

I RAPPORT DE STAGE		II M1
<u>ELEVE</u>		
Nom : BOUDISSA Prénom : Nourredine Filière : Imagerie et Réalité Virtuelle		
<u>SUJET</u> Reconnaissance de visage et suivi de personne		
<u>ENTREPRISE</u>		
Nom : Laboratoire EndiCap – Université de Versailles Saint-Quentin-en-Yvelines		
Adresse : 55 avenue de Paris, 78035 Versailles		
Adresse du lieu du stage si différent : 2 Avenue de la Source de la Bièvre, 78180 Montigny-le-Bretonneux		
<u>DATE DU STAGE</u>		
du : 6 Juillet 2020 au : 25 Septembre 2020		
durée effective en semaines : 12		
Date de remise du rapport aux membres du Jury :		
<u>SOUTENANCE</u>	Date :	Heure :
Composition du Jury :		
- Président (responsable EFREI ou ESIGETEL) :		
- Responsable du stage (Entreprise) :		
- Invité(e) :		
<u>PUBLICATION DU RAPPORT DE STAGE</u>		
Le Responsable du stage : autorise le stagiaire à publier le rapport de stage sur l’Intranet de l’Ecole. Signature		

Sommaire :

Remerciement.....	3
1.Introduction.....	4
2.Présentation détaillé de la mission.....	5
3.Méthode de travail.....	7
4.Travail effectué.....	8
4.1.Détection de personne grâce au visage.....	8
4.2.Suppression de faux positifs.....	10
4.3.Reconnaissance de visage.....	13
4.4.Fiabilité d'un programme.....	16
4.5.Tracking vidéo.....	18
4.6.Travailler sous Linux.....	21
4.7.Optimisation de code et comprendre l'embarquée.....	26
4.8.L'autonomie en entreprise.....	28
4.9.La communication.....	30
4.10.Documenter son travail.....	32
5.Conclusion.....	33
6.Annexe.....	35

Remerciements :

Avant de commencer à introduire mon stage, je tiens à remercier M. BONNIN Patrick, qui m'a permis d'effectuer ce stage en me le proposant. Malgré les conditions sanitaires peu évidentes, le stage a pu s'effectuer de façon bénéfique pour moi, et ce, grâce aussi à mon tuteur de stage, M. BLAZEVIC Pierre, qui m'a accepté en tant que stagiaire et m'a fait confiance pour cette mission.

J'ai pu apprendre beaucoup de chose et évoluer grâce à ces 2 personnes. J'ai été accueilli agréablement, et pu découvrir le monde professionnel à travers ce stage.

1.Introduction

C'est la première fois, que j'effectue un stage dans mon domaine, et pour une durée de plus d'un mois. En effet, mon stage était centré sur ce que portent mes études, c'est-à-dire l'imagerie. Le stage a duré 12 semaines et a été réalisé en grande partie en télétravail dû aux conditions particulières de ces derniers mois.

J'ai eu l'opportunité de réaliser mon stage avec l'ISTY en partenariat avec le laboratoire EndiCap. Le but du stage était d'effectuer de la recherche sur les techniques de détection, reconnaissance, et suivi de personne, à appliquer dans un hôpital ou dans les EPHAD par exemple afin de suivre des personnes âgées et détecter leur déplacement. On peut alors savoir si ces personnes ne s'égarèrent pas (dans le cas où elles auraient la maladie d'Alzheimer).

J'ai décidé d'effectuer ce stage pour des raisons évidentes qui sont que la mission me plaisait, et est en lien avec mes études et mes envies. D'autant plus que le but était de faire beaucoup de recherche, je savais que j'allais apprendre beaucoup et évoluer très rapidement que ce soit en programmation, ou au niveau de mes connaissances sur la détection/reconnaissance de personne.

Ma mission était donc de développer un programme de détection, et de reconnaissance de personne, via une caméra connectée au réseau. Le programme devait fonctionner sous Raspberry Pi 4, donc sous le système d'exploitation Debian, que je n'avais jamais utilisé auparavant.

J'ai pu donc mieux comprendre les aspects du métier, et du travail en entreprise durant ces 12 semaines.

2.Présentation détaillé de la mission

Ma mission se situait surtout dans la recherche de technique de détection et reconnaissance de personne grâce à une caméra IP (connecté à internet). Pour ce faire, je devais développer des algorithmes en C++ ou en Python, les tester, et documenter dessus.

J'avais à ma disposition une caméra IP, une Raspberry 3, ainsi que mon propre PC pour y parvenir :



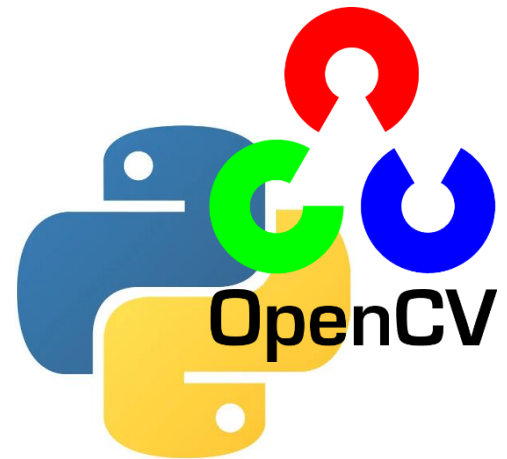
On utilise ici une caméra IP car il est possible de récupérer son flux vidéo via internet, car elle a une IP à elle. L'avantage est donc qu'il est possible de récupérer plusieurs flux vidéo, avec plusieurs caméras, qui auront chacune une IP, afin de travailler sous plusieurs angles, ou dans plusieurs pièces.

La Raspberry quant à elle, permet de rendre le travail embarqué, et donc de l'utiliser n'importe où et avec n'importe quel caméra (webcam, caméra IP, ect...). Il fallait donc développer la solution pour qu'elle fonctionne dessus. Le système d'exploitation déployé dessus est le Raspbian, qui est basé sur Debian.

J'ai donc travaillé sur Debian pour développer mon programme, pour ensuite le déplacer sur Raspberry car je n'en avais pas accès les premières semaines.

Afin d'effectuer mes programmes, j'ai eu besoin d'utiliser une bibliothèque très connue dans le domaine du traitement d'image, qui se nomme OpenCV.

OpenCV est une bibliothèque graphique libre avec laquelle il est possible d'utiliser des algorithmes déjà implantés (K-means, transformée de Hough, arbres de Décisions, réseau de neurones artificiels, méthode de Viola & Jones...)



Je devais suivre des consignes dans un certain ordre afin d'avance dans ma mission :

- Comprendre les techniques de détection et de suivi de personne
- Tester l'existant sous OpenCV4 (dernière version d'OpenCV)
- Comparer les différentes techniques pour chaque fonctionnalité implantée

Les trois techniques primordiales à implémenter étaient :

- La détection de personne
- La reconnaissance de personne
- Le suivi de personne

Tout devait se retrouver dans un seul répertoire, pour rendre le travail facilement transportable et installable.

Enfin, une étude bibliographique, avec des guides d'utilisation de chaque outil / technique utilisé était attendue.

3.Méthode de travail

Du fait des règles sanitaires particulières liés à la pandémie COVID-19, les méthodes de travail ont dû changer pour s'adapter. J'ai eu la chance de pouvoir effectuer mon stage, même si c'était à distance en restant chez moi pour la plupart du temps, je ne me rendais qu'à mon lieu de travail pour des réunions sur place (surtout en fin de stage), ou pour récupérer du matériel.

La plupart des autres réunions se faisaient via le logiciel Zoom.

Cela ne m'a pas vraiment handicapé puisque j'avais le matériel chez moi pour travailler, et je pouvais à tout moment au besoin, contacter mon tuteur de stage si un problème ou une question se présentait.

Les réunions se faisaient au rythme d'une à deux par semaine en général. Elles avaient pour but de faire un point et de montrer mon avancement.

Vers la fin du stage, (3 à 4 dernières semaines), je faisais cette fois-ci une réunion en physique par semaine, où je devais présenter mon travail, au représentant de EndiCap,

Le but étant à la fin de rendre la Raspberry fonctionnelle, les démonstrations se faisaient donc avec.

Journée de travail typique :

Je commence par lire ma feuille de route et regarde où j'en suis. Etant donné que chaque étape était détaillée, je pouvais savoir quoi faire ensuite sans forcément contacter mon tuteur.

Je continue d'avancer sur ce que j'étais précédemment, si je rencontre des problèmes, ou que je n'avance pas, je passe à la suite, pour ne pas perdre de temps et laisse ça de côté.

Je pose ensuite des questions sur les choses qui m'ont bloqué et je cherche une solution (changer de manière de faire, continuer car je m'y prenais mal, abandonner l'idée par manque de temps...). J'ai dû parfois en effet laisser certaines choses de côté car elle n'était pas primordiale et je n'y arrivais pas, ou car je n'avais plus le temps de le faire.

Je me concentrais donc surtout sur ce qui était nécessaire de faire.

Si j'obtiens un résultat concluant, alors je fais des tests dessus avec plusieurs valeurs (Partie 5).

A la fin de la journée, j'arrête le développement et rempli mes guides d'utilisation, tutoriels, et mon rapport de suivi journalier afin de compléter la documentation autour de mon travail. J'enregistre aussi tous les sites qui m'ont servi dans un document afin de préparer une bibliographie.

4.Travail effectué

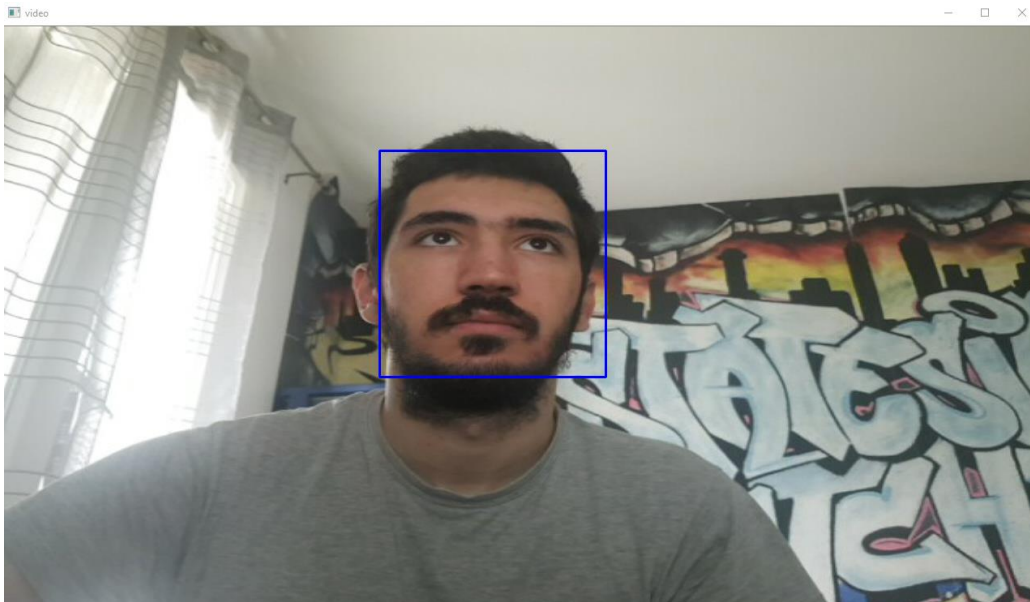
4.1 Détection de personne grâce au visage

La première étape lorsqu'on souhaite reconnaître une personne, c'est de la détecter.

Il y a plusieurs méthodes pour détecter une personne, mais la principale qui a été utilisée est la méthode dite de « Viola & Jones ».

On utilise un fichier qui a été entraîné avec plusieurs visages de face, qu'on appelle classifieur. Ce fichier contient en fait des données sur les visages, et on va rechercher sur notre flux vidéo une zone qui contient des données similaires à ceux contenus dans le fichier. Si les données sont similaires, alors on considère qu'un visage est détecté.

Il ne reste donc plus qu'à dessiner un carré autour de ce visage, pour pouvoir affirmer qu'on l'a bien détecté :



Mon visage détecté via mon algorithme en utilisant la méthode de Viola & Jones

On détecte alors une personne en détectant son visage. Le problème que j'ai rencontré avec la technique de Viola & Jones, c'est qu'elle détectait relativement bien les visages, mais détectait parfois aussi des faux positifs.

Pour reprendre l'exemple ci-dessus, elle détectait parfois, sur le mur derrière moi, des visages, là où il y avait des écritures.

En effet, il se peut que certaines portions du paysage contiennent des caractéristiques qu'on peut assimiler mathématiquement parlant à un visage.

Il fallait donc trouver une méthode pour supprimer les faux positifs. Donc, un moyen de différencier les visages du reste.

4.2 Suppression des faux positifs

Afin de supprimer les faux positifs, la technique qui a été retenue, était de détecter la peau. En effet, la couleur de peau permettait de différencier un élément de décor d'un visage.

Ainsi, en appliquant certains filtres à notre flux vidéo, il est possible d'en extraire la peau :

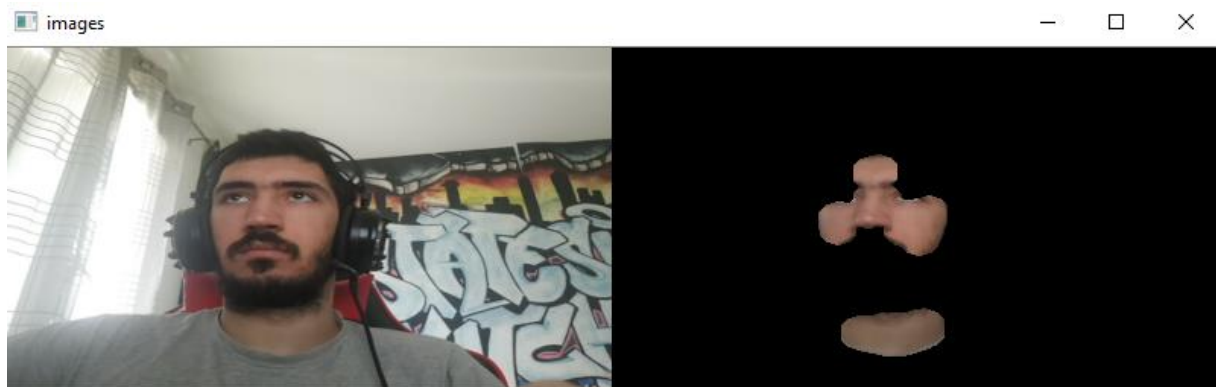


Image Source

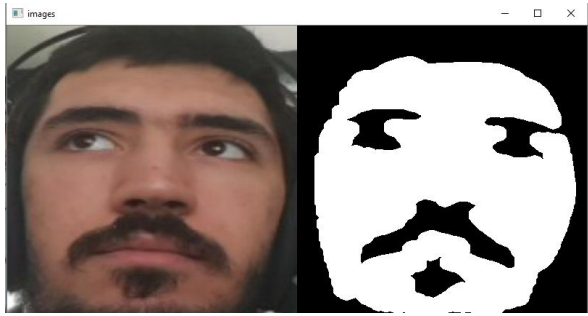
Peau détectée

Il suffit donc d'effectuer la détection de peau, dans l'unique portion de visage détecté (le carré bleu dans l'image en 4.1), puis de juger de façon subjective, à partir de combien de pixel de couleur, on considère qu'on a affaire à un visage.

Pour en tirer des données, il fallait donc transformer l'image en image binaire (image en noir et blanc) avec ce qu'on appelle un seuillage binaire (les pixels sont soit noirs, soit blanc).



Enfin, on applique la détection de peau uniquement sur la petite portion, le visage :



On compte à l'intérieur le nombre de pixel blanc.

On estime qu'à partir d'un certain pourcentage, c'est un visage.

Le pourcentage a été défini de façon arbitraire. Des tests ont dû être effectués pour savoir à partir de quel pourcentage on peut considérer que la détection de visage est fiable.

On obtient alors une détection de visage complète, avec un nombre de positif réduit voire inexistant. Il faut cependant faire attention de ne pas pousser le pourcentage trop haut car cela pourrait avoir l'effet inverse, et donc ne pas détecter un visage, on aurait alors un faux négatif, ce qui est bien plus grave que d'avoir un faux positif dans notre cas.

J'ai donc réellement compris l'importance de faire des tests, pour connaître les limites de ses programmes. C'est un point non négligeable. Si l'on livre un programme sans savoir sur quelle plage de valeur il fonctionne, où jusqu'à quel point il est fiable, alors le programme ne sert à rien. D'autant plus dans mon sujet où la détection et la reconnaissance de visage se fait finalement beaucoup avec des valeurs et des variables subjectives.

C'est-à-dire que les seuils, les limites, et les variables, doivent être choisies au préalable. La première chose que j'ai finalement appris c'est qu'il faut faire des choix, il n'y a aucun problème à ça, mais il faut juste savoir les justifier.

Un des premiers problèmes que j'ai rencontrés avec mon algorithme de détection de peau, c'est qu'il ne reconnaissait même plus les visages car j'avais mis la barre trop haute au niveau de la détection de peau.

L'algorithme considérait par exemple que 60% de pixel blanc sur 100% était trop peu pour conclure qu'on a affaire à un visage.

Il fallait donc que je fasse un bon nombre de tests pour savoir à partir de quel pourcentage je reconnaissais à tous les coups un visage.

Même après cela, je me suis rendu compte que le nombre de tests faits n'était pas assez élevé. Ce qui me semblait beaucoup ne l'était pas finalement. Je n'avais pas assez varié les tests. Je les ai faits dans un premier temps uniquement avec mon visage dans plusieurs positions alors que je me suis rendu rapidement compte d'un autre problème lorsque j'ai commencé à utiliser d'autres visages.

En effet, la couleur de peau joue une importance cruciale, si la peau est trop clair, ou trop foncé, alors elle n'est pas reconnue comme étant de la peau, conformément à la plage de couleur que j'ai défini comme étant une plage de couleur de peau.

Après plusieurs tests, j'ai pu donc trouver des valeurs intéressantes qui rendait l'algorithme beaucoup plus souple.

C'est un problème que j'ai encore rencontré pour le deuxième algorithme primordial à faire, l'algorithme de reconnaissance de visage.

4.3 Reconnaissance de visage

La reconnaissance d'un visage passe par 2 étapes distinctes :

-L'entraînement d'un set de visages pour obtenir des données permettant de le reconnaître (on enregistre en fait des vecteurs dans un fichier)

-La détection d'un visage en temps réel puis la mise en correspondance de ce visage avec le bon enregistré.

Pour reconnaître un visage, il est donc nécessaire d'abord de l'avoir détecté. Une fois détecté, on le compare avec les visages que l'on possède en base de donnée, pour savoir s'il y a correspondance. Cela nécessite donc de créer une base de donnée des visages que l'on souhaite reconnaître.

Le premier programme que j'ai créé est donc un programme qui va détecter les visages qui apparaissent à la caméra, leur attribuer un identifiant, puis les enregistrer en image dans un dossier qui portera le nom de leur identifiant.

On enregistre un même visage, plusieurs fois, dans différentes positions, afin d'en capturer le plus d'information possible. Les images sont enregistrées en noir et blanc car la couleur ne joue aucun rôle sur la reconnaissance de visage que j'ai utilisé.

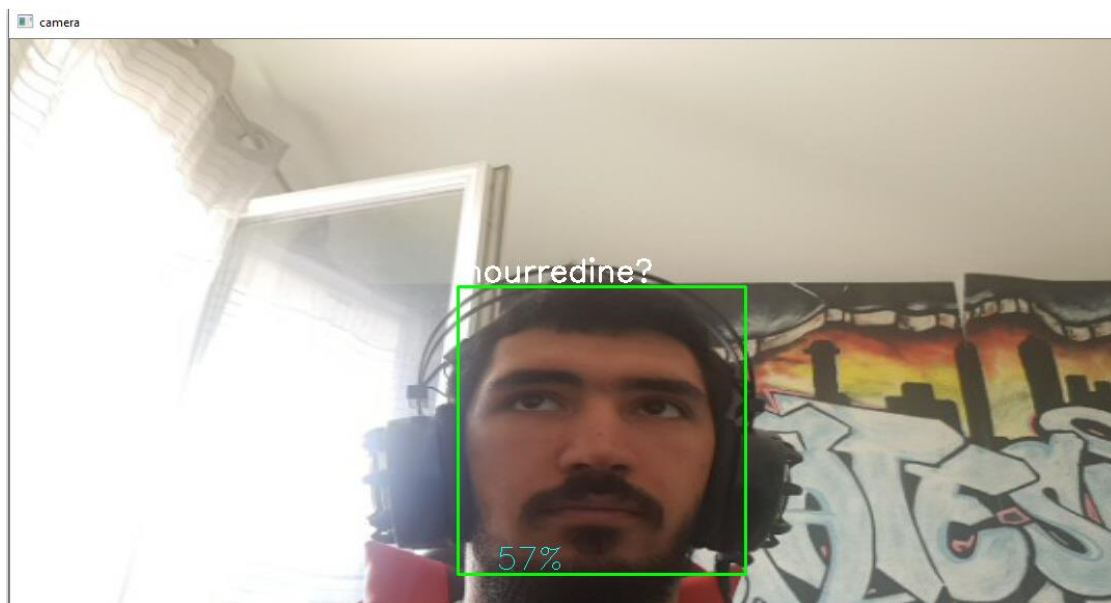
Plus un visage est pris en photo dans des angles différents et en grand nombre, plus il y aura des informations qui seront enregistrées et donc plus il sera simple de le reconnaître.



Une fois les photos prises, un classifieur est créé avec, afin de reconnaître cette personne en particulier. (Des vecteurs sont enregistrés dans ce classifieur, qui sont des caractéristiques du visage enregistrés).

Il reste alors à lancer la comparaison entre le visage courant (celui affiché à la caméra) , et tous ceux qu'on a enregistré.

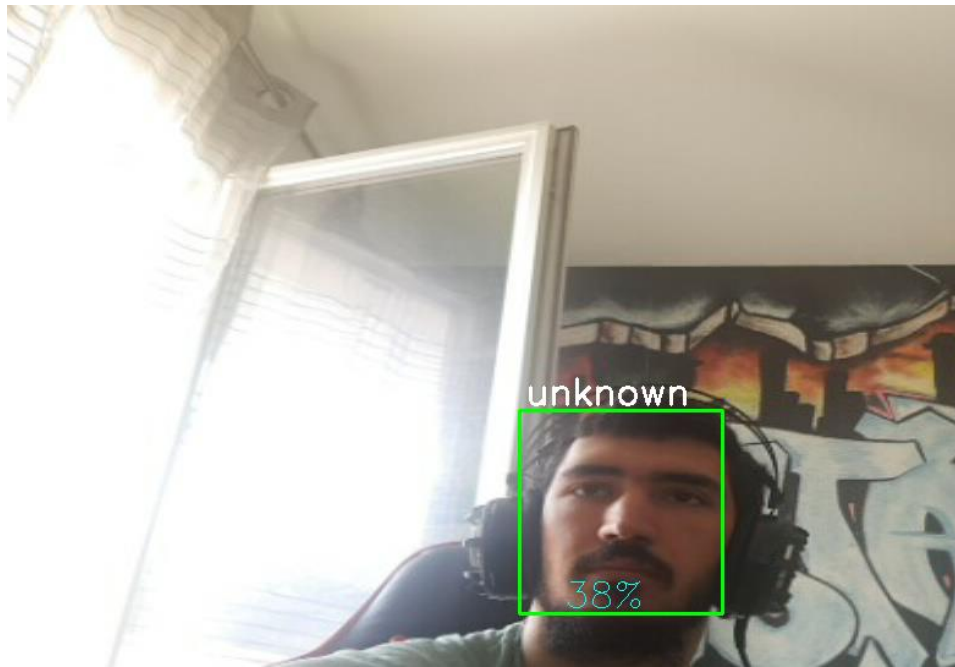
Si un des visages possèdent des caractéristiques similaires au visage courant, alors on les met en correspondance, il suffit de lire l'identifiant du visage enregistré pour connaître le nom de la personne et l'afficher :



Ici, le prénom *nourredine* est affiché au-dessus du cadre. Cela signifie que l'algorithme a reconnu ce visage comme étant *nourredine* de la base de donnée.

Il y a un taux de ressemblance qui s'apparente à 57%.

Dans certaines positions, le visage n'est pas reconnu (manque de prise) :



L'un des premiers problèmes que j'ai rencontrés est sur ce pourcentage. Je cherchais à partir de quel pourcentage on pouvait estimer que le visage reconnu est le bon.

Dans la deuxième photo affichée, malgré que « Unknown » soit affiché (inconnu), un pourcentage est écrit. J'ai en fait estimé qu'on associe le visage à quelqu'un à partir de 40%. L'algorithme fait donc bien correspondre au-dessus le visage à quelqu'un de la base de donnée, mais le taux de ressemblance n'est pas assez élevé pour oser affirmer que c'est bien cette personne, c'est pourquoi il affiche « Unknown ».

Le pourcentage de ressemblance diffère selon chaque visage. Certains visages sont plus simples à enregistrer que d'autres. J'ai pris du temps à me rendre compte que je n'enregistrais pas assez de photo d'un même visage, et pas assez dans des positions et situations différentes.

En effet, toutes mes prises étaient en général à forte luminosité, ce qui rendait les visages quasiment non reconnaissables à basse luminosité.

C'est donc ainsi que j'ai appris l'importance de comprendre, avant de programmer, le résultat que l'on souhaite obtenir.

J'ai dû remanier le programme afin de d'enregistrer les visages différemment. J'ai donc finalement perdu beaucoup de temps. Si j'avais pensé à ça avant, j'aurais pu directement aboutir à un résultat plus satisfaisant et me concentrer rapidement sur d'autre chose.

Un autre problème est venu se poser au niveau de la reconnaissance de visage, celui qui allait me prendre le plus de temps.

4.4 Fiabilité d'un programme

Rapidement, je me suis rendu compte que lorsque j'enregistrais une multitude de visages dans ma base de donnée, la reconnaissance de visage n'était plus fiable. Le visage qui apparaissait à la caméra, et la personne mise en correspondance avec ce visage, n'était pas la même. Le programme fonctionnait des fois, d'autre fois non. Il était donc inconcevable d'affirmer que le programme était fonctionnel.

J'ai donc dû trouver la source du problème, ou plutôt des problèmes. C'est la première fois que j'ai ressenti une sorte de pression puisque j'ai stagné pendant pas mal de temps sur ce problème. Je ne pouvais pas mettre non plus cette partie de côté, puisqu'elle était importante et donc il fallait la finaliser le plus rapidement possible.

J'ai donc été confronté à ma première réel difficulté.

J'ai pu apprendre grâce à cette difficulté l'importance de savoir rechercher, et de savoir se questionner.

La première chose que j'ai faite a été de tester l'existant, ce qui se faisait déjà en matière de reconnaissance de visage grâce à la bibliothèque OpenCV.

J'ai pu alors rapidement comparer mon programme et ceux des autres, et comprendre là où il y avait des erreurs.

Par exemple, l'une des premières erreurs simples que je n'avais pas remarquées, était que j'enregistrais les visages en couleurs au lieu de les enregistrer en noir et blanc. Or, l'algorithme ne fonctionne qu'avec des visages enregistrés en noir et blanc. Avec mes images en couleurs, les informations qui était enregistrés était erroné et donnait donc des résultats aléatoires.

Note :

C'est à partir de cette erreur que j'ai aussi basculé de langage de programmation.

Je suis passé du C++ au Python dans le but d'implémenter plus rapidement et facilement les algorithmes que j'avais à faire. Je voulais faire ça pour être sûr de rendre un travail, puis, si j'avais le temps en fin de stage, traduire tous mes algorithmes du Python au C++.

Je n'ai finalement pas eu le temps de le traduire même si ce n'est pas si difficile.

Après avoir choisi de basculer en Python, j'ai décidé de changer mes méthodes de travail pour moins me jeter et plus réfléchir à dans quelle direction je vais.

Après quelques suggestions de mon tuteur de stage, j'ai donc choisi d'effectuer mes tests de manière plus professionnelle.

J'ai par exemple utilisé une base de donnée de visage déjà créé sur internet, libre de droit, contenant une multitude de visage.

Cela m'a par exemple permit d'obtenir beaucoup d'échantillon de visage, et de me faciliter le travail. Je n'avais pas à enregistrer des dizaines de photos prises sur internet de façon aléatoire. Je devais auparavant enregistrer une par une les photos au bon format, aux bonne dimensions, et en noir et blanc, afin de les enregistrer en base de donnée.

Je n'ai pas tout ce travail à faire avec une base de donnée faites par des professionnels.

De plus, le nombre de photos que j'avais en ma possession me permettait d'assurer des tests d'une certaine ampleur. J'avais en ma possession plus d'une centaine de visages différent, avec une cinquantaine de photos toutes différentes de chacun des visages.

Cela me permettait donc d'avoir une base sûre, et de faire des tests sur d'autre visage fiable (Le seul qui l'était réellement était le mien, un visage n'est bien évidemment pas assez pour juger l'algorithme comme étant fonctionnel).

Après de nombreux tests, j'ai finalement compris d'où venaient chacune de mes petites erreurs et pu aboutir à un résultat beaucoup plus convaincant en matière de reconnaissance faciale.

Le programme pouvait reconnaître plusieurs visages en temps réel. Même s'il peinait parfois en basse luminosité ou confondait encore quelques fois certaines personnes, le programme était bien plus fonctionnel, et il ne restait plus qu'à l'optimiser pour l'améliorer et le rendre viable.

4.5 Tracking vidéo

Après avoir complété la détection et la reconnaissance de visage, j'ai eu l'occasion de travailler légèrement sur le tracking vidéo.

Le tracking consiste à suivre un objet visible à la caméra.

Pourquoi le tracking est intéressant dans le cadre de ma mission ?

Ma mission est de reconnaître une personne, et de la suivre. Ainsi, en sachant dans quel la zone de la caméra la personne se situe, il est possible d'en tirer des informations intéressante (La personne se dirige vers le couloir, vers la chambre, sort, rentre etc...)

J'ai d'abord pensé le tracking comme un bon moyen de continuer à reconnaître une personne, même lorsque son visage n'est plus visible à la caméra.

Partons du principe qu'une personne est présente, face à la caméra. L'algorithme va donc la reconnaître car cette personne est enregistrée en base de donnée. Cependant, si elle effectue une rotation autour d'elle-même de 180 degrés, alors elle fait dos à la caméra, et le visage n'est pas visible, donc pas détecté, et donc pas reconnu.

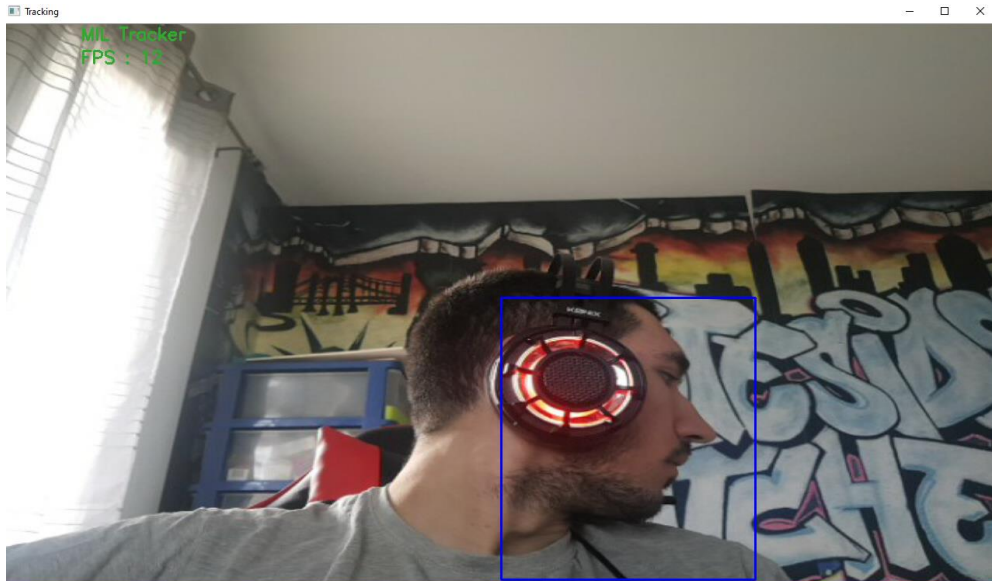
C'est un vrai problème qu'il faut palier et l'un des moyens que j'ai trouvés est de passer la main au suivi d'objet, au lieu de relancer une reconnaissance de visage toutes les secondes.

En effet, tracker un objet est bien différent de reconnaître un objet. Les algorithmes ne fonctionnent pas pareils.

En trackant un objet, on utilise les pixels environnant de l'image, c'est-à-dire, que le mouvement permet de savoir que même si l'objet qu'on suit n'a plus la même forme, c'est bien lui.

Ainsi une personne qui se retournerait, et qui aurait donc le visage non visible, serait quand même suivi, grâce à l'algorithme de tracking.

Pour cela, j'ai mis en place plusieurs algorithmes de tracking qui se base sur des techniques différentes. Je n'ai pas eu le temps d'effectuer des tests intéressants dessus, mais j'ai développé quatre techniques différentes, afin de pouvoir comparer l'efficacité de chacun des algorithmes

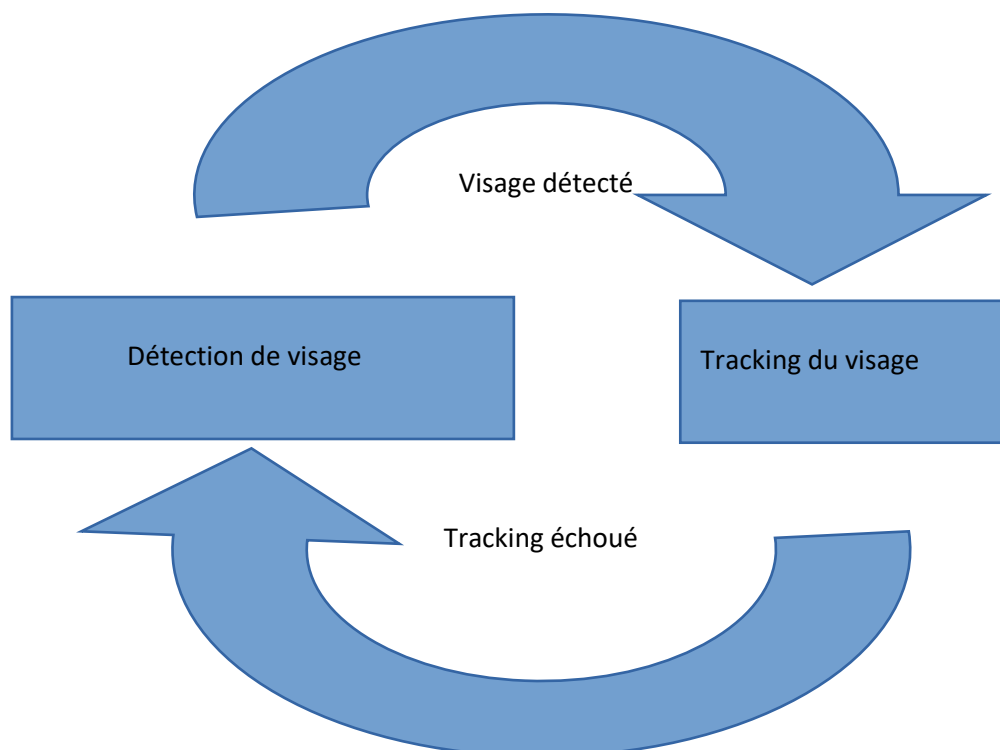


Algorithme de tracking MIL

Ici, l'algorithme de tracking suit le visage, et on voit bien que même de profil, le visage continue d'être suivi.

L'un des principaux problèmes qui se posent avec le suivi, c'est lorsque le sujet sort du champ de vision de caméra et rentre une nouvelle fois à l'intérieur.

Il y a un moyen de remédier à ça, c'est d'utiliser le principe décrit ci-dessous :



Lorsque le sujet sort du champ de vision, alors on relance une détection de visage, jusqu'à ce qu'un visage soit détecté. Une fois un visage détecté, on effectue une reconnaissance de visage sur ce visage.

Une fois le visage reconnu, on le suit via l'algorithme de tracking.

Si jamais le visage n'est plus visible car il est sorti du champ de vision, on relance une détection de visage.

Ainsi, avec cette boucle, on est censé avoir un programme qui détecte et tracke le visage continuellement et en temps réel.

Je n'ai pas eu le temps de finaliser ce programme car je me suis concentré sur le renforcement des autres qui étaient d'autant plus important.

4.6 Travailler sous Linux (Ubuntu/Debian)

Lorsque j'ai commencé à coder, je l'ai bien évidemment fait sur mon propre PC. Durant les premières semaines, je n'avais pas accès à la Raspberry, ainsi, je devais programmer sur un système similaire pour être sûr que cela fonctionnerait.

J'ai donc dû apprendre à installer Ubuntu, une première pour moi puisque je n'avais jamais réellement utilisé ce système d'exploitation.

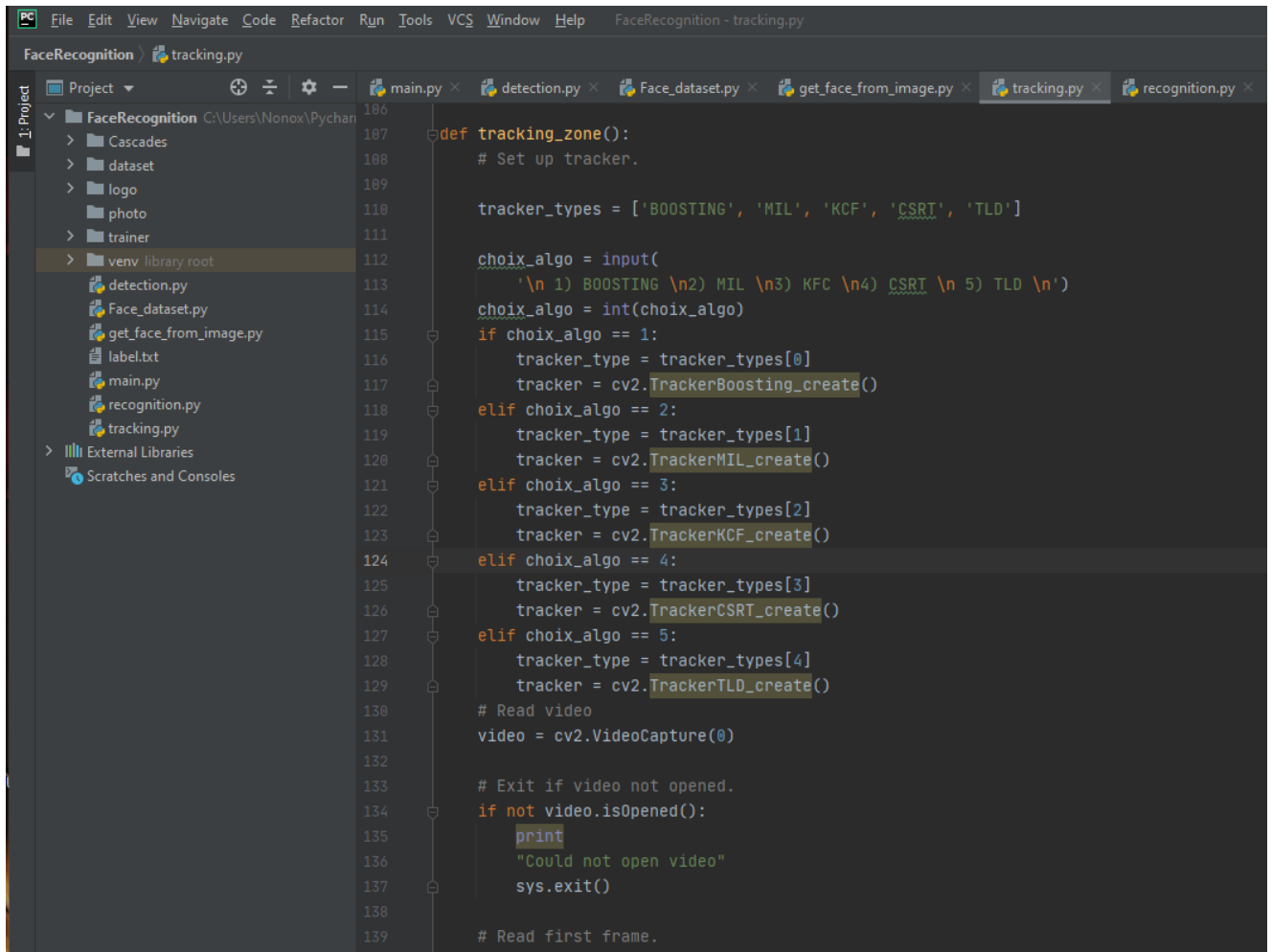
Il était important pour moi de travailler dessus afin d'être sûr que mon travail, une fois portée sur la Raspberry, fonctionnerait directement.

Le but était donc vraiment de rendre le travail installable en glissant uniquement le programme, et sans toucher à rien d'autre.

Je me suis rendu compte alors de certaines choses une fois la Raspberry en main, et pu comprendre quelles difficultés présente le portage de travail d'un PC à un autre, ou d'un PC à une Raspberry.

Le premier problème que j'ai dû faire face était de rendre le programme fonctionnable avec une version d'OpenCV installé sur la machine, et non pas sur l'EDI (Environnement de développement intégré). En effet, j'ai toujours favorisé le fait de travailler sur un EDI pour rendre plus simple la programmation, la lisibilité du code, et utiliser des fonctionnalités intéressante (l'auto-complétion par exemple, qui permet de nous suggérer directement les prochains mot clés à écrire sur notre ligne de code)

J'utilisais donc Pycharm durant mon stage, pour programmer en Python :



```

106
107 def tracking_zone():
108     # Set up tracker.
109
110     tracker_types = ['BOOSTING', 'MIL', 'KCF', 'CSRT', 'TLD']
111
112     choix_algo = input(
113         '\n 1) BOOSTING \n2) MIL \n3) KFC \n4) CSRT \n 5) TLD \n')
114     choix_algo = int(choix_algo)
115     if choix_algo == 1:
116         tracker_type = tracker_types[0]
117         tracker = cv2.TrackerBoosting_create()
118     elif choix_algo == 2:
119         tracker_type = tracker_types[1]
120         tracker = cv2.TrackerMIL_create()
121     elif choix_algo == 3:
122         tracker_type = tracker_types[2]
123         tracker = cv2.TrackerKCF_create()
124     elif choix_algo == 4:
125         tracker_type = tracker_types[3]
126         tracker = cv2.TrackerCSRT_create()
127     elif choix_algo == 5:
128         tracker_type = tracker_types[4]
129         tracker = cv2.TrackerTLD_create()
130     # Read video
131     video = cv2.VideoCapture(0)
132
133     # Exit if video not opened.
134     if not video.isOpened():
135         print
136         "Could not open video"
137         sys.exit()
138
139     # Read first frame.

```

Exemple de l'EDI Pycharm Version 2020.2.1

Pycharm permet d'installer directement OpenCV sur l'EDI, afin de faire les liens directement et de pouvoir utiliser directement les fonctionnalités d'OpenCV.

Installer manuellement OpenCV demande beaucoup plus de temps puisqu'il sera ensuite possible de l'utiliser sur n'importe quel environnement de travail.

Le fait d'avoir travaillé dans un premier temps avec une version d'OpenCV installé automatiquement m'a facilité la tâche car j'ai pu commencer à produire du code rapidement.

Je me suis rendu compte très vite qu'il y avait des problèmes lorsque je portais mon code sur Ubuntu. Lorsque je souhaite utiliser l'invité de commande pour lancer mes programmes sur Ubuntu, cela ne fonctionnait pas car OpenCV était installé de façon manuelle sur ce système d'exploitation, contrairement à l'environnement où je programmais, sous Windows.

Ainsi, j'ai compris que ma méthode de travailler sur les deux systèmes d'exploitation à la fois n'était pas la bonne et que je devais me pencher exclusivement sur Ubuntu pour être réellement sûr de la fonctionnalité de mon programme.

J'ai donc installé tout d'abord OpenCV de façon manuelle puis commencé à codé directement sur des éditeurs de texte.

Cela m'a demandé un temps d'adaptation car je n'avais pas l'habitude de travailler ainsi, mais cela m'a permis probablement de gagner beaucoup de temps sur le portage de mon programme, qui aurait été beaucoup plus compliqué si je n'avais pas procédé ainsi.

Arborescence des fichiers et chemins relatifs ou absolus

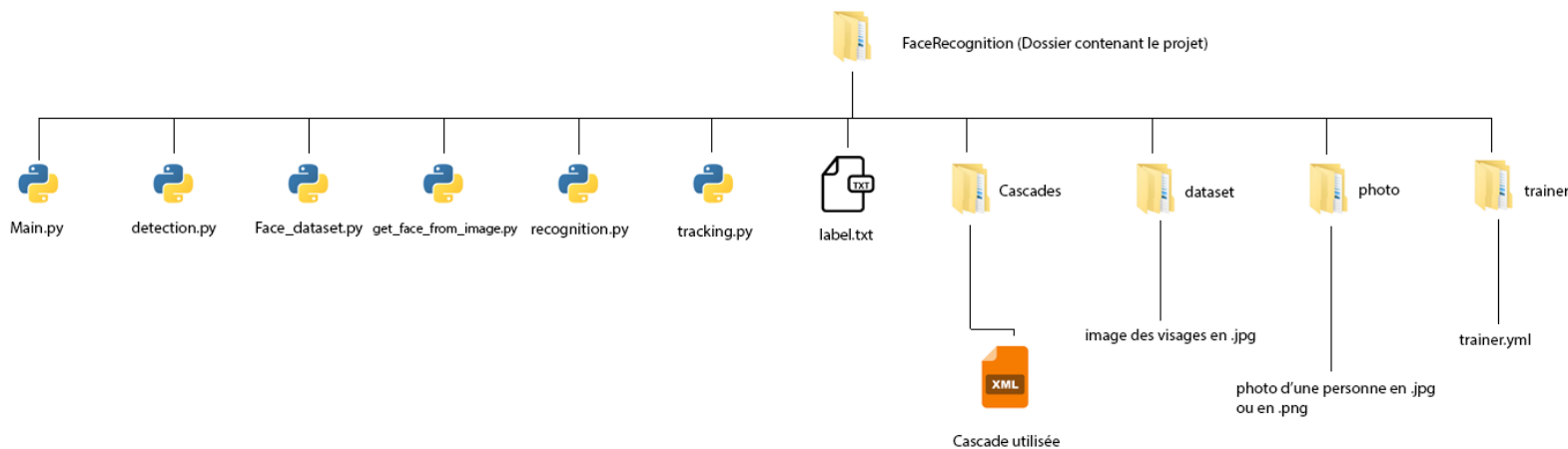
La deuxième chose qui allait me prendre du temps car je l'avais mal fait au début, était d'adopter une arborescence des fichiers intelligente.

Lorsque j'ai porté mon travail de Windows à Ubuntu, j'ai dû modifier cette arborescence puisque mes fichiers se trouvaient dans des dossiers complètement différents.

Sous Ubuntu, le but était d'installer le programme à la racine (root), ainsi, porter le programme sur une autre machine Ubuntu demanderait uniquement de porter le dossier à la racine.

Il n'y aurait donc qu'une action à faire, copier le dossier, et le coller à la racine, et cela est sensé fonctionner.

Mon programme ne fonctionnait pas sous un seul dossier, j'ai donc dû tout remodeler pour qu'il y ait un dossier source à tout ce projet :



Arborescence finale adoptée

Ainsi, tous les fichiers utilisés sont centralisés dans le dossier nommé *FaceRecognition*.

Le programme est donc portable facilement, en copiant cet unique dossier.

Après quelques tests, j'ai rapidement compris les problèmes qui allaient intervenir dû à mon ancienne structure et l'importance de l'utilisation des chemins relatifs dans un programme.

Les chemins d'accès absolus et relatifs sont deux manières différentes d'accéder à un fichier sur un ordinateur. Le premier (absolu) est le fait de spécifier l'adresse exact d'un fichier.

Exemple :

C :/dossier/fichier.py

Le deuxième (relatif) est le fait de spécifier l'adresse selon un répertoire qui est parent.

Exemple :

dossier/fichier.py

La différence est donc que le chemin relatif permet de chercher les fichiers à partir du dossier dans lequel le fichier exécuté se trouve.

En d'autre terme, travailler en chemin absolu peut créer certains problèmes lorsqu'on porte un projet d'un ordinateur à un autre, ce qui n'est pas le cas lorsque c'est bien fait quand on travaille avec des chemins d'accès relatifs.

Ce fut mon cas puisque j'ai rencontré ces problèmes-là lors du passage de Windows à Ubuntu.

La structure n'étant pas la même, les programmes ne fonctionnaient pas car lorsqu'ils appelaient d'autres fichiers, ils n'arrivaient pas à les retrouver dans les dossiers.

J'ai compris que ce problème allait se poser lorsque je porterai le programme sur la Raspberry dont je ne connaissais pas encore la structure. J'ai donc dû adapter mon code pour qu'il puisse fonctionner, peu importe sur quel ordinateur/Raspberry il serait exécuté.

J'ai retravaillé tous les chemins d'accès pour qu'ils deviennent des chemins d'accès relatifs au dossier source de mon programme.

Ainsi, cette fois-ci, porter le programme d'un ordinateur à un autre consistait bien à seulement glisser/déposer le dossier dans le dossier racine.

4.7 Optimisation de code et comprendre l'embarquée (Threading vidéo)

Lorsqu'on programme de façon scolaire, on cherche souvent à obtenir le bon résultat peu importe l'efficacité du code. Cela peut créer certains problèmes si l'on travaille de cette façon à plusieurs niveaux.

J'ai pu comprendre l'importance de se poser les bonnes questions et l'importance de l'optimisation de code lorsqu'on travaille sur un projet, et surtout lorsqu'on travaille sur un projet sur l'embarquée.

Je n'ai pas pris en compte dans un premier temps que le fait de travailler sur mon pc, pouvait donner des résultats différents une fois le code exécuté sur Raspberry. En effet, les deux machines n'offrent pas les mêmes performances. La Raspberry possède beaucoup moins de mémoire vive, et donc de puissance de calcul.

J'ai compris cela lorsque j'ai réussi à porter mon travail sur cette machine, et que mon programme ne fonctionnait pas correctement. Lorsque je captuais la vidéo de la caméra, le flux vidéo affiché à l'écran était très lent, à hauteur de 5 images par seconde, contre une trentaine sur mon pc.

La différence est énorme et non négligeable. A 5 images par secondes, il est impossible de tirer quelque chose du résultat obtenu par la Raspberry.

De plus, une grosse latence était visible (pas moins de 5 secondes d'écart entre l'image capturé, et l'action réellement effectuée. Ce sont donc bien des problèmes d'optimisation qui sont observés.

J'ai dû revoir mon code sur certaines portions, et le rendre plus rapide à la compilation, mais malgré ça, j'avais toujours ce problème d'image par seconde bien trop faible pour travailler correctement, j'ai donc dû chercher une solution efficace à cela.

Grâce à ce problème, j'ai pu apprendre beaucoup sur le threading et utiliser ce processus afin de résoudre ce problème.

Un thread est un processus qui se lance en parallèle de notre programme. En effet, lorsqu'on lance un programme sans thread, ce n'est pas toutes les capacités du processeur qui est utilisé pour calculer. De nos jours, les processeurs sont multi-cœurs, il est donc possible d'effectuer ce qu'on appelle des calculs en parallèle pour ne pas perdre de temps.

J'ai donc pu m'initier à cela en faisant calculer plusieurs fonctions de mon programme en parallèle, concernant le flux vidéo. Ainsi, j'ai pu obtenir un flux vidéo qui avoisinait les 30 images par seconde, sur la Raspberry et donc effectuer des calculs et analyse sur ce flux de façon correcte.

D'autres solutions alternatives ont été trouvées pour abaisser le temps de calcul, comme baisser la résolution de la caméra utilisée (car une grande résolution n'est pas forcément nécessaire dans le cadre du projet).

Je n'ai pas pu intégrer pleinement le système de threading au projet car cela a été réalisé à la fin de mon stage et que je manquais de temps. J'ai juste pu produire une réalisation qui prouve qu'avec le threading, il était possible de travailler correctement sur un flux vidéo sous Raspberry, avec une caméra connectée au réseau internet.

La leçon à retenir de cela est que lorsqu'on travaille sur de l'embarquée, il faut penser à tous ces problèmes d'optimisation et commencer dès le début à produire un code qui demandera un temps de calcul le plus bas possible, et utiliser tous les cœurs du processeur afin d'effectuer du calcul parallèle.

4.8 L'autonomie en entreprise

Dû à la crise sanitaire très spécial vécu durant le stage, les conditions du stage se sont avérés bien différentes de ce qui était prévu à la base. L'un des principaux changements est le fait que je travaillais sur ma propre machine. Cela empêche donc d'avoir un contact régulier avec son tuteur de stage et diminue le nombre de rencontre (que ce soit physique, ou en ligne via des logiciels tel que Zoom). Ainsi, il est important de savoir être efficace et de faire preuve d'autonomie.

Il n'est pas possible de demander de prendre contact chaque heure de la journée ou toutes les demi-journées. Cela n'est pas forcément efficace, et consomme énormément de temps. J'ai donc dû m'adapter et apprendre à faire preuve d'autonomie.

Je m'étais d'abord fixé de demander des réunions uniquement si j'en avais réellement besoin (si je bloquais réellement sur une partie importante par exemple), afin de me forcer à chercher moi-même les réponses aux questions que je me posais.

Si par exemple je n'étais pas sûr sur certaines choses, je prenais parfois quelques libertés afin de faire avancer le travail pour avoir toujours quelques choses à présenter durant les réunions obligatoires.

Un exemple précis, est la création du tracker vidéo. Ne sachant pas réellement quel algorithme serait le plus efficace concernant notre projet, j'ai donc décider de mettre en place plusieurs algorithmes différents afin dans un premier temps de les comparer, mais surtout de pouvoir montrer ces algorithmes comme un éventail de possibilités. Il est évident qu'un seul sera utilisé à la fin.

Cela faisait partie de ma mission, faire preuve d'autonomie m'a été demandé dès le début de mon stage.

Être autonome permet d'évoluer très rapidement lorsqu'on travaille efficacement car de nos jours tous ce que nous avons besoin est trouvable sur internet. Il est possible d'apprendre énormément de chose seul, et c'est une façon différente de procéder qu'il faut explorer afin de mieux se connaître.

Une organisation claire et efficace est primordiale afin de produire un travail intéressant. Je m'imposais donc personnellement des deadlines d'apprentissage de certaines choses pour ne pas perdre trop de temps à tourner en rond. Cela m'a été quelques fois utile sur certains points où je bloquais.

Être autonome ne signifie pas non plus devoir réussir tous ce que l'on entreprend. Cela signifie parfois qu'il faut savoir se résigner lorsqu'on n'y arrive pas ou que cela paraît impossible, ou alors de changer de méthode d'approche du problème confronté.

S'entêter sur un problème n'est jamais la bonne solution. Cela m'a valu une perte de temps lorsque je voulais absolument réussir à reconnaître un visage et que je n'y arrivais pas. La bonne solution était finalement de prendre du recul, changer de méthode de travail, et retenter plus tard. Je voulais absolument réussir à le programmer par moi-même, et j'oubliais l'un des principes de base de la programmation qui est le fait de s'inspirer.

Le but de mon stage n'était pas de recréer ce qui existait déjà mais de justement les tester, possiblement les comparer, les améliorer, et documenter dessus.

Ainsi, pour réussir à reconnaître un visage, la première étape fut de tester le programme d'autre personne, qui fonctionnait plus ou moins, et comprendre ce qu'il y avait de différent avec mon programme.

J'ai donc compris que faire preuve d'autonomie, c'est aussi faire preuve de maturité et savoir mettre son ego de côté pour apprendre et évoluer de façon correcte.

Même au niveau du matériel, j'ai dû m'adapter à mon environnement. J'avais en effet en ma possession une caméra que l'on m'avait fournie. Cependant, son utilisation chez moi (dû à la disposition des pièces), était très compliquée, puisqu'elle ne pouvait pas être proche de moi.

J'ai donc dû encore une fois m'adapter et trouver des solutions par moi-même, et j'ai donc ici utilisé mon téléphone comme caméra, ce qui rendait le test de mes programmes beaucoup plus simples. Une fois un programme terminé, je prenais donc mon temps pour le tester cette fois-ci sur la caméra qu'on m'avait fournie (elle nécessitait en fait plus de temps de préparation, et c'était impossible de travailler toute la journée avec de façon efficace, la perte de temps aurait été trop impactante).

4.9 La communication : savoir s'exprimer et parler en réunion

Pendant mes premières semaines de stage, chaque semaine, j'avais au moins une réunion où je devais présenter mon avancement sur le projet. Il est important de toujours pouvoir présenter quelque chose afin de montrer concrètement son avancée, que ce soit au niveau du programme ou de la documentation.

Je plaçais donc mes deadlines personnelles aux dates de réunions afin d'avoir toujours quelque chose sous la main à montrer.

Lorsqu'on s'exprime, il est important de savoir être clair. Il est compliqué pour quelqu'un qui n'a pas codé directement un programme de le comprendre car on a tous une manière différente d'aborder un problème. C'est pourquoi il est si important de commenter son code afin de laisser une trace écrite et une explication de chacune des lignes. Cela améliore la lisibilité du code.

Je devais donc m'exprimer de façon claire, sur ce que j'avais fait, ce que je comptais faire, et ce que j'arrivais ou non à produire. Il est aussi important de dire lorsqu'on n'y arrive pas, ou que l'on pense que c'est difficilement réalisable. Certains points m'ont pris plus de temps que prévu et j'ai donc dû le faire comprendre.

Vers la fin de mon stage, les réunions se faisaient cette fois-ci en présentiel, devant cette fois-ci des personnes qui allaient reprendre mon travail pour le continuer une fois que j'aurai terminé mon stage, et devant d'autres personnes un peu moins techniques.

J'ai donc dû apprendre à expliquer mon travail de façon synthétique, en le vulgarisant. L'un des principaux points en présentiel, lorsqu'on présente notre travail, est de faire en sorte que le matériel que l'on ramène fonctionne correctement. Mes premières réunions ont été marquées par des programmes qui ne fonctionnaient pas aussi correctement que chez moi dû aux problèmes d'importation du travail d'une machine à une autre vue précédemment.

Il est important de fournir un travail qui se voit visuellement pour faire comprendre que ça marche. C'est ce que je me forçais à faire après mes prochaines réunions. Cela demande de l'anticipation aussi puisqu'on me posait certaines questions, et que je me devais d'y répondre, avec une justification derrière ma réponse.

Exemple de question courante :

- Si l'on fait plutôt comme ça, est-ce que ça pourrait fonctionner ?
- Est-ce que le programme marche sur n'importe quelle machine ?
- Est-il possible d'effectuer cette tâche avec ce programme ?

Comme on peut le voir, ce sont souvent des questions de suggestions. On m'a souvent demandé si ce programme pouvait permettre d'effectuer tel ou tel chose. Pour anticiper ce genre de question, la meilleure des choses est de bien connaître ce que l'on a réalisé. Ainsi on connaît donc les limites et les fonctionnalités de notre production.

Enfin, échanger en réunion est très fructifiant, on obtient un regard différent du nôtre sur notre réalisation ce qui permet parfois de trouver des solutions que l'on n'aurait pas trouvé seul. J'avais pensé mon programme comme devant enregistrer lui-même des visages en base de données, j'avais donc créé une fonction permettant d'enregistrer une nouvelle personne. On m'a rapidement suggéré en réunion qu'il serait aussi intéressant de pouvoir enregistrer un visage à partir d'une banque d'images existantes déjà. Grâce à cette suggestion, j'ai pu fournir un programme plus logique et efficace à ce niveau-là car je n'y avait pas pensé dans un premier temps.

4.10 Documenter son travail

Etant donné que quelqu'un est censé reprendre mon travail pour le continuer après mon départ, il est important de laisser une trace de son travail afin de faciliter la compréhension l'utilisation de son programme par autrui.

Dès le début de mon stage, on m'a bien fait comprendre que la documentation faisait partie intégrante de ce que je devais produire, et que c'était même finalement l'un des points les plus importants. Sans documentation et explication, reprendre le travail de quelqu'un devient extrêmement compliqué, voire même impossible dans certains cas.

Afin de mieux savoir ce que je faisais et ce que je devais écrire, à la fin de chaque journée de travail, j'écrivais ce qui a été accomplie dans la journée, que ce soit partiellement, complètement, ou ce qui a aussi été échoué.

Ainsi, j'avais un suivi clair de mon travail et j'avais une partie de la journée qui était exclusivement réservé à écrire, de la documentation.

Tous les logiciels utilisés ont été listés, avec leurs versions, ainsi qu'un guide d'utilisation rapide.

Etant donné qu'OpenCV est une bibliothèque assez compliquée à installer, j'ai dû effectué aussi un guide d'installation d'OpenCV sous Windows, sous Linux, et sous mes différents EDI utilisés durant mon stage (CodeBlock et Pycharm, en particulier).

Même ce qui n'est pas forcément interne au projet, a été documenté, comme l'utilisation de logiciel permettant de faire de son téléphone une caméra relié au PC (Guide de Droidcam).

Le plus important reste de citer tout de même les sources utilisées afin de faire comprendre aussi au lecteur le chemin que l'on a abordé, ce qui est tout aussi important que la façon d'utiliser ou d'installer tel ou tel chose.

La documentation qui reste la plus importante reste celle sur le programme que j'ai dû créé. J'ai écrit un guide d'utilisation qui explicite chaque fonction du programme, leur limite, et le détail de chacune des variables/valeurs utilisées. Ainsi, même une personne qui ne comprendrait pas forcément tout pourrait utiliser le programme en se fiant au guide d'utilisation.

Cela passe aussi par un code commenté ligne par ligne le plus précis possible.

```
faces = faceCascade.detectMultiScale(  
    gray,  
    scaleFactor=1.2,  
    # redimensionnement de l'image pour détecter des visages plus petits (approche pyramidale de viola & jones)  
    minNeighbors=5, # permet d'éliminer les faux positifs en appliquant une approche au voisinage  
    minSize=(20, 20) # visage minimum détectable (20 * 20 pixels)  
)
```

Exemple de code commenté ligne par ligne avec détail des techniques utilisées

5. Conclusion

Ce stage de 12 semaines a été pour moi très important puisqu'il m'a permis de m'insérer pour une première fois dans le monde du travail. Il m'a permis de comprendre ce que l'on attendait de moi et comment allait se dérouler plus ou moins les choses en entreprise. On comprend rapidement les enjeux derrière nos missions et à quel point nos réalisations peuvent être importantes.

Je ne pensais pas dans un premier temps être capable de mener à bien un travail que je jugeais comme complexe. Lorsqu'on a très peu d'expérience professionnelle comme moi, on appréhende rapidement, et on se demande si on en est réellement capable, si l'on possède les capacités pour mener à bien son travail. J'ai pu donc plus prendre confiance en moi après avoir compris que souvent, on se sous-estime par rapport au travail que l'on peut effectivement produire.

J'ai en effet appris à m'adapter, à des conditions de travail particulière dû au conditions sanitaires. Le télé travail sera probablement ce qui nous attend comme étant une norme dans le futur, et avoir pu comprendre comment fonctionner efficacement à distance est forcément un avantage.

J'ai pu apprendre à être autonome, et faire confiance en mes propres capacités. Devoir respecter des deadlines nous poussent à rechercher par nous-même, et donc d'améliorer notre travail de recherche bibliographique. Cela cultive la curiosité professionnelle.

Ce stage a été très formateur au niveau de quelques points techniques. J'ai pu développer mes compétences en programmation, et explorer de nouvelle branche de la programmation dont je n'avais pas encore touché (thread, l'embarquée ect...). J'ai aussi appris à documenter de façon efficace. Certaines compétences très utiles dans n'importe quel travail ont aussi été développée comme le fait de s'exprimer en réunion, ou en visio-conférence, ainsi que savoir gérer son temps.

J'ai également compris qu'il était important de dire lorsqu'on bloque sur quelque chose ou que l'on pense que ce n'est pas réalisable. Avec de l'aide, on peut trouver des solutions ou alors dévier le sujet car effectivement, le temps ou les conditions font qu'une tâche n'est pas forcément réalisable.

Ce stage a été aussi une première porte pour m'ouvrir un réseau professionnel dans mon domaine. J'ai pu faire de bonne rencontre dans le plan professionnel.

J'ai pu aussi me conforter sur mes choix, ma filière et mes envies. Je sais maintenant ce que c'est que de travailler dans l'imagerie et je sais désormais que c'est quelque chose qui me plaît réellement. Etant de nature quelqu'un d'assez curieux, je ne doutais pas que j'allais apprécier, mais je sais désormais que je pourrais travailler dans cette branche sans aucuns soucis, même si bien évidemment j'aimerais voir d'autre chose pour élargir mon panel d'expérience personnel et de connaissance.

Enfin, je pense que ce stage fut très riche pour moi. Je sais désormais que je peux travailler dans le technique, que j'en ai les capacités. Le fait d'avoir réalisé quelque chose et d'en voir le résultat est très confortant. Cela me donne donc des idées de ce que je pourrais faire plus tard. Je me vois donc plutôt travailler dans une start-up, ou une petite boîte plutôt qu'une grosse entreprise, afin de voir une conclusion au travail que je produis, ce qui n'est pas toujours le cas dans des grosses entreprises, qui développe parfois certains projets sur plusieurs années, voire dizaine d'années. Afin de compléter mes compétences, j'aimerais cette fois-ci trouver un travail un peu moins technique, mais qui demande plus de compétences en rapport avec les relations humaines, qui demande de savoir manager et/ou d'imaginer de nouvelle chose plutôt que de forcément les concevoir.

6. Annexe

Extrait de documentation écrite



Guide d'utilisation de Droidcam

Introduction :

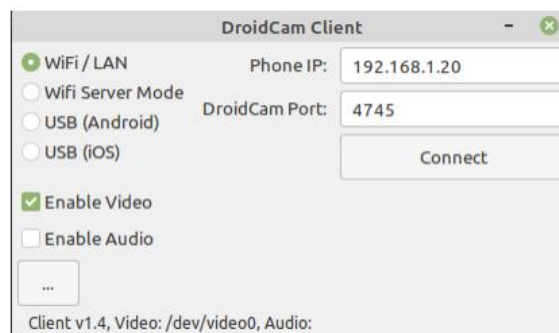
Droidcam est un logiciel permettant d'utiliser la caméra de son téléphone en tant que webcam. Il fonctionne sous toutes les dernières versions d'IOS et d'Android.

Installation :

1) Installer sur son PC :

-Télécharger le logiciel sur ce lien : <https://www.dev47apps.com/> , il est disponible pour Windows et pour Linux.

Une fois installée, ouvrir l'application , une page comme celle-ci apparaît :

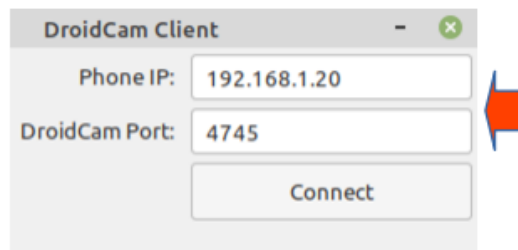


2) Installer sur son téléphone

-Se rendre sur le PlayStore ou l'AppleStore, et télécharger Droidcam.

3) Connexion du téléphone au PC

-Sélectionner le mode de connexion (Wifi ou USB). Pour le mode Wifi, il faut que l'ordinateur et le téléphone soit tous les deux connectés au même réseau.
Entrez ensuite sur le droidcam de votre ordinateur le Phone IP et le DroidCam Port noté sur l'application DroidCam de votre téléphone



-Une fois le bon port entré, cliquez sur « Connect », pour lancer la connexion.
La connexion est normalement établie.

4) Problèmes récurrents

-La connexion peut échouer si la connexion pour l'un des deux appareils est trop faible (connexion wifi trop faible), Une connexion par câble USB est alors favorable.
-La connexion peut ne pas s'établir correctement si une application du téléphone utilise en même temps la caméra (l'application Appareil photo, par exemple). Il faut donc fermer Droidcam, arrêter tous les logiciels utilisant la caméra, puis relancer Droidcam et effectuer la connexion au PC.

Bibliographie :

<https://www.dev47apps.com/>

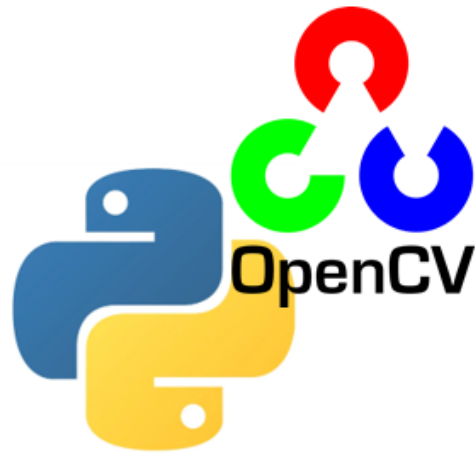
Guide d'utilisation du programme de reconnaissance faciale

Langage utilisé : Python 3.8

Bibliothèque utilisée : OpenCV 4.4.0

IDE utilisé : PyCharm Community Edition 2020.2.1

Fonctionne sur Linux, Raspberry, et Windows



Get face from image.py

Ce fichier contient 2 fonctions très semblables à celles du fichier «[Face_dataset.py](#)».

Au lieu d'enregistrer nous même les visages à partir d'une caméra, nous allons récupérer des photos déjà existante d'une même personne et les utiliser de la même façon.

Les photos de la personne à enregistrer doivent être stockés dans le dossier « **photo** ».

ATTENTION ! Uniquement les photos d'une et une seule personne doit être stockés à l'intérieur pour que l'algorithme fonctionne ! Le format des photos doit être en [.jpg](#) ou [.png](#), et la taille importe peu. Les photos doivent être en couleur ou en niveau de gris. Elles ne doivent pas forcément suivre un certain format de nom.

Exemple :

.idea	10/09/2020 14:50	Dossier de fichiers	
__pycache__	09/09/2020 13:28	Dossier de fichiers	
Cascades	03/09/2020 10:00	Dossier de fichiers	
dataset	04/09/2020 12:48	Dossier de fichiers	
photo	10/09/2020 15:41	Dossier de fichiers	
trainer	03/09/2020 12:08	Dossier de fichiers	
venv	03/09/2020 09:57	Dossier de fichiers	
detection	09/09/2020 13:26	Interactive Editor f...	5 Ko
Face_dataset	07/09/2020 12:45	Interactive Editor f...	4 Ko
get_face_from_image	07/09/2020 12:48	Interactive Editor f...	2 Ko
label	04/09/2020 12:48	Document texte	1 Ko
main	09/09/2020 12:59	Interactive Editor f...	2 Ko
recognition	07/09/2020 12:55	Interactive Editor f...	6 Ko
tracking	07/09/2020 12:56	Interactive Editor f...	6 Ko




```
cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

# Plus la variable confidence est basse, plus le visage prédit ressemble à la personne détecté
if (confidence < 100):
    fichier = open("label.txt", "r")
    a = 0
    while a != id:
        line = fichier.readline()
        a += 1
```

On effectue une prédiction via « predict() », qui prend en entrée une image, et recherche l'ID de l'objet avec le plus de ressemblance enregistrés via nos vecteurs du fichier « trainer.yml ». Il fait donc une mise en correspondance, et renvoie deux valeurs, l'ID, et la « confidence », qui est une valeur permettant de déterminer le taux de ressemblance (plus cette valeur est petite, plus l'objet ressemble à l'objet enregistré en base de donnée. On considère que si confidence = 0, alors l'objet du

```
id = line
confidence_value = round(100 - confidence)
confidence = " {0}%".format(round(100 - confidence))
if (confidence_value < 45): # plus cette valeur sera grande, plus le taux de tolérance
    id = "unknown"

else:

    id = "unknown"
    confidence = " {0}%".format(round(100 - confidence))

cv2.putText(img, str(id), (x + 5, y - 5), font, 1, (255, 255, 255), 2)
cv2.putText(img, str(confidence), (x + 5, y + h - 5), font, 1, (255, 255, 0), 1)

cv2.imshow('camera', img)

k = cv2.waitKey(10) & 0xff # Appuyez sur Echap
if k == 27:
    break
```

flux vidéo ressemble parfaitement à l'objet enregistré, ce qui en pratique impossible)

```
id, confidence = recognizer.predict(gray[y:y + h, x:x + w])
```

Si la valeur de confidence renvoyée par « predict() » est assez petite (le taux est arbitraire, ici on estime cette valeur à 100), alors on recherche le nom associé à l'ID dans le fichier « label.txt » par rapport au numéro de la ligne.

On calcule le taux de ressemblance, si ce taux est trop faible (arbitrairement, confidence < 45), alors on considère que le visage reconnu n'est pas le bon. On affiche donc comme nom « unknown »

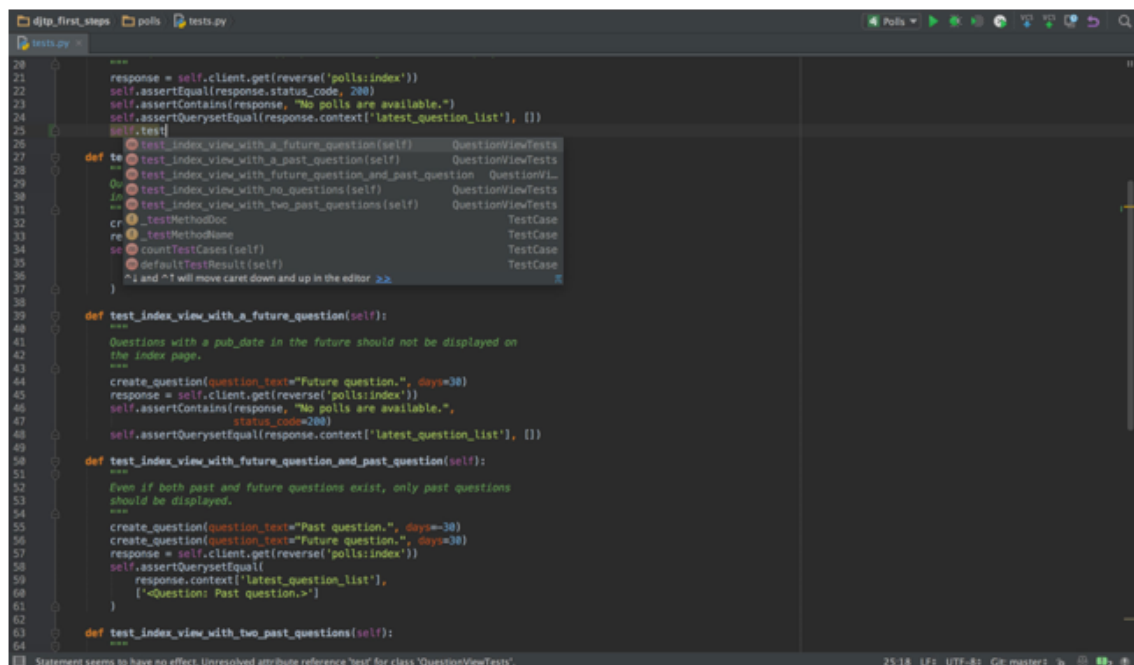
En résumé, il y a donc 2 possibilités pour qu'on ne reconnaisse pas un visage :

Installation d'OpenCV 4.4.0 sur PyCharm

Qu'est-ce-que Pycharm ?

Pycharm est un IDE permettant de coder en Python. Il permet surtout de simplifier la lecture du code et de le structurer plus facilement.

Voilà à quoi ressemble l'IDE :



1) Installation de Pycharm

Se rendre sur le lien suivant, et télécharger la version correspondant au bon OS (Disponible sous Windows, Mac et Linux) :

<https://www.jetbrains.com/fr-fr/pycharm/download/#section=linux>

Détecter un visage (Méthode en cascade)

Langage utilisé : C++

Prérequis (méthode implémenté à connaître):

CascadeClassifier : Classe d'objet permettant la détection d'objet. On peut charger un fichier xml de cascade en C++ comme suit :

`cascade.load(« Path/cascade.xml »)`

cvtColor() : Permet de convertir une image d'un espace colorimétrique à un autre. Cette méthode va surtout nous servir à transformer une image en niveau de gris.

Cascade detectMultiScale() : Méthode permettant de détecter des objets de différentes tailles dans une image en utilisant la cascade « **Cascade** »

Argument de la méthode detectMultiScale:

Mat image (une image)

vector<Rect> faces (un vecteur où l'on va stocker les rectangles contenant les visages détectés sur l'image),

double scaleFactor (change l'échelle de l'image permettant de détecter des visages plus petit, ou plus grand selon la valeur) ,

int minNeighbors (plus cette valeur sera grande, plus il faut de visage détecté « proche » pour le comptabiliser comme un. Par exemple, si un seul visage est détecté, et que minNeighbor = 3, alors ce visage ne sera pas pris en compte, cela s'appelle l'approche au voisinage) ,

int flag (cette variable est utilisée dans certaines anciennes cascades, ici, elle ne nous est pas utile, donc on laisse cette valeur à 0) ,

Size Size(height, width) (initialise le nombre de pixel à partir duquel on peut commencer à considérer qu'on détecte un visage . Un visage contenu dans un rectangle plus petit qu'un rectangle de longueur width et de hauteur height ne sera pas retenu)
