, dans un format uniforme, une nouvelle personne dans le système à partir d’images enregistrées dans le dossier /home/pi/Downloads de la Raspberry par exemple. Le fichier « haarcascade_frontalface_default.xml » est utilisé par ces programmes pour détecter le visage d’une personne sur une image.

Le dossier « createModel » contient tous les outils nécessaires pour créer le modèle VGG, tel que « create_VGG_Model.py » permettant la création du modèle en implémentant le fichier « vgg-face.mat », contenant les poids de chaque liaison de chaque couche du modèle. Le fichier « VGG_Model.h5 » contient le modèle final et est implémenté dans le programme principal.

Le dossier « dataset » contient plusieurs sous-dossiers nommés avec le nom de la personne sous la forme « Prénom_Nom » (sans accentuation), contenant eux-mêmes les images de la personne. Le fichier « dataset.txt » contient les 2622 valeurs qui décrivent une personne, multipliés par le nombre de photos de cette même personne, multipliés par le nombre de personne dans le dossier dataset (cette notion sera expliquée plus loin dans ce rapport).

Le « mainProgram.py » utilise les différentes fonctions définies dans le fichier « facialRecognitionFunction.py » pour proposer les modes d’ajout d’une nouvelle personne dans le système, de supprimer une personne du système, de réaliser de la prédiction en live ou encore de quitter le programme. Le fichier « haarcascade_frontalface_default.xml » joue le même rôle que dans le dossier « createDataset » et permet de détecter les visages des personnes en live. La photo « lastPredictPerson.png » est une sauvegarde de la dernière personne reconnue par le système pour l’afficher.

5.2.1. Détection & découpage du visage sur l'image

Ce point aborde les deux premières intelligences artificielles de la précédente image à savoir la détection et le découpage du visage. La fonction permettant de réaliser la détection et le découpage du visage de la personne sur l'image est nommée « auto_crop ». Elle est implémentée dans le « mainProgram.py » par l'appel de notre toolbox « facialRecognitionFunction.py ».

L'image qui entre dans cette fonction possède 3 couleurs (RGB) et est convertie en niveau de gris avant d'être envoyée à l'algorithme de détection de visage réalisé par le HaarCascade. Une fois un visage détecté, les coordonnées du bord inférieur gauche ainsi que sa hauteur et sa largeur sont enregistrées dans des variables. Celles-ci sont utilisées pour découper le visage sur l'image RGB et former une nouvelle image. Celle-ci est ensuite redimensionnée en 224*224 pixels et est ré-envoyée en sortie de la fonction. L'interpolation cv2.INTER_AREA, qui utilise la relation entre les zones de pixels, a été choisie pour redimensionner l'image. Cette interpolation permet d'avoir des bonnes performances, aussi bien pour agrandir l'image que pour la rétrécir.

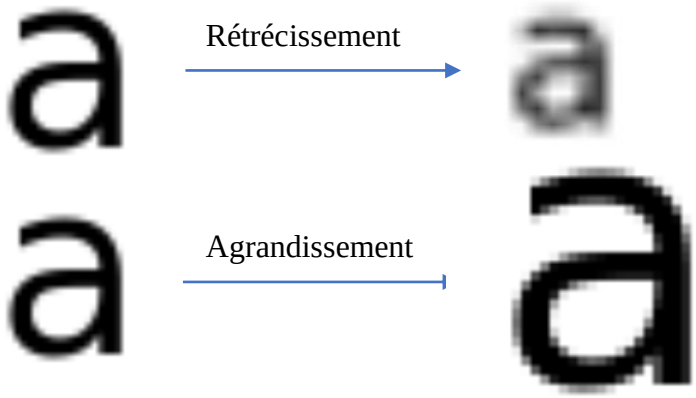


Figure 9 - agrandissement et rétrécissement de l'image

La figure suivante résume les actions réalisées sur l'image par la fonction « auto_crop ».

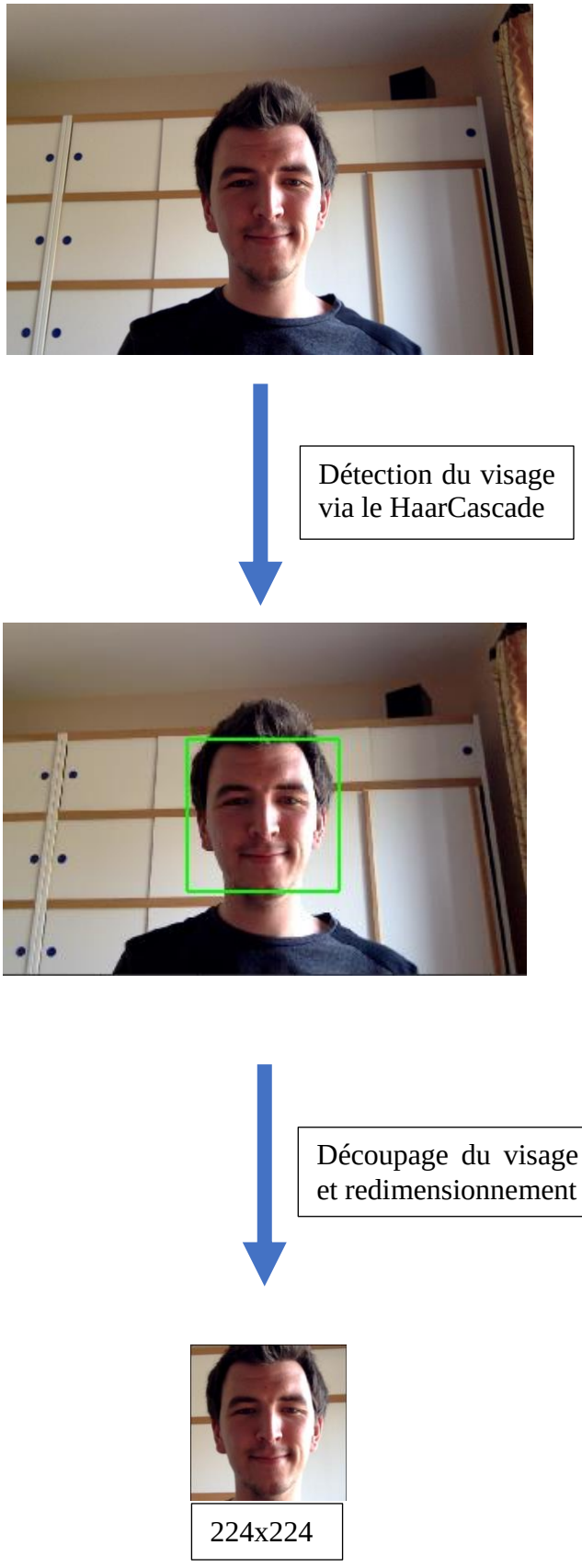


Figure 10 - principe de la détection et du découpage du visage

5.2.2. Analyse du visage

Comme précisé plus haut, la seconde étape de tests utilise un modèle basé sur la structure VGG. Cette partie réalise l'analyse du visage et utilise une intelligence artificielle (plus particulièrement le Deep Learning), par analogie à la troisième partie de la figure introductive (les 4 intelligences artificielles de la reconnaissance faciale). Les points suivants expliquent son contenu et sa création.

5.2.2.1. Structure du modèle VGG

Nous nous sommes basés sur le modèle VGG car celui-ci est assez récent et assez courant dans le domaine de la reconnaissance, comme utilisé dans l'article précédemment cité.

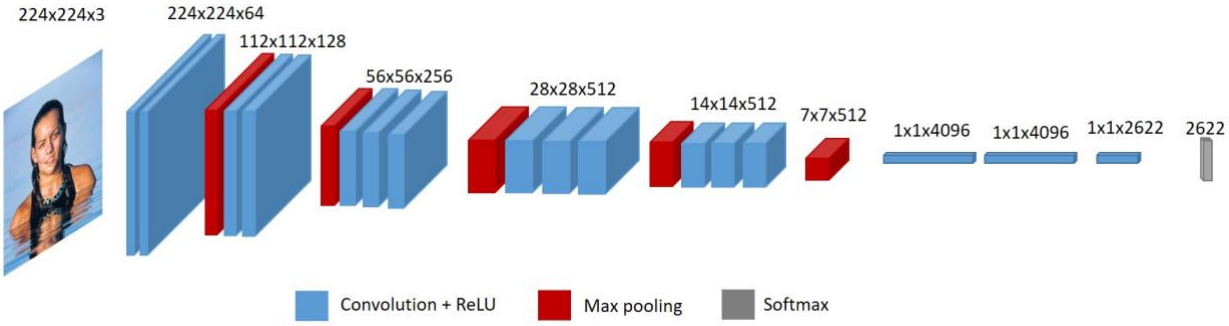


Figure 11 - structure du modèle VGG

Comme le montre la figure ci-dessus, le modèle VGG prend en entrée une image RGB (224 * 244 pixels * 3 couleurs), puis réalise une succession de Convolutions + ReLU, de Max pooling et pour finir un Softmax.

Le principe du Convolution + ReLu consiste à prendre l'image d'entrée et de réaliser une convolution (dot product) entre le filtre (aussi appelé kernel) et l'image comme il est possible de le visualiser sur la figure suivante :

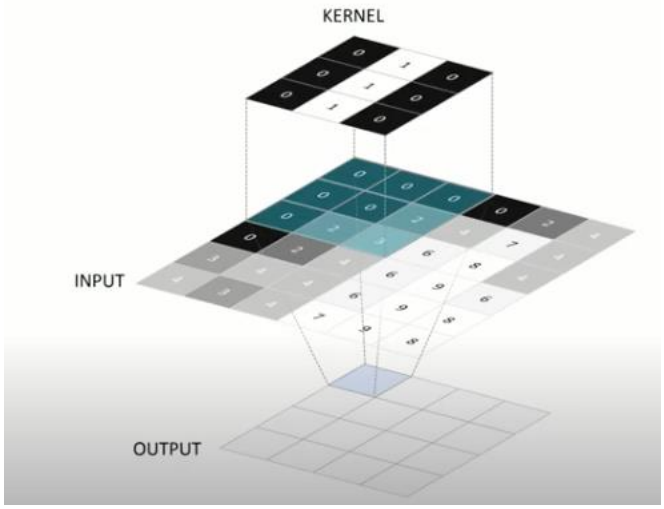


Figure 12 - principe de la convolution

Le filtre utilisé pour le modèle VGG est de taille 3x3 et est rempli par des valeurs entre 0 et 1 de manière aléatoire. Ces valeurs sont ensuite ajustées lors de l'entrainement du modèle.
Le résultat de la première convolution entre le filtre et l'image en entrée est placée dans la première case en haut à gauche de la sortie. Le filtre se déplace ensuite d'une case vers la droite (paramètre strides = 1 unité d'avancement) et une autre convolution est réalisée entre l'image et le filtre. Le résultat est placé dans la deuxième case à droite de la précédente. Cette opération est réalisée ainsi de suite sur l'entièreté de l'image. Ainsi, pour les deux premières convolutions, 64 filtres sont utilisés pour réaliser la convolution. Ensuite, il y a 112 filtres. Puis 256 et pour terminer, 512 filtres comme visualisé sur l'image de la structure du modèle VGG. Sur l'image d'exemple, il est possible de remarquer que l'image de sortie est de taille 4x4 et que l'image d'entrée est de 6x6 alors que sur le modèle VGG, les deux images ont la même taille. Ceci vient du fait que le paramètre « padding » a été placé sur « same » ce qui permet de garder la même taille en entrée et en sortie.
La fonction d'activation est une fonction qui décide si la sortie « d'un neurone » peut être propagée ou non. Dans ce cas, la fonction d'activation ReLu (Rectified Linear) a été utilisée. Elle sélectionne le maximum entre 0 et x.

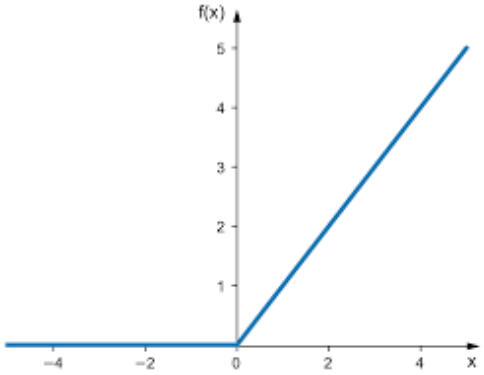


Figure 13 - fonction d'activation ReLu

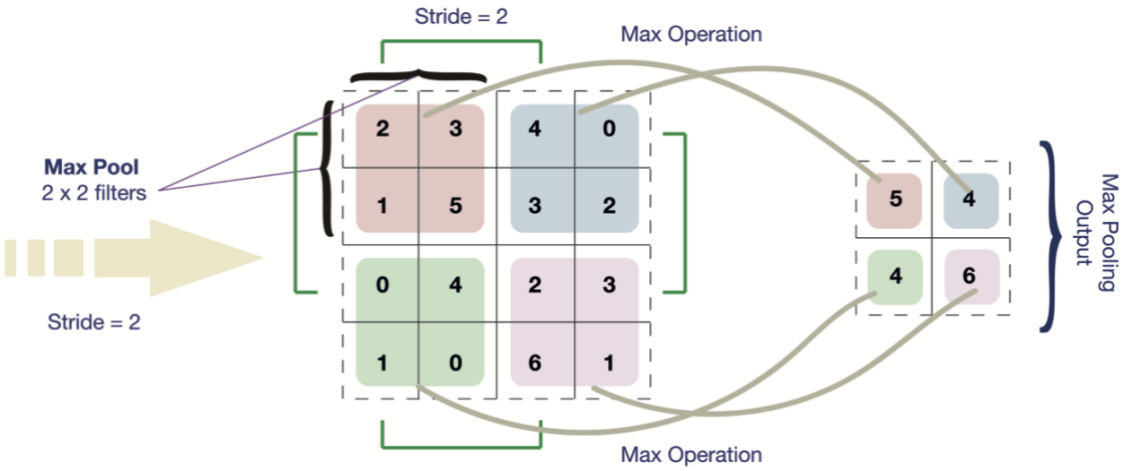


Figure 14 - principe du max pooling

Le principe du Max pooling est de prendre des blocs de pixels, de taille 2*2 dans ce cas, puis de sélectionner la valeur du pixel prédominant (le maximum). Chaque pixel en sortie montre donc la caractéristique de chaque sous-région. Entre chaque opération, les sous régions ne se chevauchent pas. Comme 4 pixels sont remplacés par 1 seul, la largeur et la hauteur de l'image est donc à chaque fois divisée par 2.

Cette action permet de réduire considérablement le nombre de paramètres que l'algorithme doit apprendre.

Finalement, la fonction Softmax attribue des probabilités à chaque classe, afin que la somme de ces probabilités soit égales à 1, ce qui permet de faire converger plus rapidement l'apprentissage. Les valeurs en sorties seront donc comprises entre 0 et 1. Suivant l'article, les valeurs utilisées sont celles juste avant le Softmax.

5.2.2.2.Création du modèle

Le modèle a été créé couche par couche, suivant la première figure de la structure du modèle VGG. Pour rappel, ce modèle prend en entrée une image de 224*224 pixels en RGB, et la fait passer dans le réseau de neurones pour en ressortir les 2622 valeurs caractérisant le visage de la personne sur la photo. Les différents poids des liaisons entre les neurones sont importés du fichier « vgg-face.mat » et sont directement appliqués sur les liaisons du modèle.

Dans le but de pouvoir utiliser le modèle dans la fonction principale, celui-ci est sauvegardé dans un fichier nommé « VGG_Model.h5 ».

5.2.2.3.Création de la dataset

Dans un premier temps, la personne peut être ajoutée dans le système de reconnaissance faciale (dans le dossier « dataset »), soit via la prise de photo par la piCamera (programme « buildFaceDataset ») soit via des photos téléchargées (programme « resizePictures »).

Dans un deuxième temps, lorsque les photos de la personnes ont été ajoutées dans le dossier « dataset », la fonction « generate_database » de la toolbox (facialRecognitionFunction) est appelée. Cette fonction envoie chacune des photos de la nouvelle personne dans le modèle afin de ressortir 2622 valeurs caractérisant la figure de la personne pour chacune des photos. Le dictionnaire (dataset.txt) permet de lier le nom de la personne à ces 2622 valeurs, multiplié par le nombre de photos enregistrées. Ce dictionnaire sera modifié avec le nom et les valeurs caractéristiques de la nouvelle personne. Il sera finalement réenregistré dans le fichier dataset.txt pour être utilisé ultérieurement.

5.2.3. Prédiction du nom de la personne en fonction du visage

La dernière intelligence artificielle, autrement dit la partie reconnaissance, est utilisée pour déterminer le nom de la personne présente sur le flux vidéo de la piCaméra. Une fois que le visage d'une personne est détecté par l'algorithme HaarCascade, le visage de la personne est découpé par la fonction « auto_crop » et est envoyé à la fonction « find_closest ». Celle-ci soumet l'image découpée au modèle afin d'obtenir les 2622 valeurs caractérisant la figure de la personne se présentant à l'entrée de la porte.

Le dictionnaire « dataset.txt » (contenant les 2622 valeurs multipliées par le nombre de photo de chaque personne, multiplié par le nombre de personne dans le dossier dataset) est chargé et la comparaison commence. Celle-ci consiste à calculer la « cosine distance » du vecteur formé par les 2622 valeurs de chacune photo, de chacune des personnes enregistrées dans la dataset, avec le vecteur (formé par les 2622 valeurs) de la personne à prédire. La valeur se trouve entre 0 et 1 et est calculée à partir de cette formule où les vecteur u et v correspondent au vecteur de l'image à prédire et aux vecteurs successifs enregistrés dans la dataset.

The Cosine distance between u and v, is defined as

$$1 - \frac{u \cdot v}{||u||_2 ||v||_2}.$$

where $u \cdot v$ is the dot product of u and v.

Figure 15 - formule cosine distance

Au plus la valeur est proche de 0, au plus la figure à prédire se rapproche de Gérôme dans l'exemple illustré à la figure suivante. Si la valeur est supérieure au seuil de détection, la personne est considérée comme inconnue (unknown). Après plusieurs, le seuil de détection de 0,23 a été adopté.

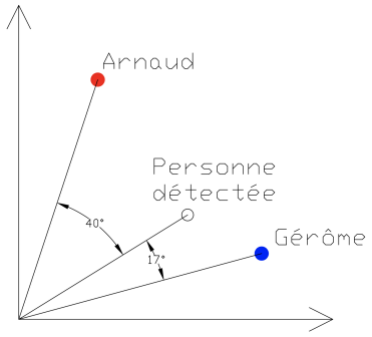


Figure 16 - principe de fonctionnement de la fonction find_closest

Pour cette application, cette méthode de mesure fonctionne correctement tant que le nombre de personnes dans la dataset est faible (une dizaine de personnes), sinon le taux d'imprécision sera plus important et il sera préférable de s'orienter vers une autre approche.

5.2.4. Mise en place du programme principal

Ce programme appelle les différents modes expliqués ci-dessus. À l'aide d'un dashboard réalisé sur NodeRed (expliqué plus loin dans ce rapport), l'utilisateur peut accéder aux différents menus proposés, à savoir :

- La prédiction en live.
- L'ajout d'une personne avec la piCamera dans le système de reconnaissance.
- L'ajout d'une personne avec des photos téléchargées dans la Raspberry (par exemple : dans le dossier /home/pi/Downloads/Arnaud_Josis).
- La suppression d'une personne dans le système de reconnaissance.
- L'arrêt du programme.

Une fois un bouton sur le dashboard pressé par l'utilisateur pour choisir le menu, une valeur est envoyée au topic etu30673@henallux.be/si/option du broker MQTT (maqiatto.com dans ce projet). Comme la Raspberry est abonnée à ce topic, elle reçoit la valeur en provenance du broker MQTT. En fonction de la valeur, le programme principal implémente le menu choisi (0 pour quitter, 1 pour la prédiction en live, 2 pour ajouter une personne avec la piCamera, 3 pour ajouter une personne avec des photos téléchargées et 4 pour supprimer une personne). Le schéma suivant résume le principe expliqué dans ce paragraphe.

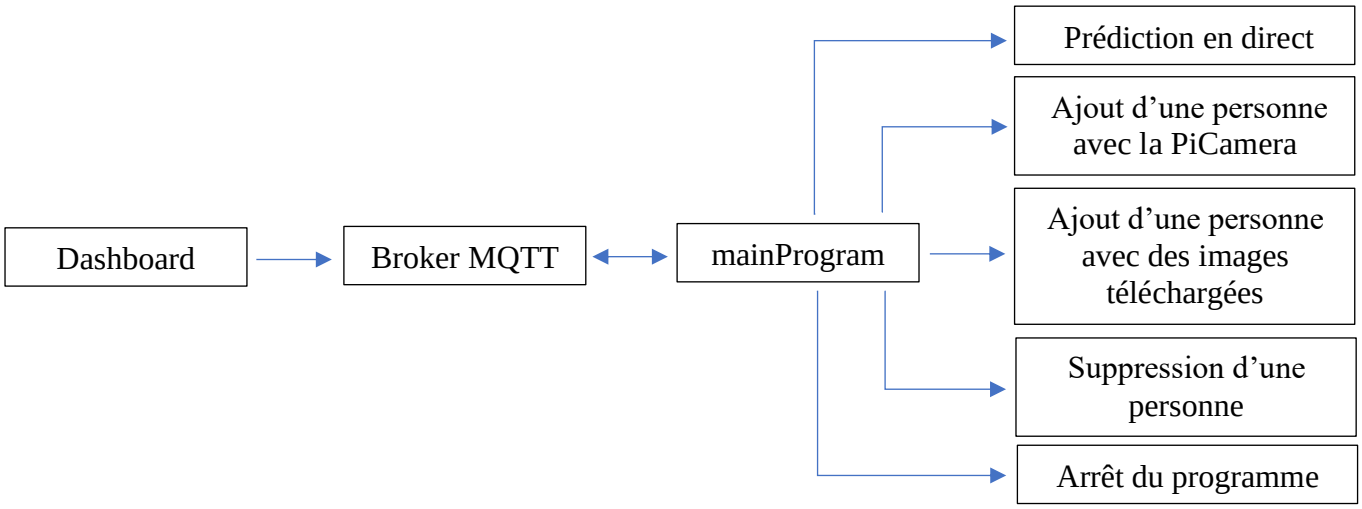


Figure 17 - principe de fonctionnement du mainProgram

Dans le cas où l'utilisateur choisi l'option de prédire en live, une nouvelle fenêtre s'ouvre et le flux de la piCamera est utilisé par l'algorithme HaarCascade pour détecter la présence d'un visage. Une fois détecté, l'image est transmise à la fonction « auto_crop » pour découper le visage à la taille 224x224 pour l'envoyer à la fonction « find_closest » afin de prédire la personne. Si celle-ci est inconnue, la valeur 0 est envoyée au broker MQTT sur le topic etu30673@henallux.be/si/recognition pour informer la partie actionneur. Si cette personne est connue, c'est la valeur 1 qui est envoyée au broker MQTT. L'image de la dernière personne prédite est enregistrée dans le fichier « lastPredictPerson.png ». Pour quitter le mode de prédiction en live, il faut appuyer sur la lettre « q ».

Dans le cas où l'utilisateur choisi d'ajouter une nouvelle personne via la piCamera, il est invité, sur le dashboard, à encoder son prénom (sans accent) et appuyer sur la touche « enter », puis son nom (sans accent) et appuyer sur la touche « enter ». Les chaines de caractères contenant le nom et le prénom de la personne sont envoyées vers le broker MQTT sur les topics etu30673@henallux.be/si/nom et etu30673@henallux.be/si/prenom avant d'être récupérées par la Raspberry. Ensuite, le programme « buildFaceDataset » est appelé, une nouvelle fenêtre s'ouvre et affiche le flux de la piCamera. Lorsque l'utilisateur se place devant la caméra, son visage est encadré avec un rectangle vert. Pour prendre une photo, il doit appuyer sur la touche « k » et pour quitter le mode, il doit appuyer sur la touche « q ». Les images sont enregistrées dans le dossier « dataset/Prénom_Nom » de la personne. Si 20 images sont prises, le programme est automatiquement quitté. Il est conseillé de prendre des photos dans différentes conditions de luminosité et avec différents accessoires (lunettes, lunettes de soleil, chapeau, casquette). La fonction « resizePictures » est appelée pour renommer les nouvelles images. Une fois cette action terminée, la fonction « generate_database » permet de mettre à jour le dictionnaire (dataset.txt) comprenant les 2622 valeurs multipliés par le nombres de photo prises.

Dans le cadre où l'utilisateur choisi d'ajouter une nouvelle personne via des photos téléchargées, il est invité à encoder sur le dashboard le chemin absolu (exemple : /home/pi/Downloads/Arnaud_Josis) où se trouve le dossier avec les photos téléchargées. La chaine de caractère contenant le chemin est envoyée au broker MQTT avant d'être récupérée par la Raspberry. Le nom du dossier contenant les photos doit absolument être sous le format Prénom_Nom (sans accent). Ce dossier est rapatrié dans le dossier « dataset » à l'aide de la fonction « resizePictures », qui redimensionne les images à une valeur maximum en hauteur et en largeur sans les déformer, avant de les renommer. De même que pour le mode précédent, la fonction « generate_database » permet de mettre à jour le dictionnaire (dataset.txt) comprenant les 2622 valeurs multipliés par le nombres de photo prises.

Dans le cas où l'utilisateur choisi de supprimer une personne, il doit mentionner le nom de la personne à supprimer dans le système de reconnaissance faciale (sous le format Prénom_Nom). La chaine de caractère contenant le nom complet de la personne est envoyée au broker MQTT avant d'être récupérée par la Raspberry. Le dossier contenant les photos de la personne à supprimer dans le répertoire « dataset , ainsi que le nom et les 2622 valeurs multipliées par le nombre de photos de cette personne sont supprimés.

6. Actionneurs

Le schéma ci-dessous correspond au montage effectué sur la partie actionneur.

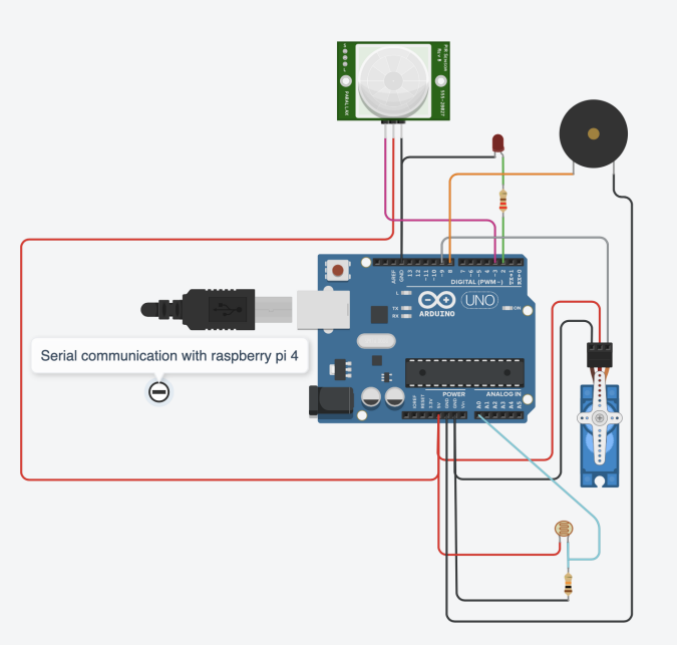


Figure 18 : Schéma de câblage des actionneurs

Comme la Figure 18 le montre, la partie actionneur est constituée d'un Arduino uno qui communique avec un Raspberry Pi 4 via le port série (USB). Celui-ci permet de recueillir les données en provenance de la Raspberry réalisant la reconnaissance. Divers modules sont connectés à cet arduino tels que :

- Un servomoteur
- Un buzzer
- Un capteur de lumière
- Une led
- Un détecteur de mouvement
- Un capteur de luminosité

Les programmes des divers éléments sont présents sur le GitHub de l'équipe.

6.1. Servomoteur

Le programme du servomoteur consiste à ouvrir la porte lorsqu'une personne connue se présente devant la porte. Cette personne sera représentée par « a=1 ».

Figure 19 : Servomoteur utilisé

Dans ce programme il faut :

- Inclure la librairie <servo.h>.
- Déclarer un object «lock ».
- Dans la fonction « setup », activer la communication via le port série (fréquence 9600 bauds) et attacher l'objet « lock » à une sortie digitale. Ici, il s'agit de la pin D9.

- Écrire une fonction « init_servo(void) » qui permet de vérifier que la porte est correctement fermée.
- Écrire une fonction « management_door(void) » qui, selon la valeur de la variable « a », fermera ou ouvrira la porte.

La fonction lock.write(0) ou lock.write(90) permet de donner l'angle de rotation que doit effectuer le servomoteur.

6.2. Avertisseur sonore

Ce programme permet de déclencher un avertisseur sonore si la personne est inconnue. Dans un premier temps, il faut inclure la librairie « pitches.h », qui est une librairie qui définit les différentes notes de mélodies pour l'avertisseur.

Ensuite il faut déclarer un tableau avec les mélodies souhaitées et un second, de la même taille, avec la durée des mélodies.

Figure 20 : Buzzer piézoélectrique utilisé

Finalement dans le void loop(), tant que la personne est inconnue, la mélodie retenti grâce à une boucle « for ».

6.3. Détection de mouvement

Une détection de mouvement a été mise en place afin d'allumer une lampe lorsqu'une personne se présente à la porte et qu'il fait noir. Un capteur de luminosité permet de déterminer si l'intensité lumineuse est inférieure à une certaine valeur (ici, 400 lux) et un capteur de mouvement vérifie la présence d'une personne. Sans ce système, la reconnaissance serait impossible car la caméra n'est pas équipée d'un système de vision nocturne (infrarouge).

Si l'intensité lumineuse mesurée par le capteur de lumière est inférieure à 400 lux, la valeur retournée par le capteur de mouvement est prise en compte. Si ce capteur détecte un mouvement, la LED connectée sur la PIN2 est allumée durant 3 secondes.

Figure 22 : Capteur de luminosité utilisé (photorésistance)

Figure 21 : Capteur PIR utilisé

6.5. Association des programmes

Après avoir programmé les différents actionneurs séparément, ceux-ci ont été associés dans un programme général. Voici une représentation graphique représentant les différentes fonctions et boucles dans le programme :

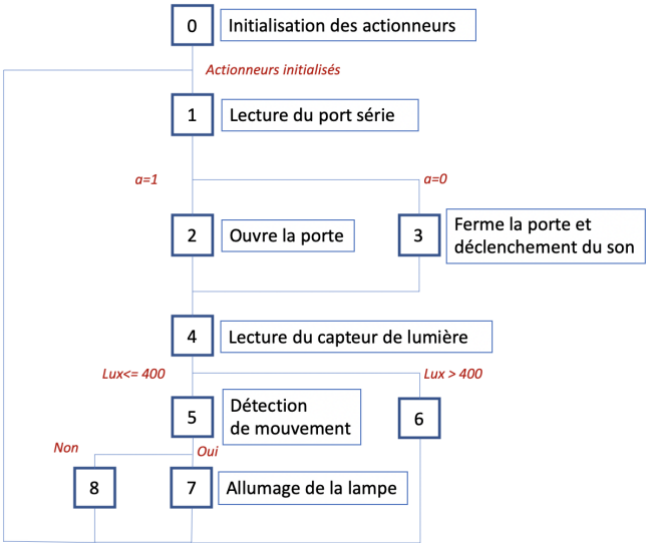
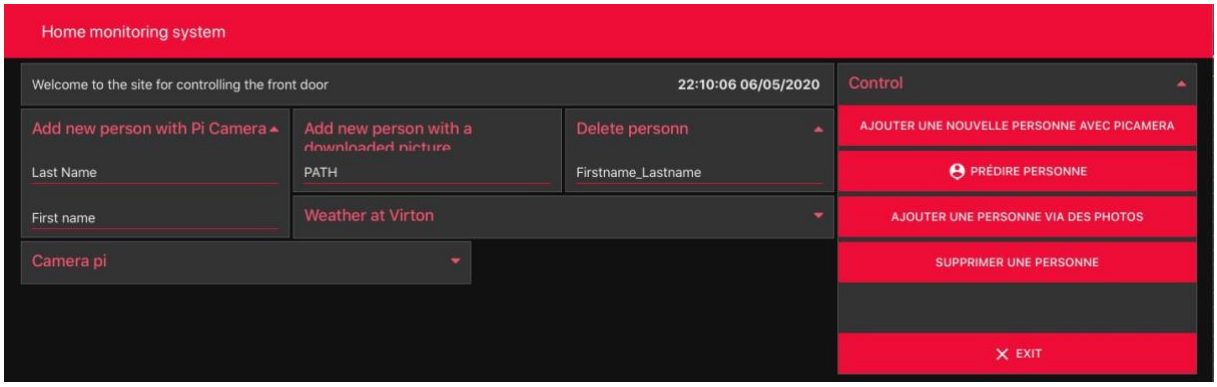


Figure 23 : Algorithme du programme de gestion des actionneurs

7. Dashboard NodeRed

Suite aux circonstances du coronavirus, une communication MQTT et un dashboard sur NodeRed ont été mis en place, afin de faciliter le transfert des informations entre l'algorithme de reconnaissance qui était installé sur une Raspberry Pi 4, située à Namur, et le programme de gestion des actionneurs sur un Arduino qui communiquait avec un Paspberry Pi 4 via le port série, située à Libramont.
Le dashboard permet également d'activer, à l'aide de boutons, des options liées au programme de reconnaissance faciale :

- Ajouter une personne via la piCamera.
- Ajouter une personne via des photos stockées sur la Raspberry.
- Supprimer une personne de la base de donnée.
- Réaliser la prédiction en live.
- Quitter le programme.



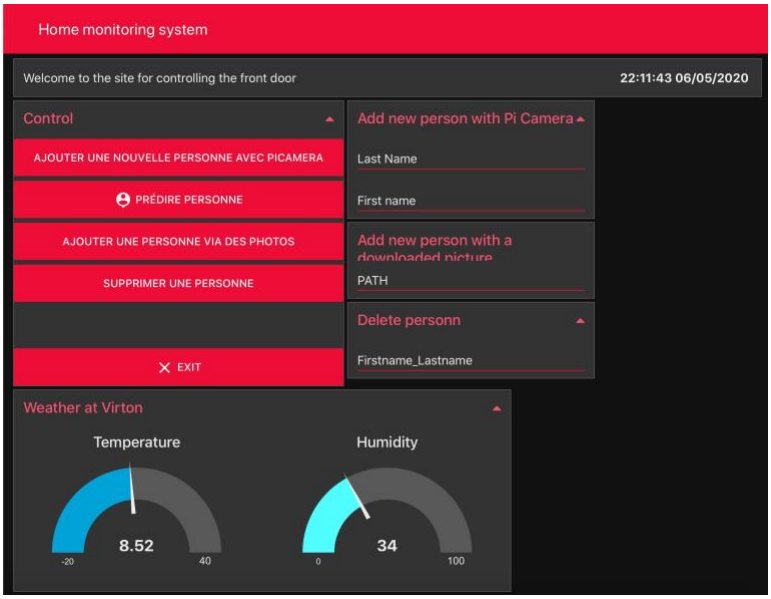


Figure 24 : Dashboard

Sur le dashboard, le flux caméra est affiché à l'aide d'un flow constitué de 5 nodes. L'affichage du flux correspond en réalité à l'affichage d'une photo prise par la PiCamera toutes les 2 secondes.

Figure 25 : Flow NodeRed, acquisition flux caméra

Afin d'agrémenter l'affichage, des données météos de la ville de Virton, en Belgique, ont été ajoutées. Il s'agit de « requêtes http » sur le site openweathermap.

Figure 26 : Flow NodeRed, requête http

Enfin, divers boutons de contrôle sont implémentés sur le panel, afin de pouvoir envoyer des informations à la Raspberry qui gère la reconnaissance faciale.

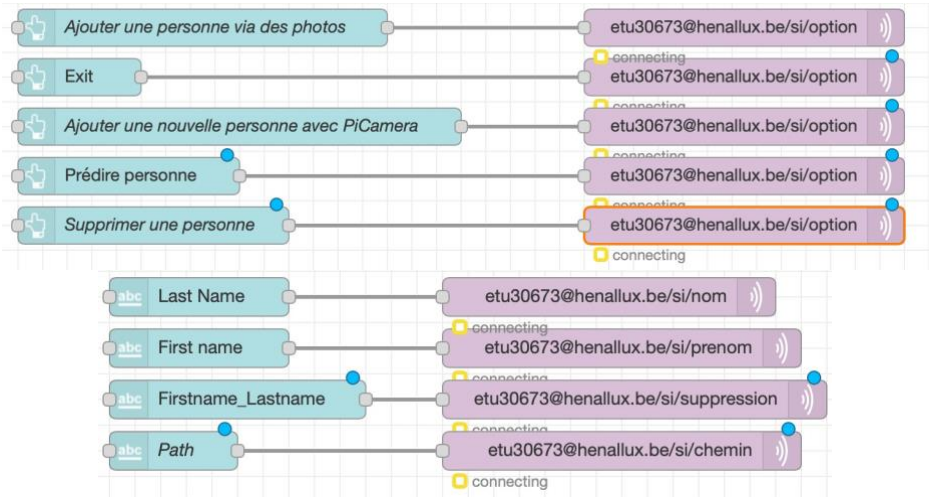


Figure 27 : Flow NodeRed pour l'implémentation des boutons et l'envoie sur les topics mqtt

Pour mettre en place ce dashboard, il était indispensable d'installer les palettes suivantes :

1. ***node-red-contrib-camerapi***
2. ***node-red-contrib-moment***
3. ***node-red-dashboard***
4. ***node-red-node-base64 (pour l'encodage des images)***
5. ***node-red-node-serialport***
6. ***node-red-contrib-ui-media***

8. Communication MQTT

Figure 28 - logo MQTT

Suite aux circonstances du coronavirus, une communication MQTT a été mise en place, afin de faire communiquer la partie reconnaissance et la partie actionneur. Le protocole de communication établi est le suivant :

Émetteur	Récepteur	Donnée envoyée	Signification
Raspberry (Reconnaissance)	Raspberry+Arduino (Actionneurs)	0	Personne détectée inconnue
Raspberry (Reconnaissance)	Raspberry+Arduino (Actionneurs)	1	Personne détectée connue
Raspberry+Arduino (Actionneurs)		2	Aucune présence

Le broker maqiatto.com configuré possède 6 topics :

- **etu30673@henallux.be/si/recognition** : Ce topic permet à l'Arduino de lire la sortie du programme de reconnaissance. Les données envoyées ainsi que leur signification sont présentes dans le tableau ci-dessus.

- **etu30673@henallux.be/si/option** : Ce topic permet d'envoyer des consignes au Raspberry (reconnaissance) afin de choisir un mode (Quitter le programme, ajouter une personne via la piCamera, ajouter une personne via des photos téléchargées, supprimer une personne dans le système ou prédiction en live).

- **etu30673@henallux.be/si/prenom:** Ce topic permet d'envoyer le prénom de la personne à ajouter à la base de données.

- **etu30673@henallux.be/si/nom:** Ce topic permet d'envoyer le nom de la personne à ajouter à la base de données.

- **etu30673@henallux.be/si/suppression:** Ce topic permet d'envoyer le prénom et le nom sous le format « Firstname_Lastname » de la personne à retirer de la base de données.

- **etu30673@henallux.be/si/chemin:** Ce topic permet d'ajouter des photos d'une personne dans la base de données lorsque les photos sont téléchargées dans la Raspberry.

9. Lien Github

La totalité des programmes (Raspberry et Arduino) de ce travail se retrouve sur Github. Dans le fichier « readme.md », il est possible de retrouver tout le protocole à suivre pour mettre en place le projet ainsi qu'une partie « mode d'emploi ». Tous les liens de références se trouvent également dans le « readme.md ».

10. Améliorations

Ce point aborde différentes améliorations recensées tout au long de la réalisation du projet afin de perfectionner le système reconnaissance faciale proposé par ce travail.

- Il aurait été intéressant de disposer du modèle Raspberry Pi 4 avec la plus grande capacité d'un point de vue de la ram, c'est-à-dire le modèle 4Go. En effet, lors des essais, il fallait près de 10 minutes pour charger le modèle et 45 secondes pour prédire la personne avec la Raspberry Pi 4 1Go de ram (celle fournie par l'école). Comme nous disposions d'une Raspberry Pi 4 4Go de ram, nous avons réalisé un test avec celle-ci. La durée du travail était beaucoup moins longue. Le chargement du modèle a été réalisé en 30 secondes et la prédiction a pris environs 5 secondes, ce qui est déjà beaucoup plus acceptable d'un point de vue temps de prédiction pour un projet de reconnaissance faciale. Le prix de la Raspberry Pi 4 avec 4Go de ram est à 69€[5] TVAC au lieu de 49,9€ [6]TVAC pour la même version en 1Go de ram, soit une différence de 19,1€.

- Une autre amélioration à réaliser est de calculer la précision fournie par le modèle sur une base de données avec un grand nombre d'images. Actuellement, la base de donnée est composée de maximum 20 photos par personnes, prises dans différentes conditions (personne portant un bonnet, une écharpe ou encore un masque).

- L'algorithme proposé dans ce travail ne permet pas prédire le nom de plusieurs personnes sur le flux vidéo de la piCamera. En modifiant la fonction « auto_crop » et le programme principal, il serait possible d'implémenter cette modification.

- L'algorithme HaarCascade utilisé pour détecter le visage des personnes commet des erreurs comme il est possible de le voir sur les images suivantes :

5 https://www.gotronic.fr/art-carte-raspberry-pi-4-b-4-gb-30209.htm
6 https://www.gotronic.fr/art-carte-raspberry-pi-4-b-1-gb-30752.htm

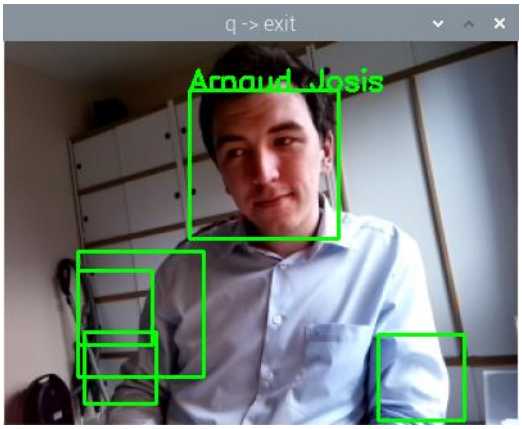


Figure 29 - Erreur avec l'algorithme HaarCascade

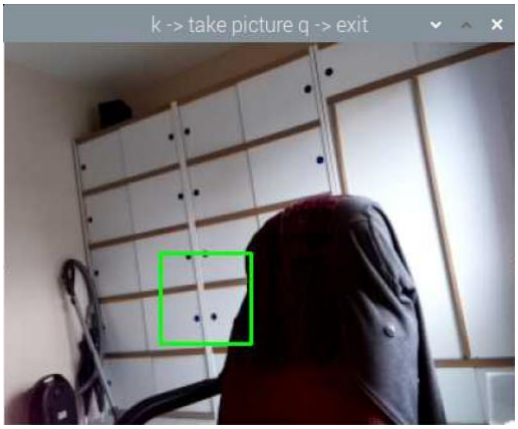


Figure 30 - Algorithme HaarCascade sans personne sur l'image

Deux propositions peuvent être énoncées :
La première proposition est d'ajuster les paramètres de l'algorithme haar cascade pour éviter la détection d'éléments indésirables.
La deuxième proposition serait d'utiliser la librairie « dlib » avec le shape predictor, qui cherche sur l'image les 68 points entourant le visage comme il est possible de le visualiser sur l'image suivante. Cet algorithme est plus précis que l'algorithme HaarCascade.

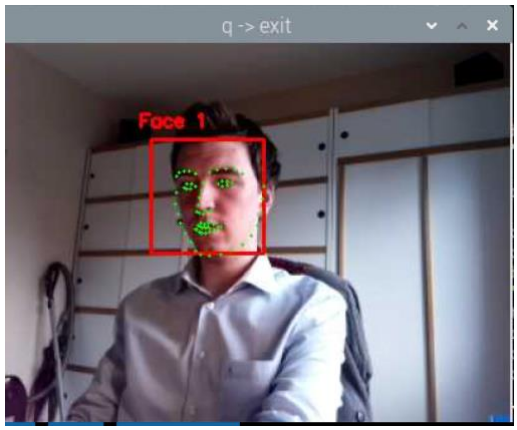


Figure 31 - Algorithme predict shape (librairie dlib)

11. Conclusion

L'idée du projet était de mettre en place un système de reconnaissance faciale, utilisé pour contrôler l'accès des personnes à leur résidence. Après avoir exploré la librairie FaceRecognition, le modèle VGG a été adopté pour reconnaitre au mieux la personne. Cet algorithme de reconnaissance faciale est capable de reconnaitre dans des conditions aléatoires (comme le port de lunettes, de bonnet, d'écharpe ou encore d'un masque) la personne voulant entrer dans la maison.

Les recherches effectuées pour réaliser ce projet nous ont permis d'approfondir nos connaissances sur le deep-learning et plus précisément sur le modèle VGG. Nous avons été capables d'implémenter au sein de ce projet des notions vues dans le cours de réseau, enseigné par Monsieur Berton, pour faire face aux mesures prises dûes au coronavirus. Ces notions sont : la communication MQTT ainsi que l'utilisation de NodeRed.

Pour finir, la reconnaissance faciale est un domaine en plein essor. Cette technologie est actuellement utilisée dans certaines entreprises, dans certaines grandes villes, ou encore dans les derniers smartphones. Elle pourrait même, pourquoi pas, aider dans la lutte contre le Coronavirus !

12. Bibliographie

GANJI M., *Installation of Opencv, numpy, scipy inside a virtualenv*, 16 décembre 2016 (page consultée en avril 2020), < https://medium.com/@manuganji/installation-of-opencv-numpy-scipy-inside-a-virtualenv-bf4d82220313 >

LAMBERT R., *TP : La Reconnaissance Faciale*, 14 mai 2019 (page consultée en avril 2020), < http://penseeartificielle.fr/tp-reconnaissance-faciale/ >

LETMEKNOW, *Utiliser un buzzer*, 10 novembre 2013 (page consultée en avril 2020), < https://letmeknow.fr/shop/fr/blog/53-tuto-utiliser-un-buzzer >

MAQUIATTO, *MaQiaTTo Broker Live*, 2018 (page consultée en mai 2020), < https://www.maqiatto.com >

NAGASHUR, *Capteur PIR et Arduino : Tutoriel*, 14 aout 2015 (page consultée en avril 2020), < http://nagashur.com/blog/2015/08/14/capteur-pir-et-arduino-tutoriel/ >

NODERED, *Show Image as icon on dashboard*, 2020 (page consultée en mai 2020), < https://discourse.nodered.org/t/show-image-as-icon-on-dashboard/806 >

NODERED, *node-red-contib-camerapi*, 2020 (page consultée en mai 2020), < https://flows.nodered.org/node/node-red-contrib-camerapi >

NODERED, *node-red-node-base64*, 2020 (page consultée en mai 2020), < https://flows.nodered.org/node/node-red-node-base64 >

NODERED, *nod-red-contib-image-output*, 2020 (page consultée en mai 2020) < https://flows.nodered.org/node/node-red-contrib-image-output >

OPENCV, *Site web d'OpenCV*, 2020 (page consultée en avril 2020), < https://opencv.org >

PARKHI M. O., *Deep Face Recognition*, 2015 (page consultée en avril 2020) < http://www.robots.ox.ac.uk/~vgg/publications/2015/Parkhi15/parkhi15.pdf >

PENSEEARTIFICIELLE, *VGG Face*, 14 mai 2019 (page consultée en avril 2020), < http://penseeartificielle.fr/wp-content/uploads/2019/05/slides-conférence-La-Reconnaissance-Faciale-5.jpg >

PROJETDIY, *Débuter avec le broker MQTT Mosquitto sur Raspberry Pi, Windows, macOS et Linux*, 24 avril 2020 (page consultée en avril 2020), < https://projetsdiy.fr/mosquitto-broker-mqtt-raspberry-pi/ >

PYTHONDOCTOR, *Interface graphique Tkinter python*, 2020 (page consultée en avril 2020), < https://python.doctor/page-tkinter-interface-graphique-python-tutoriel >

QUORA, *What is max pooling in convolutional neural networks*, 2020 (page consultée en avril 2020), < https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks >

RASPBERRYVALLEY, *Configure Default Python version on your Pi*, sans date (page consultée en avril 2020), < https://raspberry-valley.azurewebsites.net/Python-Default-Version/ >

ROSEBROCK A., *Face recognition with OpenCv, Python, and deep learning*, 18 juin 2018 (page consulée en avril 2020), < https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/ >

ROSEBROCK A., *How to build a custom face recognition dataset*, 11 juin 2018 (page consultée en avril 2020), < https://www.pyimagesearch.com/2018/06/11/how-to-build-a-custom-face-recognition-dataset/ >

ROSEBROCK A., *Install OpenCV 4 on Raspberry Pi 4*, 16 septembre 2019 (page consultée en avril 2020), < https://www.pyimagesearch.com/2019/09/16/install-opencv-4-on-raspberry-pi-4-and-raspbian-buster/ >

ROSEBROCK A., *Python, argparse, and command line arguments*, 12 mars 2018 (page consultée en avril 2020), < https://www.pyimagesearch.com/2018/03/12/python-argparse-command-line-arguments/ >

RYAN P., *Facial Recognition – a visual step by step*, 8 août 2019 (page consultée en avril 2020), < https://medium.com/swlh/facial-recognition-a-visual-step-by-step-d679289bab11 >

SEFIKS, *Deep Face Recognition with Keras*, 6 aout 2018 (page consultée en avril 2020), < https://sefiks.com/2018/08/06/deep-face-recognition-with-keras/ >

SKYWODD, *Contrôler un servomoteur avec une carte Arduino / Genuino*, 4 mai 2016 (page consultée en avril 2020), < https://www.carnetdumaker.net/articles/controler-un-servomoteur-avec-une-carte-arduino-genuino/ >

STACKEXCHANGE, *Update Python version on Raspbian*, 2016 (page consultée en avril 2020), < https://raspberrypi.stackexchange.com/questions/26286/update-python-version-on-raspbian >

STACKOVERFLOW, *Error : Environment /Users/myuser/.virtualenvs/iron does not contain activation script*, avril 2020 (page consultée en avril 2020), < https://stackoverflow.com/questions/60252119/error-environment-users-myuser-virtualenvs-iron-does-not-contain-activation-s/60292344#60292344 >

WEI J., *VGG Neural Networks : The Next Step After AlexNet*, 3 juillet 2019 (page consultée en avril 2020), < https://towardsdatascience.com/vgg-neural-networks-the-next-step-after-alexnet-3f91fa9ffe2c >

WIKIPEDIA, *Histogram of oriented gradients*, 14 janvier 2020 (page consultée en avril 2020), < https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients >

ZARADZKI M., *Face recognition with Keras and OpenCV*, 6 mars 2017 (page consultée en avril 2020), < https://aboveintelligent.com/face-recognition-with-keras-and-opencv-2baf2a83b799 >