

# Rapport d'avancement Projet Système

## Acte 2 : ATRPortable

---

N. Le Roux E. Jacquemin V. Xu A. Alotaibi S. Almutairi  
[salman.almutairi@ensta-bretagne.org](mailto:salman.almutairi@ensta-bretagne.org)  
[ahmed.alotaibi@ensta-bretagne.org](mailto:ahmed.alotaibi@ensta-bretagne.org)  
[victor.xu@ensta-bretagne.org](mailto:victor.xu@ensta-bretagne.org)  
[eliot.jacquemin@ensta-bretagne.org](mailto:eliot.jacquemin@ensta-bretagne.org)  
[nicolas.le\\_roux@ensta-bretagne.org](mailto:nicolas.le_roux@ensta-bretagne.org)

## 1 Résumé

Le projet consiste à développer une application Android capable de faire du traitement d'images dans l'objectif de faire de la reconnaissance d'individus. Ce traitement d'images se sépare en trois parties majeures : la détection , le tracking puis l'identification. Le premier objectif est d'être en mesure de rentre la totalité du code fonctionnel sur un téléphone Android dans le but de faire fonctionner l'application uniquement avec le téléphone. Le second objectif est d'être capable de récupérer sur un ordinateur une vidéo en streaming ou enregistrée par un téléphone Android. La démarche serait alors d'appliquer le traitement d'image sur l'ordinateur et d'y afficher le résultat. Cet objectif pourra être amélioré en développant un système de multi-caméras.

En cette fin de semestre, le code concernant le traitement d'images permet la multi-détection de visage et de corps humains. Cependant la partie tracking n'est pas encore complètement fonctionnelle. La partie identification n'a pas encore été abordée. L'application Android va être développée sur BeeWare. Pour le moment seul l'application développée sur Windows est opérationnelle.

## 2 Abstract

The project consists of developing an Android application capable of image processing for the purpose of individual recognition. This image processing is divided into three major parts: detection, tracking and identification. The first objective is to be able to implement all the functional code on an Android phone in order to make the application work only with the phone. The second objective is to be able to retrieve a streaming or recorded video from an Android phone on a computer. The process would then be to apply the image processing on the computer and display the result. This objective could be improved then by developing a multi-camera system.

At the end of this semester, the code concerning image processing allows multi-detection of faces and human bodies. However, the tracking part is not yet fully functional. The identification part has not yet been addressed. The Android application will be developed on BeeWare. For now, only the application developed for Windows is operational.

## Table des matières

<b>1 Résumé</b>	<b>2</b>
<b>2 Abstract</b>	<b>2</b>
<b>3 Introduction</b>	<b>5</b>
3.1 Présentation du projet . . . . .	5
3.2 Exigences du projet . . . . .	5
3.3 Situation au début du semestre 4 . . . . .	6
<b>4 État de l'art</b>	<b>6</b>
4.1 Détection . . . . .	6
4.1.1 Méthodes pertinentes . . . . .	7
4.1.2 Sélection des régions candidates . . . . .	8
4.1.3 Détermination de présence dans des régions candidates . . . . .	8
4.2 Tracking . . . . .	9
4.2.1 Qu'est-ce qu'une image ? . . . . .	9
4.2.2 Qu'est-ce qu'une video ? . . . . .	10
4.2.3 Qu'est-ce que le tracking ? . . . . .	10
4.2.4 Le tracking 2D . . . . .	11
4.2.5 Le tracking 3D . . . . .	11
4.2.6 Le tracking planaire . . . . .	11
4.3 Identification . . . . .	12
4.3.1 Méthode d'identifications existantes . . . . .	12
4.3.2 Solutions potentielles adaptées à notre étude . . . . .	12
<b>5 Ingénierie-Système</b>	<b>14</b>
5.1 Exigences . . . . .	14
5.2 Architecture Externe . . . . .	14
5.3 Architecture Interne . . . . .	17
5.4 Architecture Physique . . . . .	18
<b>6 Réseaux et Application</b>	<b>19</b>
6.1 Gestion du réseau . . . . .	19
6.2 Application . . . . .	20
6.2.1 Android Studio . . . . .	20
6.2.2 BeeWare . . . . .	21
6.2.3 DroidCam : choix final . . . . .	21
<b>7 Traitement de Flux Vidéo</b>	<b>22</b>
7.1 Détection . . . . .	22
7.2 Suivi . . . . .	23
7.3 Identification . . . . .	23
7.4 Bilan . . . . .	24
<b>8 Méthode Agile</b>	<b>25</b>
8.1 Diagramme de Gantt . . . . .	25
8.1.1 Premier diagramme de Gant . . . . .	25
8.1.2 Deuxième diagramme de Gant . . . . .	25
8.2 Evolution de la gestion de notre projet . . . . .	26

<b>9 Tests unitaires, d'intégration et validation</b>	<b>27</b>
9.1 Tests unitaires . . . . .	27
9.2 Tests d'intégration . . . . .	27
9.3 Tests de validation fonctionnelle et écarts observés . . . . .	27
<b>10 Impact environnemental</b>	<b>29</b>
<b>11 Conclusion</b>	<b>30</b>
<b>12 Bibliographie</b>	<b>31</b>

### 3 Introduction

#### 3.1 Présentation du projet

De nos jours, de nombreuses techniques numériques et d'algorithmes de traitement d'image ont été développées afin de répondre au besoin de reconnaissance. Cette reconnaissance peut être définie sous plusieurs aspects. Premièrement le type d'objet que le système souhaite reconnaître : des humains, des animaux, des voitures... Ensuite, une fois l'acteur détecté, plusieurs types de données peuvent être récoltées afin de répondre aux objectifs du système. Ces dernières années, l'identification et le tracking des individus est un domaine qui s'est considérablement développé, jusqu'à être utilisé de manière très concrète par des gouvernements (le gouvernement chinois par exemple). C'est dans cet esprit de détection et identification en temps réel d'êtres humains que notre projet naquit.

Notre équipe s'est proposée de développer une application sur Android permettant le traitement d'image, la détection, puis le tracking et enfin l'identification de visages humains. L'objectif principal proposé est effectivement assez vaste, ce qui nous permet d'avoir une marche de manœuvre sur plusieurs aspects du projet. En effet, nous avons à l'origine de ce projet beaucoup d'aspect qui sont flous, sur lesquels nous allons devoir lever le voile. Nous n'avons pas mise en place directement un contexte particulier dans lequel nous aimerais faire fonctionner notre système. Celui-ci se précisera au fur et à mesure de nos performances et de notre avancement global sur le projet. L'objectif originel est d'être capable de détecter tous les humains présents dans un environnement quelconque et de leur attribuer un identifiant. On souhaiterait ensuite que notre application soit capable de tracker les humains détectés, grâce notamment à leur identifiant, peu importe leur mouvement ou leur orientation par rapport à la caméra. Enfin, une dernière contrainte importante est celle de savoir identifier les humains détectés grâce à une base de données récoltées au préalable.

#### 3.2 Exigences du projet

Une première interrogation fut éthique et liée à l'identification des visages. En effet nous souhaitons préciser qu'il est primordial pour nous que les humains identifiés aient acceptés d'être intégrés dans notre base de données et de ce fait être potentiellement identifiés par notre application.

Rappelons que les trois objectifs principaux de notre projet sont donc : détecter, suivre et identifier. Les codes répondants à ces objectifs seront développés sous Python. En effet même si Matlab était une possibilité, la majorité des membres de l'équipe sont à l'aise avec Python mais seulement des initiés à Matlab. Ensuite, il est important de développer notre application sur un environnement de développement lié à Android. D'un point de vue de l'application Android Studio semblait être le choix de prédilection, seulement le récent environnement BeeWare a également éveillé notre curiosité, tout comme l'application DroidCam. Dans tous les cas l'objectif reste le même : développer une méthode permettant d'effectuer de la reconnaissance, du suivi et de l'identification d'êtres humains en streaming sur ordinateur, à partir des données vidéos d'un téléphone portable.

En effet, une partie importante de notre projet est le streaming. En effet, nous souhaiterions développer notre application de manière à ce qu'il soit possible de l'utiliser de la manière suivante : utiliser notre téléphone simplement pour récolter les flux vidéos afin de les transférer à un ordinateur exécutant notre code et affichant les résultats sur l'écran de la machine. Il serait pertinent de mettre en place par la suite une méthode de multi-caméras. En d'autres termes, la finalisation de la partie "streaming" du projet serait de pouvoir récolter les données vidéos de plusieurs téléphones en même temps, puis qu'elles soient toutes traitées et affichées sur la même machine, à l'instar des caméras de surveillance dans des lieux publics.

Ainsi, ce projet comporte déjà plusieurs exigences qui, une fois satisfaites, permettraient de posséder une application de traitement d'images complète mais qui possède assez d'élasticité pour s'appliquer à un contexte encore non définitif. Cependant il est important de prévoir que les parties d'identification et multi-caméras seront les plus compliquées à achever. Les résultats de fin de semestres permettront d'avoir un premier aperçu de notre avancement.

### 3.3 Situation au début du semestre 4

Au début de ce semestre 4, notre équipe s'est fixée des objectifs clés pour mener à bien notre projet de développement d'une application de reconnaissance d'humains. Notre entretien avec nos encadrants Mr. Toumi et Mr. Cexus nous a permis de clarifier certains points ainsi que de mieux nous orienter pour la suite du prochain semestre.

L'un de nos objectifs principaux était d'affirmer un objectif clair d'utilisation de notre algorithme de reconnaissance. Nous souhaitions définir un objectif clair afin de pouvoir adapter au mieux notre code ainsi que la partie réseaux de notre projet. En effet, selon le contexte dans lequel nous souhaitons adapter notre application nous devrons développer une méthode de gestion des réseaux en adéquation, ce qui peut être parfois une tâche très difficile. Après entretien avec les encadrants, nous étions très intéressés à l'idée de faire de la reconnaissance en streaming sur l'ordinateur. Le but serait par conséquent de récupérer le flux vidéo d'un téléphone Android sur notre ordinateur afin d'y appliquer notre programme.

Un autre objectif important que nous étions fixés était donc d'avoir un plan clair de notre méthode de gestion du réseau que nous souhaiterions mettre en place. Nous avions prévu un temps conséquent à consacrer à la partie réseau. Dans un premier temps, nous pensions être important d'effectuer un recensement des différentes options possibles que nous pourrions adapter à notre projet sans trop de difficultés. Nous avions peur de passer trop de temps sur cette partie jusqu'à s'y perdre dans le projet, donc nous souhaitions connaître tout le panel de méthodes auxquelles nous avons accès afin d'utiliser celle qui s'adapte au mieux à notre projet sans que la difficulté soit trop haute.

En outre, nous avions également pensé à notre ultime objectif de pouvoir faire de la reconnaissance en mode multi-caméras. Remplir cet objectif nous permettrait d'élargir la portée de notre algorithme et de l'utiliser dans un plus grand nombre de situation. Evidemment, cette méthode reste pertinente seulement tant que la qualité de la reconnaissance faciale est satisfaisante. Nous étions conscients que le multi-caméras demande une qualité d'algorithme et de gestion du réseau supérieur. C'est pourquoi nous avions prévu de garder à l'esprit cet objectif tout au long du semestre en prévision.

Enfin, nous souhaitions améliorer la qualité de notre programme Python de détection, tracking et identification d'humains. Pour les parties détection et tracking qui étaient fonctionnelles, nous voulions simplement chercher à améliorer son efficacité tout en optimisant la latence de l'application. Pour ce qui est de l'identification, nous avions à cette heure-ci seulement notre état de l'art. En effet, nous n'avions pas encore une partie de notre code consacrée à l'identification. Nous souhaitions réaliser cette tâche au cours du semestre 4. On savait également qu'il y avait quelques questions éthiques à se poser avant d'utiliser notre programme final.

## 4 État de l'art

### 4.1 Détection

De manière générale, les méthodes de recherches se différencient en deux grandes catégories : template matching approaches (approches de comparaison de modèles) et *trained classifier approaches* (approches du classificateur entraîné). La première catégorie a pour but de détecter en utilisant

différentes fonctionnalités mathématiques capables de détecter à partir d'un modèle. La deuxième catégorie s'effectue en deux étapes : l'obtention d'informations grâce au machine learning puis la détection grâce toutes les données récupérées.

#### 4.1.1 Méthodes pertinentes

Dans cette partie on présente les méthodes utilisant des fonctionnalités mathématiques permettant de modéliser une personne dans un premier temps, pour ensuite la détecter sur une image. Pour se faire, la disposition de pixels rangés en bloc appelés « cellule » ainsi que la connaissance de leurs caractéristiques mathématiques seront cruciales.

##### *Haar-Like*

Cette méthode de détection a été développée principalement par Viola et Jones ([Petitpa](#), ). Pour la détection de visage, cette approche utilise les propriétés de nombreuses images positives et négatives ([Viola et Jones, 2001](#)). Dans le domaine de la vision artificielle, elle reste actuellement la technique la plus demandée. Son utilisation atteint des sujets plutôt complexes tels que la détection des mouvements des yeux. Cette méthode est fournie par OpenCV. Cette bibliothèque permet d'avoir accès à cette méthode en offrant un concept nommé cascade de classificateurs ([Palowski, 2019](#)).

##### Histogrammes d'orientation des bords (HOB)

Cette approche cherche à discerner la silhouette ainsi que les contours de la personne afin de pouvoir le discriminer du reste de l'image. L'avantage d'HOB est qu'il est possible de détecter toutes formes géométriques recherchées. L'HOB est aujourd'hui utilisé pour la détection de panneaux de signalisation permettant par la suite d'informer le conducteur. Cela permet de l'avertir de certains dangers dans le cas où il n'aurait pas remarqué le panneau. Dans le milieu de la reconnaissance de mouvement du visage, l'HOB est envisagé, pour par exemple détecter les sourires. À notre échelle, il ne semblerait pas que la librairie OpenCV permette réellement d'utiliser l'HOB.

##### Histogramme des graphes orientés (HGO)

De la même manière que l'HOB, la méthode HGO s'intéresse à la silhouette ainsi qu'à son contour pour la détection. Cependant HGO est plus complexe et plus efficace. En effet, ici chaque pixel possède une orientation de contour qui possède son propre sens ainsi que sa propre intensité ([Dupoirion, 2018](#)). Ainsi chaque pixel possède un vecteur. Une HGO est réalisée par groupe de pixel appelé « cellule ». Les vecteurs subissent ensuite une classification grâce à un SVM (Support Vector Machine) ([Wikipedia, 2018](#)). C'est de cette manière-là que la détection pourra ensuite se faire. HGO a été par exemple utilisé pour la reconnaissance automatique d'objet sur des images en couleur. Un logiciel de reconnaissance d'écriture a également nécessité l'efficacité de la méthode de détection HGO, ainsi que le programme de classification SVM pour mener à bien le projet. Nous concernant, OpenCV possède des fonctions/bibliothèques nous permettant d'appliquer cette méthode. C'est d'ailleurs OpenCV qui est utilisé par la grande majorité des utilisateurs de l'HGO.

##### Modèle Binaire Local (MBL)

L'approche a plutôt pour but de réaliser la description d'un visage. Le principe consiste à déterminer des seuils pour chaque pixel, puis traduire le résultat en binaire pour réaliser ensuite une classification. En comparant la valeur de pixel concerné avec ses voisins, les résultats seront donc 0 ou 1 en fonction du résultat ([Trefny, 2010](#)). Des « cellules » sont dessinées, puis un histogramme est calculé ainsi que normalisé. Dans le domaine de la détection, MBL se penche sur le domaine de la description d'expressions faciales. Actuellement, il est utilisé puis complémenté par certaines extensions dans le

but par exemple de faire de la détection d'objet rapide ([Pietikäinen, 2010](#)). Il est envisageable pour nous d'utiliser LBP: OpenCV et Python permettent de réaliser le Modèle Binaire Locale d'une image ([loc, 2020](#)).

### Contexte de forme

Le principe est de prendre un certain nombre de points formant appartenant au contour. Pour chacun de ces points, on calcul sa distance euclidienne puis son angle par rapport à l'axe vertical, horizontal,... On peut alors réaliser un histogramme. On normalise ces points, puis applique certaines fonctions avant d'atteindre le résultat ([Nikishaev, 2018](#)). Plusieurs logiciels de reconnaissance d'objet ont été développés grâce à cette méthode. Cette méthode est accessible sur Python ou OpenCV.

### Descripteur de covariance de région

Le principe de cette approche est de prendre en compte un maximum de données sur chaque pixel de l'image tel que l'intensité, la couleur, ... Pour chaque pixel de chaque cellule de l'image, une matrice vectorielle est remplie par ces données. Une matrice carrée est déduite selon les différences de valeurs pour une même cellule. Chacune de ces matrices permet une description précise de la cellule ([Klingler,](#) ). Cette approche est actuellement utilisée dans le domaine de la détection de véhicule depuis un satellite, mais également pour développer un logiciel de re-identification de personnes. Sachant qu'il est possible de réaliser des calculs de matrices covariantes avec OpenCV ou avec Python, il semblerait possible d'appliquer cette approche à notre échelle.

#### 4.1.2 Sélection des régions candidates

Les approches qui vont suivre sont applicables seulement si on possède de nombreuses images positives de la personne en question.

### Recherche par fenêtre glissante

Cette approche s'utilise lorsque l'on ne possède aucune information spécifique sur l'image. Le principe est de scanner avec une fenêtre glissante. L'échelle de la fenêtre ou la taille de l'image (selon le contexte) peut être modifiée ([Datalya, 2018](#)). Chaque taille de fenêtre correspond à celle disponible dans les classificateurs SVM... OpenCV et Python peuvent être intéressants, plusieurs codes et documents ont déjà été publiés à ce sujet ([Rosebrock, 2015](#)).

### Filtrage par connaissance du monde réel

Le principe est le même que la méthode précédente, mais il n'y a pas de modification de l'échelle de scan, cependant celle-ci est très faible. Cette méthode impose d'importantes contraintes sur les moyens mis à disposition pour la détection.

#### 4.1.3 Détermination de présence dans des régions candidates

La dernière étape de la détection de personnes, décider si la région candidate contient une personne. La majorité des algorithmes modernes reposent sur le principe des classificateurs. Même si l'apprentissage profond introduit des nouvelles techniques, par soucis de complexité de la théorie sous-jacente, nous allons nous focaliser sur l'apprentissage automatique plus classique. Ces méthodes restent en effet toujours pertinentes dans le domaine de la détection de personnes, même à l'ère du deep learning. L'usage des classificateurs est l'approche la plus répandue, et les méthodes de Boosting et SVMs sont les plus utilisées ([Souded, 2013](#)). Sans entrer dans les détails, nous pouvons expliquer les concepts des

différentes approches. Dans les deux cas, on dispose de bases de données à classifier par appartenance ou non à une classe. Les données d'entraînement sont déjà identifiées.

### Support Vectors Machines (SVM) et Boosting

Pour les SVMs, Support Vector Machines, le but est de représenter les données dans un espace fictif, et de trouver un hyperplan séparateur des données positives, qui appartiennent à la classe, et des négatives, qui n'en font pas partie. Grâce à une fonction “noyau”, on se place dans un espace de dimension supérieure où il est plus probable de trouver une droite séparatrice linéaire ([dat](#), ). Quant aux méthodes dites de Boosting, on prend aléatoirement des données positives ou négatives, et on entraîne un classificateur avec. Ensuite, on prend les connaissances de ce premier classificateur et les ajoute à celles d'un prochain classificateur. On progresse itérativement, en ajoutant des classificateurs dont les poids sont pondérés par leurs compétences à bien classifier des données. De plus, les données mal-classées par les classificateurs antérieurs auront une probabilité supérieure de réapparaître dans les générations suivantes, pour les entraîner sur des données plus difficiles. En procédant ainsi, on construit des classificateurs de plus en plus fort. Il existe de nombreux algorithmes de boosting différents, nous pouvons en citer trois populaires: Adaboost, Gradient Boosting et XGBoost ([IBM, 2021](#)). Ensuite, on peut décider de procéder par détection d'un corps humain entier, où la fenêtre de recherche aura un rapport hauteur/largeur similaire à un humain, ce processus est souvent utilisé pour la détection de piétons. L'autre procédé est la détection de membres : tête, bras, jambes, mains... Où il faut d'abord détecter différents composants d'un humain, puis les positionner relativement aux autres, avant d'en déduire la présence d'un humain.

### Bibliothèques Python pertinentes

Pour l'implémentation éventuelle de ces procédés dans notre projet, nous allons nous appuyer sur des bibliothèques Python, dont leur diversité et leur richesse sont de grands atouts du langage. Une référence pour le traitement d'images et vidéo sur Python est la bibliothèque multi-langage OpenCV. Elle possède des outils pour travailler avec des SVMs ([García](#), ), et la documentation officielle sur ceux-ci est copieuse et illustrée avec des exemples. Il en est de même pour le Boosting ([OPe](#), ). Bien sûr, OpenCV n'est pas l'unique bibliothèque Python capable de faire du machine learning, une autre référence est Tensorflow ([InfoWorld, 2022](#)). Celle-ci sert beaucoup à entraîner des réseaux de neurones, et elle est plus apte à entraîner des modèles que OpenCV. Cependant, OpenCV est plus optimisé pour du traitement d'images et de vidéo ([Science, 2019](#)). Ainsi, si nous voulions une détection plus efficace, il serait sage d'entraîner nos modèles grâce à Tensorflow, puis de les appliquer à des vraies données avec OpenCV. D'autres options sont ouvertes à nous, comme Numpy, Scipy, Matplotlib, etc. Mais leurs réputations pour nos accomplir objectifs sont moins bonnes que nos deux premiers exemples. Dû à ses capacités de traitement d'images et vidéo, nous nous tournerons certainement vers OpenCV.

## 4.2 Tracking

Avant de parler de tracking, il nous semble judicieux de rappeler ce qu'est une image et ce qu'est une vidéo.

### 4.2.1 Qu'est-ce qu'une image ?

Une image numérique est un tableau de valeurs, une matrice de nombres compris entre 0 et 255 (codés sur 8 bits soit 256 valeurs possibles). Une case de ce tableau correspond à un pixel (qui provient de l'anglais, PIcture Element) de l'image. Chaque pixel possède donc des coordonnées (x,y) au sein de l'image.

Pour une image en niveau de gris, il n'y a qu'une seule matrice de valeurs comprises entre 0 et 255.

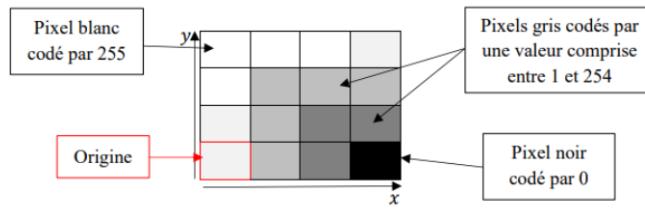


FIGURE 1 – Matrice pour image monochrome

Le pixel noir a pour coordonnées (3,0). Pour une image couleur, il y a trois matrices de valeurs comprises entre 0 et 255 : une matrice pour la couleur rouge, une pour la couleur bleue et une dernière pour la couleur verte.

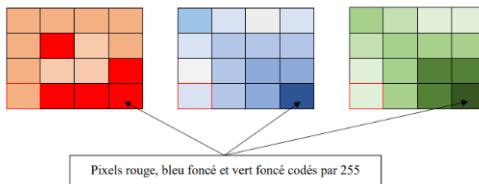


FIGURE 2 – Matrices pour une image RGB

Le pixel blanc est un mélange de rouge (255), de vert (255) et de bleu (255).

#### 4.2.2 Qu'est-ce qu'une vidéo ?

Une vidéo est une succession d'images qui défilent les unes à la suite des autres avec une certaine fréquence  $f$ . Par exemple, au cinéma, la fréquence des images est de 24Hz soit 24 images par seconde et, à la télévision française, la fréquence est de 25Hz et 50Hz.

#### 4.2.3 Qu'est-ce que le tracking ?

Le tracking est une technique qui va permettre de déterminer la trajectoire d'un objet cible en mouvement dans une vidéo. Cette technique est aujourd'hui très utilisée aussi bien pour le cinéma, les reportages télévisés que par des applications contenues au sein des téléphones portables. La technique du tracking utilise 2 méthodes :

- La détection d'un objet cible dans chacune des images qui constituent la séquence vidéo.
- La correspondance d'une image à l'autre pour assurer une cohérence dans le suivi de l'objet cible au cours du temps.

Il existe différentes techniques de tracking : - Le tracking 2D - Le tracking 3D - Le tracking planaire. Chacune de ces techniques va être expliquée brièvement dans les parties suivantes.

#### 4.2.4 Le tracking 2D

Cette technique va permettre de repérer un objet cible au sein d'une image. Pour cela, il faut déterminer 2 zones au sein de l'image : - Une zone de référence qui est une petite zone qui contient tous les pixels ciblés dont on veut faire le suivi. - Une zone de recherche qui est une zone qui contient la zone de référence. Elle peut être de taille variable : elle peut se restreindre à une petite zone autour de la zone de référence (plutôt quand les mouvements sont lents) ou englober l'image complète (plutôt lorsque les mouvements sont rapides). Ces zones sont généralement des carrés. La zone de référence établie, il est possible de connaître sa position, ses coordonnées ( $x,y$ ) au sein de l'image. En faisant de même sur toutes les images qui composent la vidéo, il est possible de reconstituer la trajectoire de l'objet cible. Les algorithmes qui permettent la recherche et la détermination de la position d'un objet cible au sein d'une image utilisent 2 paramètres : - La luminance qui correspond au rapport entre l'intensité lumineuse de l'objet cible et l'intensité lumineuse du reste de l'image. Plus ce rapport est grand et plus le contraste est grand. - La chrominance qui correspond à l'étude des couleurs. Il semblerait que la bibliothèque OpenCV de Python permette de travailler avec ces deux paramètres pour la détection d'un objet cible. Par exemple, il est plus facile de suivre le mouvement de la pupille d'une personne qui a les yeux bleus que d'une personne qui a les yeux marrons voire noirs car le contraste entre le bleu et le noir est beaucoup plus important que le contraste entre le marron et le noir.



FIGURE 3 – Comparaison de chrominance

La technique du tracking 2D peut se faire à l'aide : - D'un point, d'une zone de référence. Un seul des deux yeux est utilisé comme zone de référence par exemple. Cette technique de tracking peut se faire à l'aide du logiciel Motion 5 d'Apple. - De 2 points, de 2 zones de référence. Les deux yeux sont utilisés comme zones de références par exemple. Cette technique reste pourtant limitée car l'objet cible doit toujours être présent sur les images pour pouvoir assurer son suivi. Dans le cas contraire, ce serait à l'utilisateur d'estimer sa position.

#### 4.2.5 Le tracking 3D

Le tracking 3D est similaire au Tracking 2D excepté sur le nombre de points de référence. On passe de 1 à 2 points pour le tracking 2D à plusieurs dizaines voire centaines pour le tracking 3D. Du fait de ce grand nombre de points, les informations à traiter sont importantes et nécessitent donc des machines performantes. Cette technique n'est utilisée que par un nombre limité de personnes, d'artistes spécialisés appelés « Matchmover Artist ».

#### 4.2.6 Le tracking planaire

La méthode du tracking planaire a été développée au début des années 2000 pour apporter « une amélioration » au tracking 2D. Effectivement, cette méthode ne s'appuie pas sur une zone de référence rassemblant un petit nombre de pixels cibles pour assurer le suivi de mouvement mais sur des motifs de plus grande dimension. Le repérage du motif au sein de l'image peut aussi bien

se faire lorsque le motif est complètement visible que lorsque le motif n'est que partiellement visible car en partie caché par son environnement. Cela rend le suivi de l'objet plus facile. Le logiciel Mocha est le logiciel le plus connu pour effectuer du tracking planaire. La bibliothèque OpenCV de Python peut également être utilisée pour effectuer le tracking planaire et assurer le suivi d'un objet cible.

### 4.3 Identification

Les technologies d'identification de personnes sont aujourd'hui omniprésentes. Nous en rencontrons tous les jours, avec les systèmes de reconnaissance faciale intégrés à nos téléphones (comme le Face ID d'Apple ([Shen, 2018](#))), ou encore dans les portes automatiques de contrôles frontaliers.

#### 4.3.1 Méthode d'identifications existantes

Il existe aujourd'hui deux types prédominants d'identification de personnes : l'identification biométrique et l'identification par l'apparence [13]. En matière de reconnaissance biométrique, la technique la plus utilisée de nos jours est la reconnaissance faciale, car elle est la plus facilement déployable à grande échelle. Certaines entreprises ont déjà développé des solutions de reconnaissance faciale directement intégrées aux téléphones mobiles. C'est le cas du français Thalès avec le FRP Mobile ([frp](#), ). Un autre type de reconnaissance biométrique est la reconnaissance de la démarche d'une personne. En effet, chaque personne possède une manière distincte de marcher. Cette approche est utilisée par exemple par l'entreprise chinoise Watrix qui aborde le problème de l'identification humaine en caractérisant et en discriminant la façon dont ils marchent ([wat](#), ). L'identification par l'apparence est surtout utile pour les problématiques de tracking sur un flux de vidéos provenant de plusieurs caméras. En effet, pour reconnaître une personne d'une séquence vidéo à une autre, il est plus aisément de sauvegarder l'apparence de la personne dans la première séquence puis de comparer la seconde séquence à la sauvegarde. Cette méthode, couplée à des méthodes de reconnaissance biométrique, est aujourd'hui utilisée dans les systèmes de surveillance de masse afin de détecter les comportements suspects.

#### 4.3.2 Solutions potentielles adaptées à notre étude

La solution la plus adaptée à notre problématique, semble être d'élaborer un algorithme de deep learning et de l'entraîner sur des bases de données de personnes existantes. Certaines de ces bases de données sont accessibles gratuitement sur internet : Viper, PRID11, ETHZ...

**Identification par patch ("patch based")** Parmi toutes les méthodes d'identification existantes, nous avons penché pour l'approche Patch based ([Zhao, 2013](#)) par sa facilité à être mise en place sous Python. Cette approche permet de diviser l'image d'une personne repérée dans la partie "Détection" en multiples carreaux formant une grille de "patches". L'information de chaque patch sera ensuite extraite sous forme d' "information couleur". Cette information sera ensuite comparée à celle des patches des images de la base de données d'images que nous possédons. En comparant chaque patch un à un, il sera alors possible d'obtenir un degré de similitude entre deux images selon la formule :

$$\text{Sim}(\mathbf{x}^{A,p}, \mathbf{x}^{B,q}) = \sum_{m,n} \frac{\text{score}_{knn}(x_{m,n}^{A,p}) \cdot s(x_{m,n}^{A,p}, x_{i,j}^{B,q}) \cdot \text{score}_{knn}(x_{i,j}^{B,q})}{\alpha + |\text{score}_{knn}(x_{m,n}^{A,p}) - \text{score}_{knn}(x_{i,j}^{B,q})|}$$

FIGURE 4 – Degré de similitude

où  $s$  est la ressemblance entre 2 patches, "score" est la "singularité" de chaque patch en question et  $\alpha$  un paramètre contrôlant la différence de similitude entre les 2 patches. En fonction de cette

similitude, nous pourrons alors déterminer l'identité la plus proche de la personne détectée. Les bibliothèques python pour le traitement d'images que nous pouvons utiliser sont : imutils, skimage et openCV.

## 5 Ingénierie-Système

Les méthodes de l'ingénierie-système sont de puissants outils pour concevoir tout type de système. Dans le cadre de notre projet nous avons utilisé ces outils.

### 5.1 Exigences

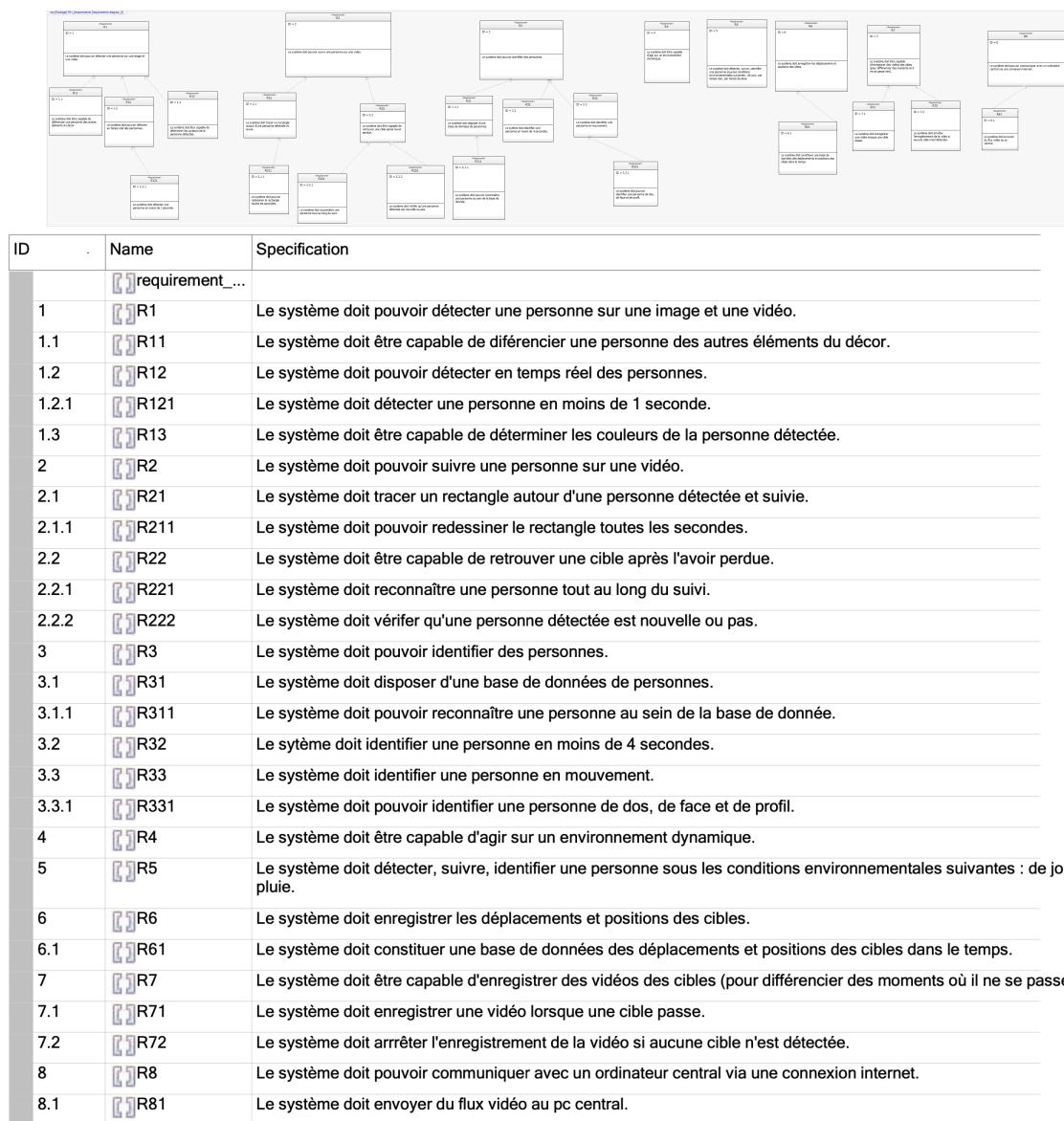


FIGURE 5 – Diagramme et tableau des exigences

La première partie de la pratique de l'ingénierie-système est la détermination des exigences du système. Nous y avons procédé par l'élaboration du diagramme ci-dessus, qui a donné le tableau qui le succède. A l'heure actuelle, ce tableau ne nous paraît pas irréaliste, même s'il pose des problèmes dans l'immédiat à cause de la demande de *streaming*.

### 5.2 Architecture Externe

Nous pouvons passer ensuite à l'architecture externe, en commençant par un *use case diagram*. Ce diagramme montre les relations entre le système, ces fonctions principales, contraintes et acteurs.

Nous avons identifié trois fonctions principales, qui correspondent aux trois grandes parties de notre algorithme : détection, suivi et identification.

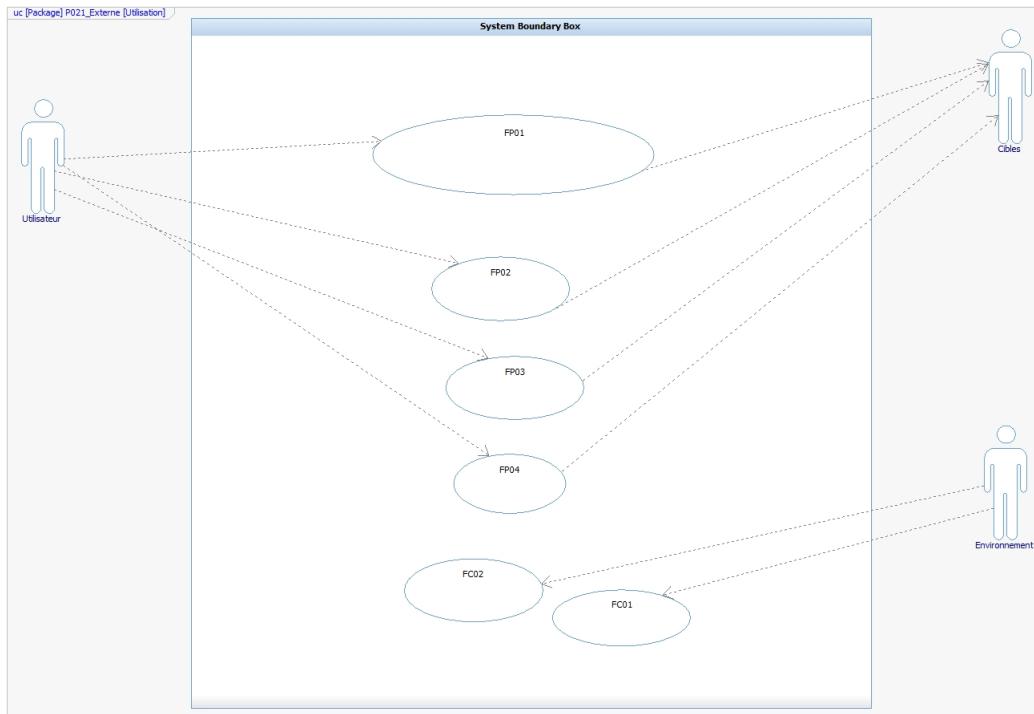


FIGURE 6 – Use case diagram

Une autre façon de représenter les fonctions principales est grâce à un *block definition diagram*, qui se concentre plus sur les flux. Comme notre projet est du monde du numérique, tout flux sera du binaire.

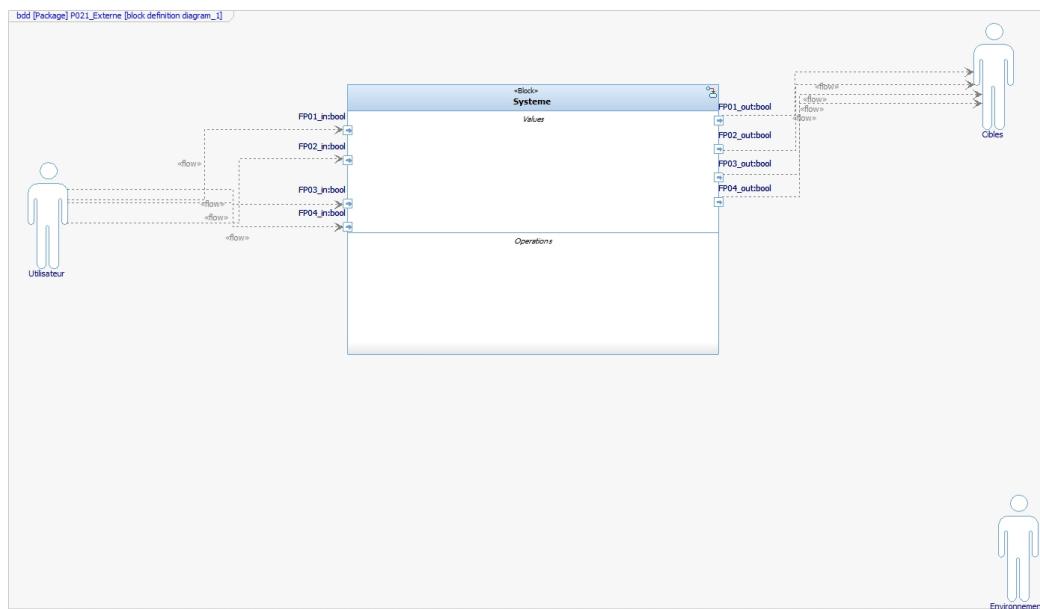


FIGURE 7 – Block definition diagram

Il faut ensuite préciser les fonctions principales et leurs contraintes. La manière dont nous avons défini notre système auparavant fait que le gros du tableau ci-dessous repose sur les fonctions principales,

avec peu de "contraintes".

Pour la détection, nous voulons que l'algorithme différencie entre le décor et les cibles (des personnes), et en absence d'une meilleure unité pour quantifier ce caractère, nous le qualifions ainsi: Oui. Quant aux couleurs, la norme HSV, (*Hue, Saturation, Value*), est une version plus avancée du classique RGB, qui prend en compte la luminosité d'une couleur.

Pour le suivi, il doit pouvoir tracer un rectangle toutes les secondes, donc mettre à jour la position de la cible à une fréquence supérieure à celle-ci. Et il ne doit pas se tromper à plus de 0,05.

Quant à l'identification, la cible doit pouvoir se déplacer à une vitesse de marche rapide, mais nous n'exigeons pas de vitesse de course (en pratique, sauf à très grande distance, c'est dur à réaliser). Elle devrait idéalement être identifiable selon plusieurs angles, pas juste de face.

Enfin, quant aux contraintes, nous nous sommes inspirés des valeurs moyennes en lux pour un ciel couvert et une journée ensoleillée.

Fonction	Désignation	Critère	Niveau	Flexibilité	Exigences
FP01	Détecter	Détection temps réel	Oui	F0	R11
		Détection temps réel	<1 s	F1	R12
		Détermination couleurs	HSV	F2	R13
FP02	Suivre	Tracé rectangle	1/s	F1	R21
		Réidentification	sûr à 95%	F1	R22
FP03	Identifier	Existence BDD	Oui	F0	R31
		Temps identification	<4 s	F1	R32
		Vitesse cible max	5 km/h	F2	R33
		Orientation cible	Dos, face, profil	F2	R331
FP04	Afficher	Images par seconde	12 fps	F1	R21
		Luminosité ambiante	[1 ; 1,2e5] lx	F2	R5
FC01	Etre utilisable en tout temps	Vitesse rotation téléphone	<3,5 rad/s	F2	R4
		Vitesse rotation téléphone	<3,5 rad/s	F2	

FIGURE 8 – Tableau des fonctions principales et de contraintes

### 5.3 Architecture Interne

Maintenant que les interactions avec l'extérieur ont été mieux définies, nous pouvons réfléchir au fonctionnement du système. C'est là que sert le *activity diagram*.

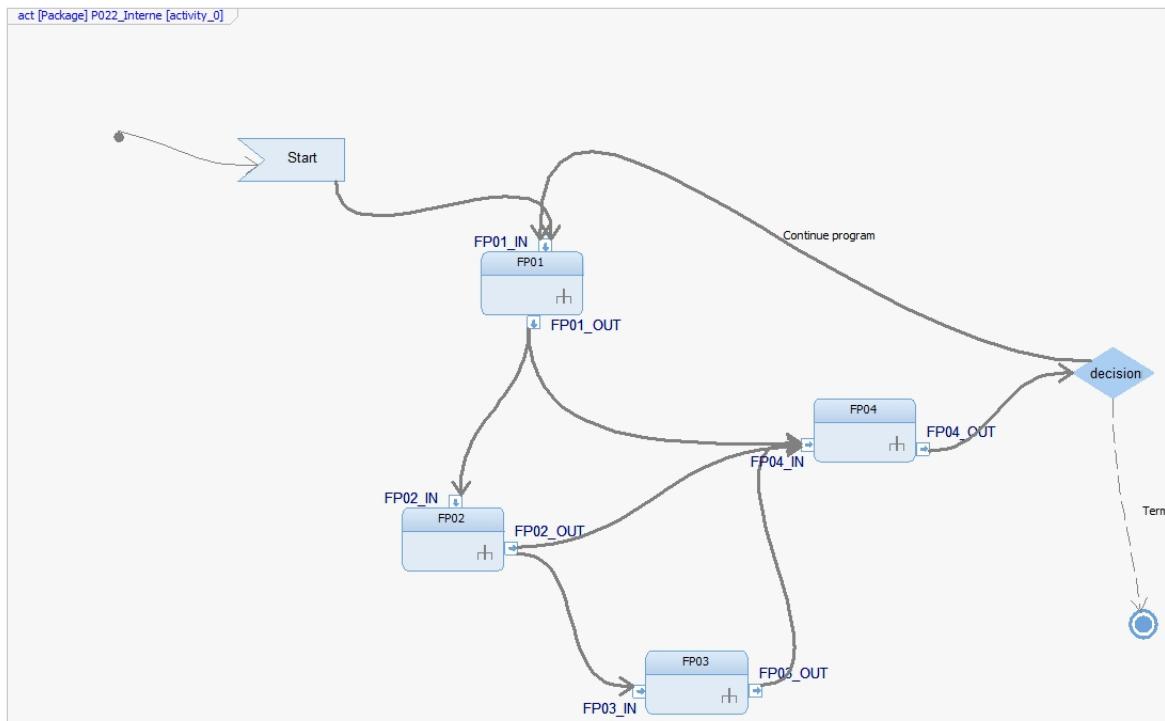


FIGURE 9 – Activity diagram

Tout commence quand l'utilisateur démarre le programme. Le système commence par détecter des personnes. Une fois cette tâche accomplie, il va procéder au suivi des personnes détectées précédemment. Enfin, il cherchera à identifier ces personnes. Tout au long du processus l'affichage joue un rôle important, car il communique les trouvailles de l'algorithme, qui interviennent donc à toutes les étapes précédentes. Cet enchainement peut continuer à l'infini, ou s'arrêter dès que l'utilisateur le souhaite.

## 5.4 Architecture Physique

Nous n'avons pas encore beaucoup avancé sur l'architecture physique. Cependant nous avons quelques hypothèses sur une configuration possible, présentée ci-dessous.

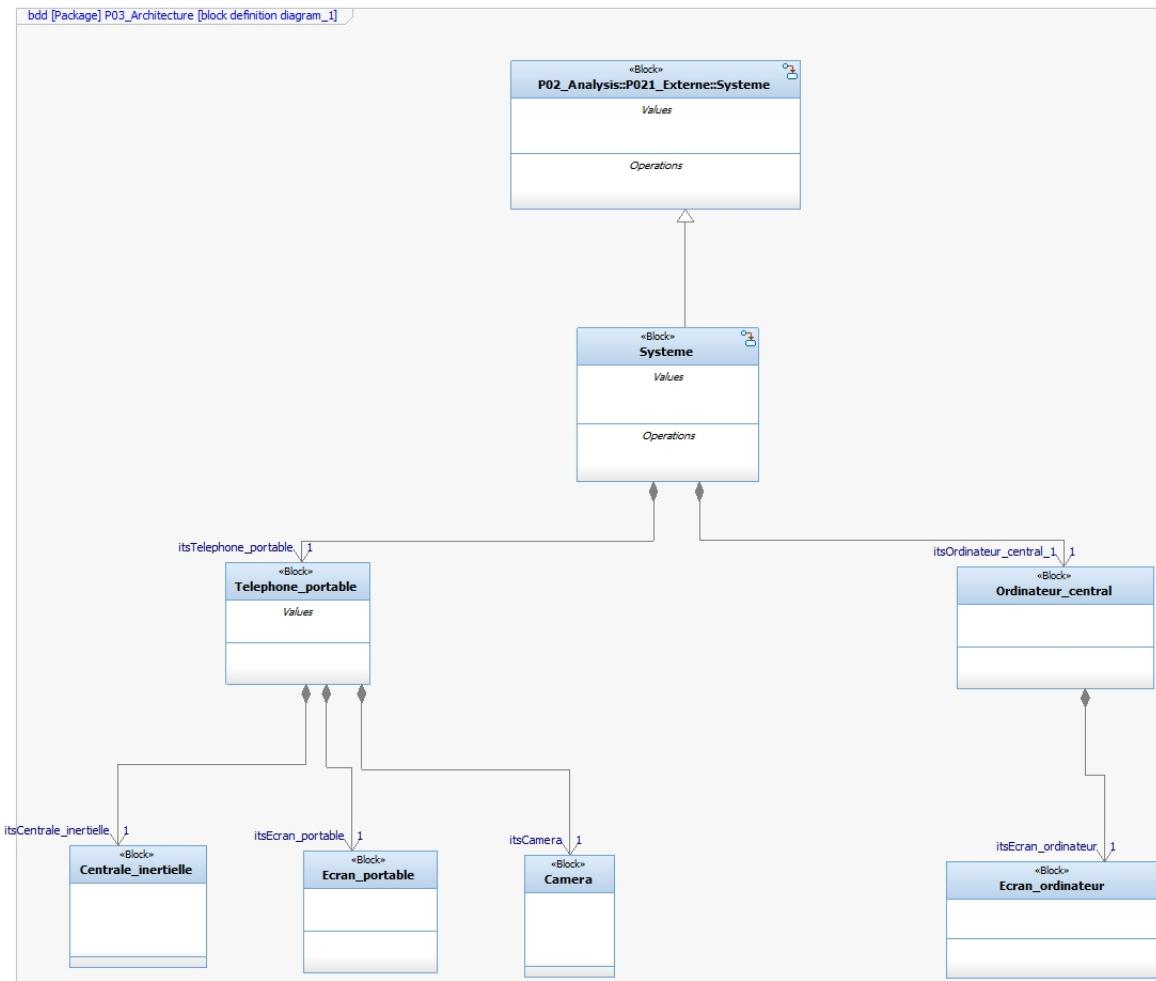


FIGURE 10 – Architecture externe du système

Sur la version finale, nous comptons nous servir simultanément d'un téléphone portable et d'un ordinateur central. Ce dernier servira au minimum à récupérer le flux vidéo de la caméra, et peut-être même à effectuer une grosse partie du calcul. Mais ces considérations restent actuellement au stade d'hypothèse.

## 6 Réseaux et Application

### 6.1 Gestion du réseau

Dans le cadre de notre projet, il nous paraissait crucial de prendre en compte cette partie réseau pour s'assurer que les données du flux vidéo seront transmises de manière fiable et efficace. Si la connexion réseau est instable, cela peut entraîner des interruptions de transmission, une dégradation de la qualité de l'image, voire une perte de données. Une mauvaise gestion du réseau mettrait en danger la performance de notre algorithme.

Par conséquent, il y a eu plusieurs points sur lesquels se pencher. Il fallait d'abord posséder un téléphone, capable d'être reconfiguré si nécessaire. Ensuite, s'assurer de son adaptabilité avec notre programme Python. Enfin, le choix de la méthode de transmission, telle que la connexion Wi-Fi directe, l'utilisation d'un câble USB ou l'utilisation d'un réseau local, était une étape à ne pas négliger. En effet, en fonction de la méthode de transmission choisie, plusieurs facteurs tel que la portée pouvaient fortement évoluer.

La partie réseau des différentes applications que nous avons tenté de développer aura souvent été ce qui nous a fait abandonner telle ou telle option, comme nous le verrons par la suite. En effet, soit c'était l'adaptabilité d'OpenCV avec notre application qui posait soucis lorsqu'il s'agissait de détecter le flux vidéo récolté par le téléphone, soit c'était simplement la réception des données vidéo que l'on ne parvenait pas à effectuer.

Nous avons commencé par expérimenter avec des manipulations d'adresses IP avec les outils de la bibliothèque OpenCV, mais pour des raisons que nous ignorons toujours, cela n'a pas marché. C'est là que nous avons découvert Elgato Camera Hub, une application qui nous a permis de lier un téléphone à un ordinateur portable. Il suffit de télécharger l'application Camera Hub sur un téléphone et sur l'ordinateur qui devra également lancer l'application Python. Dans cette dernière, l'entrée principale de flux vidéo est remplacée par le flux de Camera Hub, c'est-à-dire la vidéo du téléphone portable. Cette association permet donc de se déplacer avec un téléphone portable et de relayer la vue de sa caméra à un ordinateur, où le traitement sera effectué, le système est enfin portable.

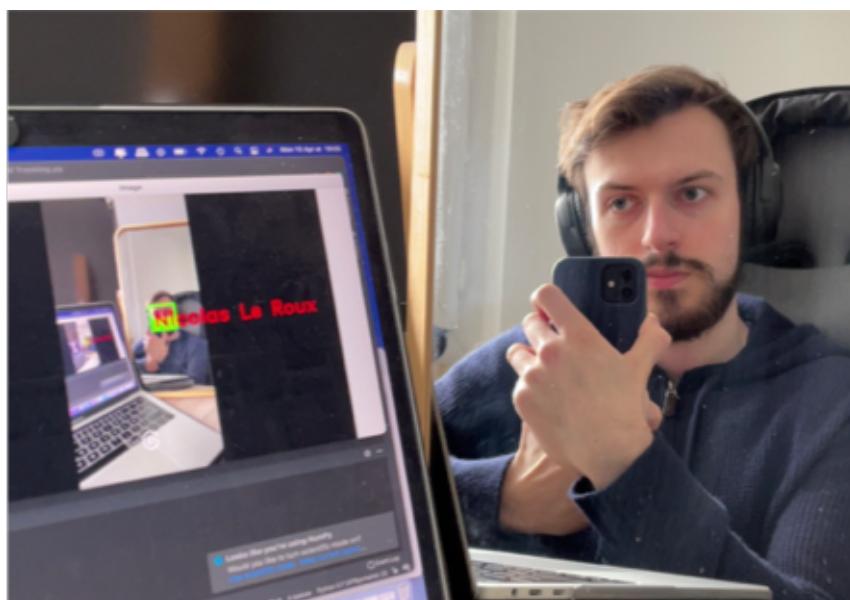


FIGURE 11 – Point de vue du téléphone portable

## 6.2 Application

### 6.2.1 Android Studio

Afin de développer des applications sur Android, Google a créé un environnement de développement : Android Studio. Android Studio est gratuit d'utilisation et propose la création de projet d'application en langage Java ou Kotlin, sur lesquels on reviendra dans la suite. C'était sans aucun doute l'environnement le plus approprié pour nous, car il existe de nombreuses sources d'informations disponibles sur internet pouvant nous instruire à la maîtrise de cet environnement afin de nous aider à développer notre application. Il nécessita l'appropriation de plusieurs notions encore inconnues pour la majorité de l'équipe. Dans un premier temps, nous avions appris la nécessité de comprendre ce qu'est Gradle.

Gradle est un moteur de production, c'est-à-dire que c'est un logiciel ayant pour objectif d'automatiser et d'optimiser toutes les actions de n'importe quel software. Ensuite, on a dû en apprendre plus sur la fonctionnalité des fichiers xml. Dans notre contexte, ces fichiers correspondent à l'interface graphique de notre application. De nombreux cours existent sur internet (sur OpenClassRoom par exemple) permettant de comprendre facilement comment mettre en place une interface graphique de qualité sur Android Studio.

Lorsque nous nous sommes penchés sur l'architecture type des projets sur Android Studio, nous avons découvert une architecture classique dans ce domaine nommée MVE : Modèle-Vue-Contrôleur. Cette architecture sépare nos codes en trois compartiments gérant des tâches différentes. La partie Modèle gère globalement toutes les données qui nécessitent d'être stockée. La partie Vue, concerne comme son nom l'indique toute la partie graphique. C'est dans ce compartiment que sont notamment présent tous les fichiers xml gérant l'affichage lors de l'utilisation de l'application. La partie Contrôleur gère la totalité de la logique de notre code. C'est cette partie que gère l'évolution de l'application en fonction des choix réalisés par le réalisateur au fur et à mesure du temps. Il serait vraiment intéressant et instructif de se forcer à utiliser cette architecture classique dans le monde du développement. Ça ne peut qu'être utile pour notre futur d'ingénieur.

Ces quelques notions ont été étudiées et comprises par les membres de l'équipe occupant cette partie du projet. En fin de premier semestre, plusieurs autres objectifs ont été réalisés sur Android Studio. Premièrement, on savait coder l'accès à la caméra de notre téléphone portable Android. Ensuite, on maîtrisait la librairie Chaqueopy permettant de faire tourner un code python dans notre Kotlin sur Android Studio, de même que nous savions comment importer presque n'importe quel package dont nous avions besoin. On avait un très bon aperçu de notre modèle Vue, alors que nous avions débuté la partie Contrôleur, qui dépend grandement de l'évolution du code Python (celui-ci n'étant pas encore finalisé). Nous avions analysé plusieurs exemples de code de développement d'application sur Android Studio afin d'être à l'aise avec les généralités des applications Android. Ces généralités pourront être appliqués lors du codage final pour la finalisation du projet.

Même si nous avions appris le langage Java lors de ce semestre, nous codions sur Android Studio avec le langage Kotlin. Ce langage de programmation orienté objet a été créé expressément pour le développement d'application sur Android. Il est dérivé de Java. La différence entre les deux est qu'il est largement simplifié en termes de syntaxe : beaucoup d'abréviation ou de modification syntaxiques rendant la lecture plus fluide et intuitive. Sa maîtrise demande cependant une bonne connaissance de Java. Nous maîtrisons assez ce langage pour l'utiliser dans notre projet.

Plusieurs de nos membres se sont instruits sur Android Studio grâce à la plateforme Udacity. Cependant, même avec ces connaissances, nous avons eu beaucoup de mal à développer une application fiable sur Android Studio, permettant de répondre à nos objectifs. La complexité des protocoles de

communication tels que RTSP nous ont posé des soucis. Nous avons donc malheureusement peu à peu chercher d'autres solutions que Android Studio.

### 6.2.2 BeeWare

Un autre environnement de développement très récent nous a fortement intéressé. Il se nomme BeeWare et a été créé par le créateur de Chaquepy. BeeWare permet l'implémentation et l'exécution de code Python sur n'importe quelle plateforme : Windows, macOS, mais surtout Android. Son utilisation semblait relativement simple et en cas de succès elle pouvait nous permettre de créer aisément une application sur Android.

Nous avons aisément pu développer une application sur Windows grâce à BeeWare, nous permettant de répondre à des objectifs très proches de ceux que l'on s'était proposés de faire sous Android.

C'est lorsque nous avons entamé l'implémentation sur Android que des problèmes ont commencés à émerger. Nous recevions beaucoup d'erreurs lorsque nous essayons d'executer l'application sous Android. Le problème étant que nous ne parvenions pas à comprendre ces erreurs et, l'environnement étant encore très récent, peu de documentation était disponible et pertinente sur internet.

### 6.2.3 DroidCam : choix final

DroidCam est une application Android qui permet de transformer son téléphone en une webcam sans fil pour ordinateur. Elle est développée par Dev47Apps et est disponible en téléchargement gratuit sur Google Play.

Une fois que l'on a installé l'application DroidCam sur son téléphone Android et son client sur son ordinateur, on peut utiliser notre téléphone comme une caméra sans fil pour notre ordinateur, et accéder au flux vidéo de notre téléphone sur notre ordinateur via un réseau sans fil ou un câble USB. Nous avons choisi d'utiliser le Wifi comme technique de transmission, car celle-ci permet d'élargir la portée de notre caméra portative sans altérer la qualité du retour vidéo.

Nous pouvons par la même occasion récupérer assez aisément le flux vidéo de plusieurs téléphone Android sur un même ordinateur, ce qui nous a rassuré dans le but d'effectuer du multi-caméras par la suite. A noter qu'il est tout à fait possible d'utiliser des iPhone, DroidCam n'est pas limité aux Android. C'est donc cette méthode que nous avons au final choisi. Elle est facile d'utilisation et performante, ce qui nous a permis de plus nous focaliser sur l'aspect codage sans trop s'inquiéter de l'adaptabilité de notre algorithme final avec DroidCam.

## 7 Traitement de Flux Vidéo

La partie principale de notre application consiste en capter un flux vidéo puis le traiter pour détecter, suivre et identifier des personnes. Après notre étude initiale dans l'état de l'art, nous avions décidé de coder le traitement du flux vidéo en Python. Nous pourrions ensuite nous interroger sur une éventuelle conversion de ce code dans un autre langage. Ces recherches initiales ont également souligné l'importance d'une bibliothèque qui sert de base à tout notre traitement : OpenCV. Il s'agit d'une bibliothèque spécialisée dans le traitement d'images et de vidéos en temps réel, elle est disponible dans de nombreux langages de programmation, dont Python.

### 7.1 Détection

Dans un premier temps nous nous sommes concentrés sur la détection. Une méthode très courante est l'utilisation d'une combinaison d'histogrammes de gradients orientés (HOG en anglais) avec des machines à vecteurs de support (SVM en anglais), cette combinaison HOG + SVM est instanciée en deux lignes sur Python.

Comme nous l'avons expliqué dans l'état de l'art, cette méthode va parcourir une image avec une fenêtre glissante dans laquelle sera appliquée la combinaison HOG + SVM. Elle renvoie les coins des fenêtres rectangulaires dans lesquelles sont détectées des personnes (ainsi que les poids associés).

L'implémentation de ces méthodes repose sur une combinaison d'une boucle while pour l'acquisition du flux vidéo, la modification des images pour simplifier le traitement, (passage en noir et blanc, taille réduite), l'initialisation des outils ci-dessus, le traitement d'une image, puis le traçage des rectangles associés sur l'image initiale affichée à l'écran. Notre expérimentation avec les HOG repose surtout sur la modification des paramètres padding, scale et winStride, pour équilibrer rapidité et précision de détection. Les paramètres les plus conséquents sont scale et winStride, qui contrôlent respectivement la diminution de la taille de l'image (pour coïncider avec les tailles de référence des modèles) et le pas de la fenêtre glissante. Nous avons également exploré l'utilisation d'outils Haar-like pour la détection de personnes et de visages. Ces outils sont plus vieux que les HOG + SVM, mais nous avons eu de bons résultats pour la détection de visages vues de face. Cependant, ils sont dépassés par les HOG + SVM en matière de détection de personne entière.

Pour l'instant les deux modèles souffrent toujours d'un problème de faux positifs et de faux négatifs. Il leur arrive à tracer aléatoirement des rectangles sur les murs, ou de ne pas ou plus détecter une personne légèrement occluse. Nous progressons sur les deux problèmes, mais comme les modèles ne sont pas parfaits, nous ne pourrons certainement pas éliminer les deux problèmes complètement.

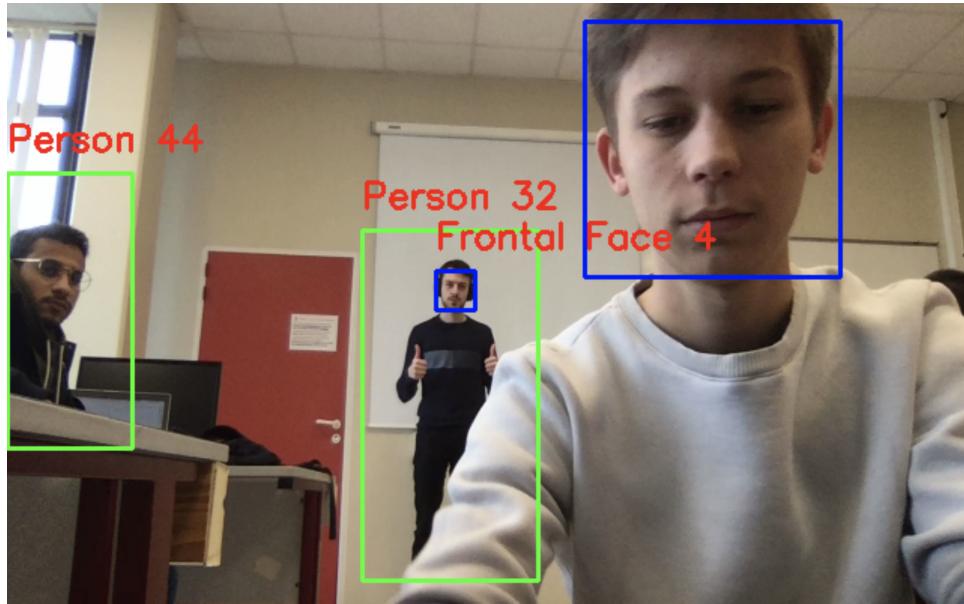


FIGURE 12 – Détection de plusieurs personnes

## 7.2 Suivi

Dans un second temps nous avons abordé le suivi des personnes. Pour rappel, la distinction entre détection et suivi repose principalement sur la reconnaissance d'une cible déjà détectée, et éventuellement la prédiction de son futur déplacement. Il existe de nombreux algorithmes de suivi, dont certains qui incluent également détection. Le plus simple, mais également un qui fonctionne assez bien, est basé sur les centroïdes : il s'agit de prendre un des rectangles contenant une personne, de calculer son centre, stocker l'information, puis dans une image future la comparer à des centroïdes de nouveaux rectangles. Si la distance entre deux centres pris à des instants différents est inférieure à un seuil fixé, on considère qu'il s'agit de la même personne. Ceci fonctionne si la personne se déplace assez lentement devant la fréquence d'acquisition et de traitement des images. En pratique, à une distance suffisante pour observer une personne en entier, une personne qui marche normalement est facile à suivre.

Un deuxième élément graphique qui se rajoute aux rectangles dans cette partie, est l'identifiant attribué au rectangle. Grâce à une classe sur-mesure de suiveurs permettant de stocker les centroïdes, on leur rajoute un identifiant incrémenté par un compteur qui va servir à illustrer quand l'algorithme a bien reconnu une personne. Cet identifiant s'affiche au-dessus de son rectangle associé.

Le suivi comprend ses propres difficultés, comme l'influence de la proximité à la caméra (et donc de la vitesse apparente de déplacement à l'écran) et la confusion potentielle des cibles. En effet, dans notre état actuel, si deux personnes se croisent il est possible qu'elles échangent d'identifiants. Pour y remédier, nous envisageons d'implémenter un prédicteur de Kalman, ce qui devrait empêcher cette confusion et également faciliter la ré-identification en cas de décrochage.

## 7.3 Identification

Dans le contexte du projet, l'identification désigne l'association d'une personne détectée par l'algorithme à une personne dans une base de données. Avant de nous lancer à l'aveugle dans de la théorie que nous ne maîtrisons pas, il était nécessaire de reprendre brièvement l'état de l'art précédent. En complément de ceci, nous avons fait d'autres recherches dessus, qui nous ont mené à la découverte de Face Recognition, une bibliothèque Python codée en C++ et basée sur l'apprentissage profond. Cette bibliothèque permet d'encoder les contours du visage d'une personne (nez, sourcils, yeux, bouche, etc.), de stocker ces informations, et de comparer les paramètres d'un nouveau visage à ceux d'une base de données. Cette bibliothèque est très facile à utiliser, et une seule photo (prise suffisamment de face)

suffit à identifier une personne. Une fois les bibliothèques téléchargées, nous avons modifié du code, trouvé sur le GitHub d'un tutoriel YouTube, pour l'intégrer dans notre code précédent de détection et suivi de personnes. Actuellement, les deux processus n'interagissent pas entre eux, il serait intéressant à l'avenir d'y remédier, mais dans l'immédiat ce n'est pas nécessaire. Les deux sont utiles, le premier (le processus de détection et suivi par HOG+SVM) pour détecter des personnes dont le visage n'est pas face à la caméra, utile pour compter des individus dans une scène, le deuxième pour reconnaître ces individus. Comme Face Recognition nécessite une vue de face du visage, il ne sera applicable que dans quelques cas, ou avec des caméras bien placées.

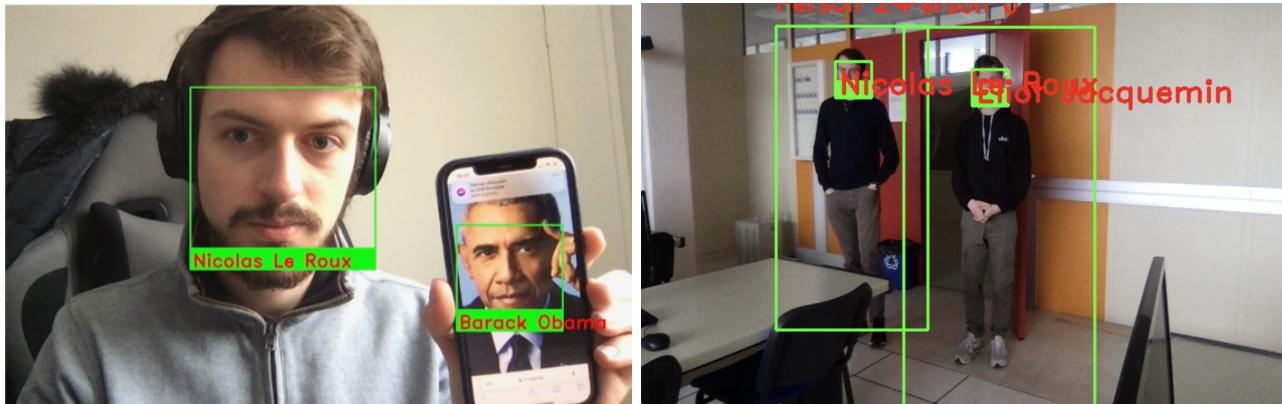


FIGURE 13 – Identification de plusieurs individus

Pour construire notre base de données de visages, nous avons d'abord testé avec nos propres visages, puis avec des photos trouvées sur Google de différentes célébrités. C'est là que nous avons réalisé le dangereux potentiel de cet outil, nous pourrions très facilement prendre des photos de parfaits inconnus sur les réseaux sociaux et les intégrer à notre algorithme. Comme cela est illégal, nous avons préféré envoyer un sondage sur les groupes Messenger des promotions de l'ENSTA Bretagne, afin d'obtenir les consentements de nos camarades. Ainsi, nous avons constitué une base de visages conséquente, qui nous pourrons utiliser lors de la démonstration dans le hall d'honneur.

## 7.4 Bilan

Au final, les deux plus grandes difficultés dans le traitement du flux vidéo sont : la multitude de possibilités, et la spécificité de notre cas d'application. La première peut paraître comme un avantage, car il existe beaucoup de documentation sur le sujet, mais cette surabondance nous complique le choix d'une méthode, surtout quand les avis sont partagés sur leurs efficacités. La deuxième est le simple fait qu'une méthode qui fonctionne parfaitement dans un cadre très précis n'est pas forcément adaptée à nos cas d'études. On trouve des multitudes d'algorithmes qui marchent sur des cibles lentes à grande distance, mais beaucoup moins sur des cibles proches et facilement occluses comme dans un bâtiment. La solution : tester, analyser et améliorer.

## 8 Méthode Agile

La méthode Agile est une façon de penser la gestion d'un projet informatique, dans lequel le lien entre les différentes parties prenantes du projet est la priorité numéro une. Agir agile signifie notamment prôner la méthodologie scrum qui propose d'avancer pas-à-pas de manière claire et concise. Pour ce faire, la collaboration avec le client ( ce qui correspond à nos encadrants dans le cadre de notre projet ) est primordiale. Il est favorable de réaliser des test récurrents afin de reconnaître et supprimer chaque erreur le plus rapidement possible.

Enfin la méthode Agile prône une flexibilité face aux objectifs du projet définis en début de cycle, mais également l'adaptabilité de l'équipe face aux imprévus en milieu de projet.

Dans notre cadre, la méthode Agile peut s'appliquer en appliquant différentes techniques de gestion simple et pourtant rendant le suivi de notre projet significativement plus agréable. Cette méthode est privilégié pour la majeur partie des projets en entreprise dans le monde entier. Par conséquent, se l'approprier pour ce projet semble bénéfique pour notre futur en tant qu'ingénieur.

### 8.1 Diagramme de Gantt

Un diagramme de Gantt est un outil très utile pour une gestion de projet informatique. Ce diagramme est un moyen simple et efficace de déterminer les tâches à réaliser lors d'un projet, en précisant l'intervalle de temps prévu pour le réaliser. Il permet par exemple de rendre claire la potentielle superposition de plusieurs tâches pour un même interval de temps.

#### 8.1.1 Premier diagramme de Gant

ATR portable - Diagramme de Gantt

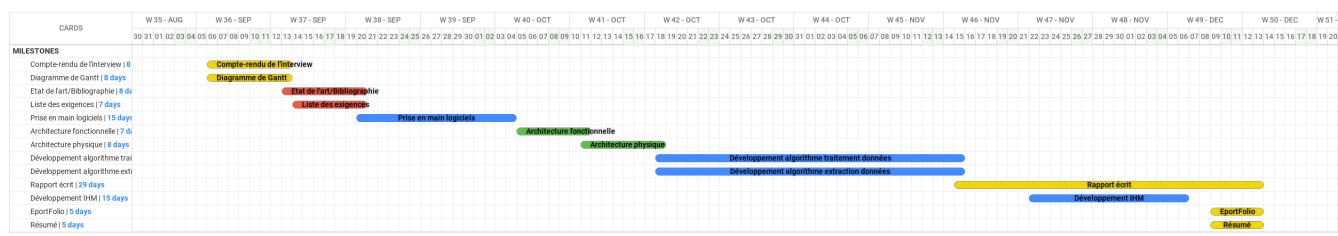


FIGURE 14 – Diagramme de Gantt, semestre 3

#### 8.1.2 Deuxième diagramme de Gant

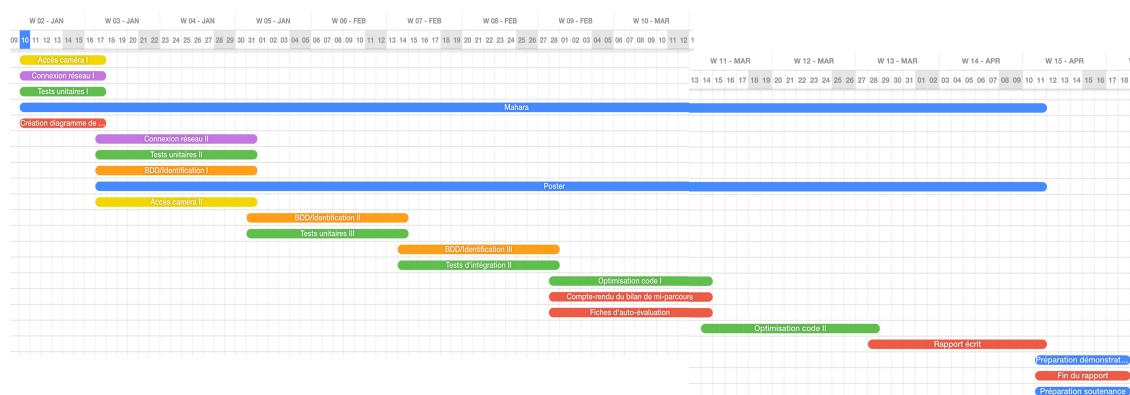


FIGURE 15 – Diagramme de Gantt, semestre 4

## 8.2 Evolution de la gestion de notre projet

Au début de notre projet, nous avions eu du mal à utiliser le diagramme de Gantt et à le comprendre. Cependant, au fil du temps, nous avons compris que le diagramme de Gantt était essentiel pour la bonne gestion de notre projet. Voici quelques raisons pour lesquelles cet outil nous a aidé durant le projet.

Premièrement, le diagramme de Gantt nous a permis d'organiser les différentes tâches du projet et de les répartir dans le temps. Nous avons ainsi pu mieux planifier notre travail, en évitant les retards. Cette planification nous a également permis d'anticiper les risques et les problèmes potentiels et de mettre en place des solutions adaptées pour y faire face.

Ensuite, il nous a aidé à mieux communiquer et à coordonner nos actions. Grâce à cet outil, nous avons pu voir plus facilement l'avancement des différentes tâches et les dépendances entre elles. Cela nous a permis de mieux collaborer et de travailler plus efficacement en équipe.

Enfin, le diagramme de Gantt nous a aidé à mieux gérer les différents objectifs du projet. Nous avons ainsi pu voir plus clairement les priorités et les étapes clés du projet, ce qui nous a permis de nous concentrer sur les tâches les plus importantes en premier.

Même si nous avons connu des difficultés au début pour bien utiliser le diagramme de Gantt, nous avons finalement compris qu'il était utile pour la bonne gestion de notre projet. Nous avons ainsi utilisé cette méthode en la combinant avec la méthode Agile. En effet, la méthode Agile nous a également aidé à mieux organiser notre travail en nous permettant de travailler de manière itérative et de nous adapter rapidement aux changements.

En conclusion, le diagramme de Gantt nous a permis d'optimiser l'organisation, la communication et la gestion des différents objectifs du projet. Même s'il reste des améliorations à apporter, nous sommes convaincus d'avoir utilisé cette méthode de manière efficace tout au long du projet. Nous avons ainsi pu mener à bien notre projet avec succès.

## 9 Tests unitaires, d'intégration et validation

### 9.1 Tests unitaires

La validation d'un code passe par plusieurs étapes, dont la première est celle des tests unitaires. Grâce à Python, ces tests sont faciles à réaliser.

Une première série de tests a été réalisée au semestre 3, et ils traitaient de la classe tracker, en étudiant l'initialisation de cette classe et la mise à jour d'un tracker. Trois tests mineurs, tous validés, vérifient qu'un nouveau tracker a bien le bon nom, un compteur à zéro et un dictionnaire vide. Ensuite, un test vérifie la mise à jour du tracker à travers la méthode update. Il détermine si l'entrée correspondante dans le dictionnaire contient bien le centre des points donnés, ainsi que l'enregistrement du temps auquel a été réalisé la mise à jour. Ce test, lui aussi, est validé.

Suite aux développements du semestre 4, il était intéressant de préparer d'autres tests unitaires. Nous avons décidé d'en rajouter deux supplémentaires. Le premier s'intéresse à la gestion de mémoire de la classe tracker. Ce test vérifie trois choses différentes : que rajouter un rectangle augmente le compteur, que si des éléments ne sont pas mis à jour au bout d'une minute, le compteur diminue, et que si le dernier élément n'est pas mis à jour, le dictionnaire des éléments est remis à zéro. Ceci couvre toutes les situations possibles avec le tracker, et le test est validé. Le deuxième test concerne l'identification : on prend une image (pour le test, c'est une photo d'un membre du groupe) et on la compare à une base de données limitée (contenant une photo de référence du membre, et celle d'une autre personne). On vérifie que le membre du groupe est reconnu dans la nouvelle image. Ce test n'est pas validé à chaque relance, il semble que Face Recognition ne réussit pas systématiquement. Cela dit, en pratique, comme on relance ce processus continuellement dans l'algorithme principal, les échecs passent largement inaperçus.

### 9.2 Tests d'intégration

Les deux sous-systèmes physiques de notre projet sont le téléphone qui sert de caméra, et l'ordinateur portable qui reçoit le flux vidéo et s'occupe du traitement. L'intégration des deux sous-systèmes dans l'ensemble a été validé dans les tests de validation ci-dessous.

### 9.3 Tests de validation fonctionnelle et écarts observés

Les tests de validation fonctionnelle suivants ont été effectués par Eliot Jacquemin et Nicolas Le Roux :

**La reconnaissance de visages :** Cela implique de vérifier si le système est capable de reconnaître les visages dans les images fournies, ainsi que la précision de la reconnaissance, et les limites de celle-ci.

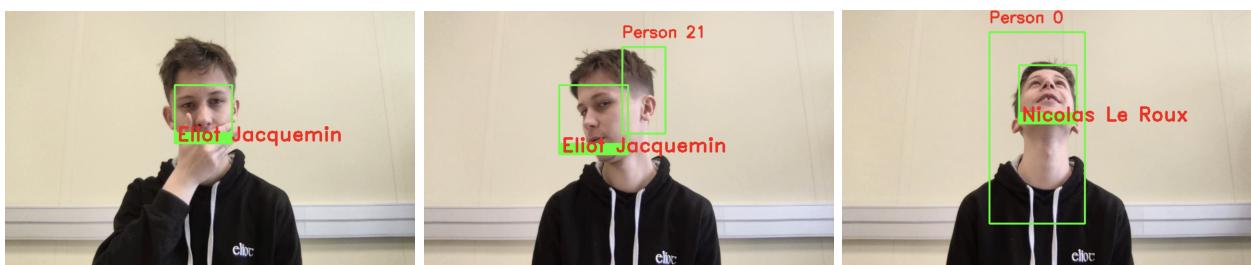


FIGURE 16 – Tests de l'orientation du visage pour identification

Nous avons pu constater que, de face, Face Recognition arrive à reconnaître un visage partiellement occulté dans certains cas. La bouche semble ne pas avoir d'influence sur la performance de l'algorithme,

tandis que l'occultation d'un oeil ou des sourcils, pose problème. Un résultat surprenant de ces tests: relever la tête semble confondre l'algorithme, ici il a confondu Eliot et Nicolas.

**La vitesse de traitement :** Il s'agit de vérifier si le système peut traiter rapidement les images pour fournir des résultats en temps réel. Dans le cahier des charges initial, notre objectif était de détecter un humain en moins d'une seconde, et identifier une personne connue en moins de quatre secondes.



FIGURE 17 – Tests de vitesse de traitement

Ces deux conditions sont largement validées, sauf en cas d'éloignement trop important de la caméra (empêchant l'identification), ou de rapprochement trop important (empêchant une bonne détection du corps entier). Il est donc clair que pour obtenir un bon équilibre entre détection/suivi et identification, il existe une fine plage de distances à la caméra qui satisfait les deux cas. D'après nos mesures, cette plage correspond à [2,5m; 3,0m] d'éloignement d'une caméra bien orientée. Plus loin, et on détectera facilement un corps, mais le visage sera difficilement identifié, plus proche, et la détection du corps est impactée. Un moyen de contourner cette limitation d'identification, est de faire un zoom avec la caméra. En effet, nos tests montrent que pour des zooms standards de téléphone, c'est l'espace à l'écran pris par la tête qui compte, pas la distance.

**La stabilité de la connexion :** Il s'agit de vérifier quel impact la distance a sur la stabilité de la connexion. Pour cela, notre sujet, Eliot, a commencé au deuxième étage du bâtiment E de l'ENSTA Bretagne, et il a progressivement descendu les étages, et est allé sur le parking devant la place d'armes.



FIGURE 18 – Tests de la stabilité de connexion

A quelques occasions lors de l'expérience, la connexion a disparu momentanément, mais elle est revenue toute seule (dans les limites de la portée maximale) à chaque fois, sans devoir se rapprocher de l'ordinateur. En fin de compte, la connexion s'est arrêtée à la limite entre le parking et la place d'armes, où nous supposons que le réseau Wifi ne fonctionne plus de façon optimale. Une fois Eliot re-rentré dans le bâtiment, la connexion est revenue. Tout cela montre : que la portée est limitée par la qualité de la wifi, que la connexion peut se perdre et se rétablir sans rapprochement de l'ordinateur, et

que les performances de l'algorithme (en cas de connexion stable) sont indépendantes de la distance à l'ordinateur.

En conclusion, la version actuelle du projet fonctionne très bien entant que système de surveillance mobile, capable de surveiller un immeuble à plusieurs étages, et ses environs.

## 10 Impact environnemental

Il est important d'évaluer au préalable le potentiel impact environnemental de notre projet. Les seuls appareils que nous utilisons sont un ordinateur et des téléphones portables. L'équipe est consciente de l'existence de l'impact environnemental du numérique ainsi que de son importance. En effet, plusieurs caractéristiques participent dans une certaine mesure à l'émission de gaz polluants, et cela tout au long du cycle de vie de chaque appareil numérique.

Premièrement, intéressons-nous à la phase de conception. La conception de matériel numérique (ordinateur et téléphone notamment) nécessite l'extraction et l'utilisation de plusieurs composants chimiques comportant pour certains de fortes contraintes environnementales. Par exemple, le plastique ou la silice sont des matériaux représentant la majeure partie de la conception d'un ordinateur, plus de 20% chacun. Or, on est conscient des responsabilités majeures qu'ont ces matières dans la pollution. Une fois ces ordinateurs devenus des déchets électroniques, leur traitement devient par conséquent une complication pour la santé de l'environnement. À notre échelle, les ordinateurs et téléphones que nous utilisons existaient avant le début de notre projet et continueront d'exister après. De ce fait, l'impact environnemental de notre projet semble assez minime. Nous avons très peu de marge pour limiter cet impact sachant que nous n'avons pas le choix du type de matériel numérique à utiliser de préférence, ceux-ci possédant à peu près tous la même composition.

Vient ensuite la question des consommations électrique et énergétique. D'un point de vue énergétique, notre projet ne demande pas plus d'énergie que pour n'importe quelle autre activité sur un appareil numérique. Ainsi cet aspect ne semble pas très pertinent pour notre projet : nos émissions de  $CO_2$  dues à ce projet ne sortent pas de l'ordinaire. D'un point de vue électrique, notre consommation dépend largement de la puissance utilisée pour faire fonctionner notre application. En d'autres termes, plus nos exigences en termes de rapidité, fluidité, et d'efficacité du traitement d'images, mais également de la détection, du suivi et de l'identification de plusieurs visages seront élevées, plus notre consommation augmentera. Il semblera donc pertinent d'être conscient qu'une élévation du niveau de nos objectifs entraînera un impact environnemental relativement plus important de notre projet. Cependant cela ne doit pas faire baisser nos ambitions de façon abusive, mais simplement nous sensibiliser afin de ne pas dépasser certaines limites inutiles.

Ainsi, l'équipe considère que les impacts environnementaux résultant de notre projet restent relativement faibles. Toutefois il reste toujours pertinent d'être informé de certains aspects du projet qui peuvent facilement être adaptés grâce à une méthode d'écoconception, tout en gardant les mêmes objectifs.

## 11 Conclusion

Dans le cadre de notre projet visant à réaliser de la reconnaissance en streaming à partir d'un téléphone Android sur notre ordinateur, nous avons réussi la majorité de nos objectifs et obtenu des résultats satisfaisants en termes d'efficacité et de performance. En effet, nous avons réussi à mettre en place une solution fiable pour récupérer le flux vidéo de notre téléphone Android sur notre ordinateur. Nous avons utilisé l'application DroidCam pour transformer notre téléphone en une webcam sans fil et ainsi diffuser le flux vidéo en temps réel. Cette solution s'est avérée efficace et nous a permis d'appliquer notre algorithme de reconnaissance de visages directement sur le flux vidéo en streaming.

Cependant, nous avons échoué dans le développement d'une application Android permettant d'utiliser notre algorithme de reconnaissance de visages directement sur le téléphone. Cette difficulté était principalement liée à la partie réseau, où il aurait été nécessaire de mettre en place une communication fiable et sécurisée entre le téléphone et l'ordinateur. Nous avons donc privilégié une solution qui permettait de réaliser la reconnaissance directement sur notre ordinateur.

Enfin, la méthode de multi-caméra que nous avons testée s'est avérée être une réussite, bien que la performance ait diminué lorsque nous avons ajouté plusieurs téléphones en streaming simultanément. Cette solution pourrait être utile dans le cadre de projets nécessitant la surveillance de plusieurs angles de vue simultanément. Ce projet nous a permis de mettre en pratique et développer nos connaissances en reconnaissance de visages, mais aussi en réseaux de communication et en développement d'applications mobiles. Nous avons pu constater que la partie réseau était un aspect crucial dans le cadre de notre projet, et que son développement pouvait s'avérer complexe. Nous aurions pu continuer à développer notre application Android pour utiliser notre algorithme de reconnaissance de visages directement sur le téléphone avec plus de temps et de ressources, mais nous sommes malgré tout satisfaits des résultats que nous avons obtenus avec la solution de streaming via DroidCam.

Au final ce projet a des applications pratiques dans le domaine de la sécurité et de la surveillance, où il est nécessaire de surveiller des zones en temps réel à distance. Nous espérons que nos résultats pourront inspirer d'autres projets similaires et ainsi contribuer à l'amélioration de la sécurité et de la surveillance dans différents domaines.

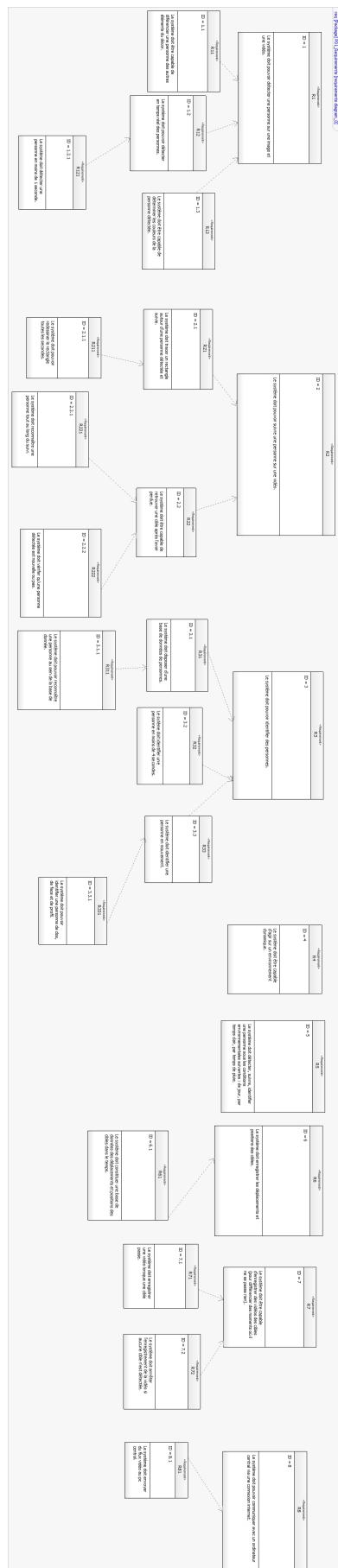
Nous tenons sincèrement à remercier nos encadrants Mr.Toumi et Mr.Cexus pour leur précieuse aide et soutien tout au long du projet.

## 12 Bibliographie

### Références

- Data analytics post. svm. 9
- Opencv. boost class reference. 9
- Reconnaissance faciale : solutions pour 2020 i thales (thalesgroup.com). 12
- (watrix.ai). 12
- (2020). Create a local binary pattern of an image using opencv-python. 8
- (2021). Ibm cloud education. 9
- DATALYA (2018). Object detection with sliding window algorithm. 8
- DUPOIRON (2018). Nouveaux paradigmes de capture d'images et traitements associés pour futurs soc en noeuds cmos nanométriques. *Université Grenoble Alpes*, pages 97–106. 7
- GARCÍA, F. I. Introduction to support vector machines. 9
- INFOWORLD (2022). What is tensorflow? the machine learning library explained. 9
- KLINGLER. Deep learning for person re-indentification. 8
- NIKISHAEV (2018). Shape context descriptor and fast characters recognition. 8
- PALOWSKI, J. (2019). Détection faciale et reconnaissance faciale avec opencv4 en c++. *Creative tech, devoteam*. 7
- PETITPA. Détection d'object par le classificateur de haar. *inconnu*. 7
- PIETIKÄINEN (2010). Local binary patterns. *Scholarpedia*. 8
- ROSEBROCK (2015). Sliding windows for object detection with python and opencv. 8
- SCIENCE, O. D. (2019). The beginners guide for video processing with opencv. 9
- SHEN, Y. (2018). Faceid-gan: Learning a symmetry three-player gan for identity-preserving face synthesis. 12
- SOUDED, S. (2013). *People Detection, Tracking and Re-identification Through a Video Camera Network*. Thèse de doctorat, Ecole doctorale STIC, Université de Nice - Sophia Antipolis, Nice, France. 8
- TREFNY, M. (2010). Extended set of local binary patterns for rapid object detection. *Czech Society for Cybernetics and Informatics*. 7
- VIOLA et JONES (2001). Rapid object detection using a boosted cascade of simple features. 7
- WIKIPEDIA (2018). Histogramme de gradient orienté. *OWLapps*. 7
- ZHAO, R. (2013). Unsupervised salience learning for person re-identification. 12

Annexes : diagramme des exigences et diagramme de Gantt agrandis



### ATR portable - Diagramme de Gantt

