



## Travail de Bachelor

Filière télécommunications, orientation internet et communication

---

### DeepCounter

Deep Learning to Detect and Count People in Video Sequences

---

### Rapport

Version 8.0

#### Rédigé par

Patrick Audriaz

#### Superviseurs

Jean Hennebert, Houda Chabbi

#### Experts

Julien Bégard, Emeka Mosanya

Fribourg, 9 juillet 2019

Copyright © 2019 Patrick Audriaz  
Haute école d'ingénierie et d'architecture Fribourg  
LaTeX-template-for-Word by sebnil on GitHub

# DOCUMENT

---

## 1 Métadonnées

---

**Auteur :** Patrick Audriaz

**Superviseurs :** Jean Hennebert, Houda Chabbi

**Experts :** Julien Bégard, Emeka Mosanya

**Collaborateurs :** Flavia Pittet, Matthieu Jourdan

**Date d'édition :** 11 juillet 2019

**Version :** 8.0

## 2 Organisation et liens

---

Tous les documents concernant le projet sont déposés sur la forge à l'adresse :

*<https://forge.hefr.ch/projects/deepcounter>*

Tous les fichiers du projet sont déposés sur le dépôt Git disponible à l'adresse suivante :

*<https://gitlab.forge.hefr.ch/patrick.audriaz/tb-audriaz>*

## 3 Table des versions

---

Version	Date	Remarque
1.0	22.5.2019	Création du document
2.0	31.5.2019	Analyse du projet existant
3.0	10.6.2019	Analyse technologique
4.0	20.6.2019	Conception
5.0	28.6.2019	Réalisation
6.0	4.7.2019	Analyse économique
7.0	9.7.2019	Tests
8.0	11.7.2019	Finalisation



# TABLE DES MATIÈRES

---

<b>Document .....</b>	<b>3</b>
1    Métadonnées .....	3
2    Organisation et liens .....	3
3    Table des versions.....	3
<b>I.    Introduction .....</b>	<b>11</b>
1    Préface .....	11
2    Contexte .....	11
3    Problématique.....	12
4    Objectif du projet .....	12
5    Contraintes .....	12
6    Plan .....	13
7    Activités.....	13
7.1    Activités principales.....	13
7.2    Activités optionnelles.....	15
<b>II.   Analyse du projet existant .....</b>	<b>16</b>
Introduction.....	16
8    Vue d'ensemble.....	16
8.1    City Pulse .....	16
8.2    Prototype .....	17
8.2.1    Roadmap (1) .....	17
8.2.2    Etat actuel du prototype .....	18
8.3    Architecture .....	20
8.4    DeepCounter dans CityPulse.....	21
9    Base de données.....	22
9.1    BBData .....	22
9.1.1    Fonctionnement .....	23
9.1.2    Stack technologique .....	26
Synthèse.....	26
<b>III.  Analyse technologique .....</b>	<b>27</b>

Introduction.....	27
<b>10 Fiabilité.....</b>	<b>27</b>
<b>11 Solutions hardware .....</b>	<b>28</b>
11.1 Traitement distant.....	29
11.1.1 Architecture et hardware .....	29
11.2 Traitement intégré.....	30
11.2.1 Architecture .....	30
11.2.2 Hardware.....	31
11.3 Choix pour ce projet .....	32
<b>12 Streaming depuis le RPI.....</b>	<b>32</b>
12.1 Systèmes de streaming.....	33
12.1.1 Netcat.....	33
12.1.2 GStreamer .....	34
12.1.3 MJPG-Streamer .....	34
12.1.4 Motion .....	34
12.1.5 VLC.....	35
12.1.6 UV4L.....	35
12.2 Comparaison des solutions.....	36
12.3 Encodage du flux vidéo.....	36
<b>13 Détection et comptage.....</b>	<b>38</b>
13.1 Ressources .....	38
13.2 Computer vision.....	38
13.3 Deep Learning (6).....	40
13.4 Convolutional Neural Network (CNN) (6) .....	41
13.5 Modèles pré-entraînés .....	43
13.6 Architectures de CNN (6).....	43
13.7 Object Detection .....	45
13.7.1 Méthodes anté-Deep Learning.....	46
13.7.2 Region-Based Object detection (R-CNN, R-FCN).....	46
13.7.3 Single Shot Object Detection (SSD, Focal Loss, YOLO) .....	47
13.7.4 YOLO (You Only Look Once) (13) .....	49
13.7.5 Résumé.....	51
13.7.6 Comparaison des performances .....	51
13.8 Object Tracking (19) .....	54
13.8.1 Multiple Object Tracking.....	54

13.8.2	Euclidean Distance Algorithm .....	55
13.8.3	IoU (Intersection over Union) .....	56
13.8.4	Correlation Filters.....	56
13.8.5	SORT (Simple Online and Realtime Tracking) (20) .....	56
13.8.6	Deep SORT (21) .....	57
13.8.7	Comparaison des performances .....	58
13.9	Système de comptage.....	59
13.10	Datasets d'entraînement .....	60
13.10.1	COCO .....	60
13.10.2	Pascal VOC.....	60
13.10.3	ImageNet Bounding Boxes.....	60
<b>14</b>	<b>Problèmes potentiels.....</b>	<b>61</b>
	Synthèse.....	62
<b>IV.</b>	<b>Analyse économique .....</b>	<b>63</b>
	Introduction.....	63
<b>15</b>	<b>Réglementation.....</b>	<b>63</b>
15.1	Protection des données .....	64
15.2	Cadre légal suisse.....	65
15.2.1	La LPD et la vidéosurveillance .....	65
15.2.2	Le cas DeepCounter .....	66
15.3	Europe (RGPD).....	68
<b>16</b>	<b>Étude de marché.....</b>	<b>69</b>
16.1	Offre .....	69
16.1.1	Proposition de valeur .....	69
16.2	Demande .....	70
16.2.1	Ciblage de la clientèle et cas d'utilisation.....	70
16.3	Concurrence .....	72
16.3.1	Morphean .....	72
16.3.2	XOVIS.....	73
16.4	Ambitions industrielles .....	73
16.4.1	Traitemet distant .....	73
16.4.2	Traitemet intégré (Edge Computing) .....	74
16.4.3	Coûts .....	74
16.4.4	Comparaison avec une offre concurrente .....	76
16.5	Potentiel commercial .....	77

Synthèse.....	78
<b>V. Conception .....</b>	<b>79</b>
Introduction.....	79
<b>17 Installations physiques .....</b>	<b>79</b>
17.1 Boîtier Raspberry Pi.....	79
17.2 Positionnement du boîtier .....	80
<b>18 Diagramme flux de donnée .....</b>	<b>82</b>
18.1 Détection sur chaque frame .....	82
18.2 Détection chaque n frame .....	83
<b>19 Dataset de test.....</b>	<b>84</b>
19.1 Scénarios problématiques.....	84
<b>20 Environnement.....</b>	<b>85</b>
20.1 Jupyter Notebook (6).....	85
20.2 OpenCV .....	85
20.2.1 Python / C++ .....	86
<b>21 Object Detection.....</b>	<b>86</b>
21.1 OpenCV DNN Module.....	86
<b>22 Tracking.....</b>	<b>87</b>
22.1 OpenCV MultiTracker.....	88
22.2 Dlib Correlation Filters .....	88
22.3 Centroid Tracking.....	88
<b>23 Algorithme de comptage.....</b>	<b>88</b>
23.1 Approche basée sur la direction de déplacement.....	89
23.2 Approche basée sur des zones.....	89
23.3 Choix de l'algorithme de comptage .....	90
Synthèse.....	90
<b>VI. Réalisation .....</b>	<b>91</b>
Introduction.....	91
<b>24 Raspberry Pi.....</b>	<b>91</b>
24.1 Hardware.....	91
24.2 Installations préalables .....	91
24.2.1 UV4L (stream) .....	92
<b>25 Récupérer le flux vidéo .....</b>	<b>93</b>

25.1	OpenCV VideoCapture Class .....	93
25.2	Lire le flux vidéo .....	93
<b>26</b>	<b>Object Detection .....</b>	<b>94</b>
26.1	OpenCV DNN Module.....	95
26.2	Chargement du modèle.....	96
26.2.1	Darknet (YOLO).....	96
26.2.2	Tensorflow .....	96
26.3	Faire la prédiction (inférence) .....	96
26.4	Traitements des prédictions .....	98
26.4.1	SSD .....	98
26.4.2	YOLO .....	98
26.4.3	Résultat des détections.....	99
<b>27</b>	<b>Object tracking .....</b>	<b>100</b>
27.1	Objet "suivable" .....	100
27.2	Tracking.....	101
27.2.1	Résultats du tracking .....	102
<b>28</b>	<b>Algorithme de comptage .....</b>	<b>103</b>
<b>29</b>	<b>Détection chaque n frame .....</b>	<b>104</b>
<b>30</b>	<b>Envoie sur BBData.....</b>	<b>105</b>
	Synthèse.....	106
<b>VII.</b>	<b>Tests et VALIDATION.....</b>	<b>107</b>
	Introduction.....	107
<b>31</b>	<b>Vitesse .....</b>	<b>107</b>
31.1	OpenCV vs Darknet sur CPU .....	108
31.2	FPS théoriques sur CPU Intel i7-4770 .....	108
31.3	Temps d'inférence .....	110
<b>32</b>	<b>Fiabilité.....</b>	<b>110</b>
32.1	Fiabilité de la détection d'objets .....	111
32.2	Fiabilité et comparaison des modèles .....	112
32.2.1	Points de vue de face .....	113
32.2.2	Points de vue de côté .....	115
32.2.3	Redressage du cadre du point de vue de côté .....	117
32.3	Influence du nombre de FPS sur la fiabilité.....	119
32.4	Récapitulatif des résultats des tests.....	121

32.5	Scénarios spécifiques .....	121
32.5.1	Point de vue de face.....	122
32.5.2	Point de vue de côté (redressé) .....	122
32.5.3	Constatations .....	123
	Synthèse.....	123
<b>VIII.</b>	<b>Conclusions .....</b>	<b>125</b>
<b>33</b>	<b>Conclusion du projet.....</b>	<b>125</b>
33.1	Validation des objectifs.....	125
33.2	Problèmes rencontrés et solutions apportées .....	126
33.3	Perspectives futures .....	127
33.4	Remerciements .....	128
33.5	Conclusion du projet.....	128
33.6	Conclusion personnelle.....	129
<b>34</b>	<b>Conclusion du document.....</b>	<b>130</b>
34.1	Licences .....	130
34.2	Déclaration d'honneur .....	130
<b>IX.</b>	<b>Références.....</b>	<b>131</b>
<b>X.</b>	<b>Glossaire.....</b>	<b>133</b>
<b>XI.</b>	<b>Table des figures.....</b>	<b>135</b>
<b>XII.</b>	<b>Annexes .....</b>	<b>138</b>
<b>1</b>	<b>Planning.....</b>	<b>138</b>
<b>2</b>	<b>Résultats des tests .....</b>	<b>139</b>
2.1	Fiabilité point de vue de face .....	139
2.2	Fiabilité point de vue de côté .....	139
2.3	Fiabilité point de vue de côté (redressé) .....	140
2.4	Fiabilité en fonction des FPS .....	140

# I. INTRODUCTION

---

## 1 Préface

---

Je me suis vu attribuer un projet pour mon travail de bachelor en filière télécommunication, orientation internet et communication, à la HEIA-FR. Il vise à développer plusieurs compétences, dont la gestion de projet, les présentations orales et la rédaction de rapport.

Un projet concret sera donc réalisé durant huit semaines par mes soins avec l'assistance et la supervision de deux professeurs : Monsieur Jean Hennebert et Madame Houda Chabbi et deux experts : Monsieur Julien Bégard et Monsieur Emeka Mosanya.

## 2 Contexte

---

Ce travail de bachelor s'inscrit dans un autre projet de plus grande envergure nommé **CityPulse**<sup>1</sup> ("Rendre visible les pulsations de la ville") des instituts **iCoSys**<sup>2</sup>, **ENERGY** et **TRANSFORM** de la **HEIA-FR**<sup>3</sup> et en partenariat avec le **Smart Living Lab**<sup>4</sup>.

CityPulse vise à collecter, mesurer et modéliser en direct, sur une maquette physique, aux moyens de l'installation d'un réseau de capteurs IoT les différents flux (de véhicules, de personnes, de pollution, d'énergie...) de la ville. C'est un projet à long terme qui a pour ambition d'aider le développement des quartiers du futur et d'améliorer le confort et la qualité de vie de ceux-ci en créant un outil d'aide à la conception interactif et dynamique. Il utilise les récentes évolutions dans le Big Data, les capteurs ainsi que les domaines tel que le Machine Learning, le Computer Vision ou les réseaux de communication afin de visualiser, d'analyser et de mesurer l'impact de ces flux.

Un prototype capturant, stockant et modélisant sur une maquette les données numériques collectées sur le plateau de Prolles est déjà existant et sert de "Proof of Concept" et de démonstrateur avant d'étendre le projet vers d'autres horizons.

---

<sup>1</sup> <https://www.smartlivinglab.ch/fr/projects/city-pulse/>

<sup>2</sup> <https://icosys.ch/>

<sup>3</sup> <https://www.heia-fr.ch/>

<sup>4</sup> <https://www.smartlivinglab.ch/>

## 3 Problématique

---

Le système actuel ne permet pas de mesurer les flux de personnes. Des tentatives sont effectuées dans cette direction au moyen de capteurs mais il est soupçonné que cette approche n'est pas la meilleure. En effet, le système produirait trop d'erreurs en cas de passage trop important à un endroit donné, les capteurs ne permettant pas une assez grande granularité.

## 4 Objectif du projet

---

1. Il faut fournir un système de capture de vidéo autonome envoyant les données en direct sur le réseau afin qu'elles soient utilisées sur une workstation distante.
2. Il faut fournir, sur une workstation, un système de Deep Learning permettant de détecter en temps réel les personnes depuis le flux vidéo reçu en direct.
3. Il faut fournir une logique de suivi et de comptage de ces personnes depuis les données retournées par la détection de personnes.
4. Un système de mesure et d'évaluation des performances doit également être proposé.
5. Il faut envoyer les données fournies par le modèle de Deep Learning afin de les rendre utilisables par le projet CityPulse.
6. Une étude économique doit être faite afin de juger du cadre légal de la surveillance de personnes et d'ensuite juger le potentiel commercial.

## 5 Contraintes

---

Les données de la solution de Deep Learning fournie doivent être formatées correctement afin de pouvoir être utilisées et stockées dans la base de données du projet CityPulse qui utilise le système **BBData**<sup>1</sup>.

---

<sup>1</sup> <https://daplab.gitlab.io/bbdata-docs/>

Le hardware utilisé devra être relativement bon marché afin de favoriser le déploiement massif de la solution.

Nous allons filmer des personnes et "surveiller" leurs déplacements ce qui équivaut basiquement à faire de la vidéosurveillance. Nous sommes donc soumis à la "Loi fédérale sur la protection des données<sup>1</sup>". Au niveau Européen, c'est la RGPD qui fournit les bases légales pour l'utilisation de systèmes de vidéosurveillance. Il faudra faire attention à les respecter.

## 6 Plan

---

Ce document aura pour but de refléter les étapes de réflexion ainsi que de documenter le travail effectué au cours des prochaines semaines et mettre en avant les résultats obtenus.

Après la rédaction d'un cahier des charges, un planning sera réalisé afin de garantir une planification optimale du temps imparti. Une fois ceci validé, une analyse technologique sera réalisée sur les différentes technologies qui seront employées, une analyse économique détaillée sera également faite afin de mieux visualiser la place de ce projet dans le monde réel. Ces analyses nous permettront d'enchaîner sur une partie conception en ayant toutes les clés en main pour réaliser un travail cohérent, réaliste, conforme au cahier des charges et servant de base solide pour la suite du projet. Le travail se terminera sur la réalisation de la solution ainsi que son test pour en valider la fiabilité.

## 7 Activités

---

### 7.1 Activités principales

#### 1. Analyse du projet existant

- 1.1. Compréhension et modélisation du fonctionnement global du projet CityPulse
- 1.2. Définition du format des données souhaitées en sortie afin d'être intégré dans la base de données BBData

→ **Livrable** : Analyse succincte du projet existant afin de soutenir les analyse économique et technologiques et permettant de comprendre complètement ses tenants et aboutissants afin de modéliser une solution cohérente et intégrable.

---

<sup>1</sup> <https://www.admin.ch/opc/fr/classified-compilation/19920153/index.html>

## 2. Analyses technologiques

- 2.1. Recherches, réflexions et documentation sur les techniques de Deep Learning et de Computer Vision permettant de faire de la reconnaissance d'objets en temps réel
  - 2.2. Comparaison entre YOLO<sup>1</sup> et ses concurrents
  - 2.3. Recherches, réflexions, documentation et comparaisons sur les moyens de suivre des entités depuis un flux vidéo
  - 2.4. Recherches, réflexions, documentations et comparaison des méthodes permettant d'envoyer un flux vidéo en direct à travers un réseau local
- **Livrable :** Choix de technologies permettant de répondre aux objectifs de manière optimale.

## 3. Analyse économique

- 3.1. Etude de la loi en vigueur en Suisse et dans le canton de Fribourg pour ce qui concerne la surveillance vidéo
  - 3.2. Analyse du marché pour comprendre les enjeux d'un tel dispositif
  - 3.3. Analyse de la concurrence pour mettre en évidence les solutions déjà en place
- **Livrable :** Analyse permettant de respecter la loi Suisse, de positionner le produit sur le marché et d'en évaluer le potentiel économique.

## 4. Conception et réalisation d'une solution permettant d'envoyer un flux vidéo à travers un réseau local

- 4.1. Modélisation de la solution de streaming
  - 4.2. Mise en place de la solution de streaming
- **Livrable :** Système permettant d'envoyer un flux vidéo en direct à travers un réseau local et étant encodé au bon format.

## 5. Conception, réalisation et évaluation d'une solution de Deep Learning permettant de détecter des individus dans un flux vidéo

- 5.1. Modélisation de la solution de Deep Learning
  - 5.2. Programmation de l'algorithme de Deep Learning
  - 5.3. Tests et évaluation du fonctionnement ainsi que de la fiabilité de la solution
- **Livrable :** Algorithme de Deep Learning répondant à l'objectif de détecter une personne passant dans un flux vidéo

---

<sup>1</sup> <https://pjreddie.com/darknet/yolo/>

**6. Conception, réalisation et évaluation d'une solution permettant de compter des individus depuis les données de la détection d'individus**

6.1. Modélisation de la solution

6.2. Programmation de l'algorithme

6.3. Tests et évaluation du fonctionnement ainsi que de la fiabilité de la solution

→ **Livrable :** Algorithme répondant à l'objectif de compter les personnes passant devant une caméra

**7. Collecte des données, formatage et intégration à la base de données de CityPulse**

7.1. Modélisation l'envoi des données à la base de données

7.2. Connexion et envoi des données sur la base de données

→ **Livrable :** Solution de formatage et d'envoie de données fonctionnant en harmonie avec BBData

## 7.2 Activités optionnelles

**1. Comparaison des performances de différentes méthodes de détection**

1.1. Mettre en œuvre les alternatives à YOLO mentionnées dans l'analyse technologique et les comparer en termes de performances et de fiabilité

→ **Livrables :** Système fonctionnel alternatif à YOLO dont les différences des performances sont documentées

**2. Anonymisation et fiabilisation du flux vidéo**

2.1. Recherches, documentation et mise en œuvre de techniques permettant de flouter des visages en temps réel sur un flux vidéo

2.2. Encryptions du flux vidéo

2.3. Rendre le flux vidéo fiable via un démarrage automatique du serveur, une détection des surcharges, des crashes...

→ **Livrable :** Flux vidéo sur LAN fiable, sécurisé et anonyme

## II. ANALYSE DU PROJET EXISTANT

---

### Introduction

Je vais analyser le projet **City Pulse** existant afin de m'aider à modéliser ma solution. Comprendre quel est son architecture et comment il fonctionne afin de proposer par la suite une solution cohérente qui puisse s'y intégrer au mieux et qui répondra aux exigences. Les points suivants vont être documentés :

- Explication de City Pulse dans son ensemble et modélisation de son architecture
- Place du projet DeepCounter dans City Pulse
- Documentation du fonctionnement de la base de données BBData

Les instituts de la HEIA-FR ont déjà passablement bien documenté et expliqué le projet global et le système actuellement en place. C'est pourquoi je n'entrerai pas trop dans les détails et étudierai uniquement les aspects qui sont intéressant pour mon travail.

## 8 Vue d'ensemble

---

### 8.1 City Pulse<sup>1</sup>

Comme déjà introduit dans le contexte, le projet City Pulse vise à mesurer et modéliser les "pulsations" de la ville en temps réel ainsi que leurs influences grâce aux technologies de l'IoT et de la Data Science. Ces mesures seront enregistrées sous formes de données et pourront ensuite être utilisées pour aider la conception de nouveaux quartiers ou améliorer la qualité de vie de ceux déjà existant. Ce projet vient en secours au canton de Fribourg qui ne dispose que de très peu de données liées aux charges de trafic. Le projet City Pulse vise, entre autres, à combler cette lacune et permettre au canton d'agir en faveur du bien-être de ses habitants.

Un projet d'une telle ambition requiert la mise en place d'une architecture complexe (Big Data) afin d'interconnecter tous les capteurs qui s'occupent de l'acquisition de données. Il faut harmoniser le tout et créer un système robuste, modulaire et documenté sur lequel de nombreuses personnes peuvent travailler en parallèle et sur lequel des nouveaux éléments d'architectures (capteurs) et des extensions peuvent être branchés.

---

<sup>1</sup> <https://www.smartlivinglab.ch/fr/projects/city-pulse/>

Il faut également offrir la possibilité à des entités tierce comme les TPF (Transports Publics Fribourgeois) de venir se raccorder aisément sur le système afin de l'enrichir de leurs données. Le tout afin de rendre la modélisation des flux la plus complète et précise possible. Au plus les données récoltées seront nombreuses et diversifiées, au plus le projet deviendra intéressant.

## 8.2 Prototype

### 8.2.1 Roadmap (1)

Un "storyboard" a été réalisé afin de modéliser la différente phase du projet, de sa conception à sa mise en œuvre à grande échelle. Ses phases principales sont les suivantes :

1. Définitions des flux à mesurer
2. Choix de localité
3. Stratégies d'acquisition des données
4. Campagne de mesures sur le site choisi
5. Développement du prototype "proof of concept"
6. Présentation du prototype
7. Extension du projet

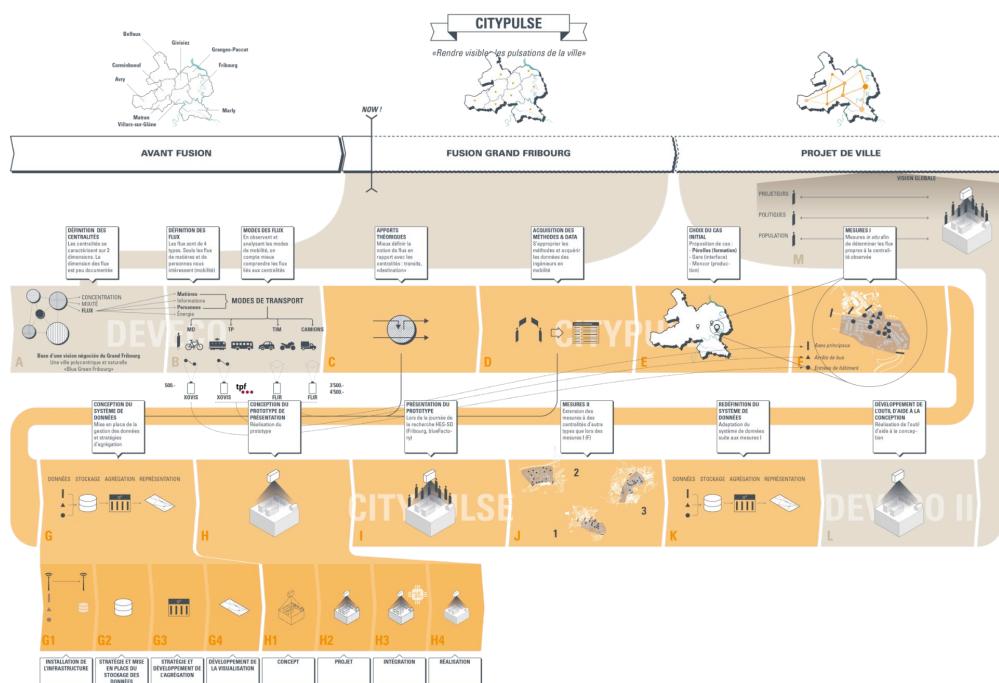


Figure 8.1 : Storyboard du projet City Pulse<sup>1</sup>

<sup>1</sup> <https://www.smartlivinglab.ch/files/180830-citypulse-jrhesso-imageprincipale-2357x1667.png>

### 8.2.2 Etat actuel du prototype

Il a été choisi, pour le prototype, de se concentrer uniquement sur la zone du Plateau de Pérrolles. Ce choix a été dicté par la nécessité de déployer un nombre relativement important de capteurs et caméras. Le faire sur des bâtiments qui appartiennent à la HEIA-FR étant, évidemment, bien plus aisé que sur une zone qui n'appartiendrait pas à l'institution.

Une maquette de la zone en matériel translucide a été réalisé par la filière Architecture de la HEIA-FR. Elle été posée sur un écran sur lequel sont modélisées les différents flux. Cette maquette est donc rendue très engageante grâce rendu 3D créé par le mix physique et numérique de celle-ci. Cela crée un médium de visualisation séduisant pour la présentation du prototype à des personnes hors du domaine technique.

Les flux actuellement modélisés sur la maquette sont les suivants :

- Véhicules sur les routes
- Personnes dans les bâtiments

Les données des véhicules ont été recueillis au moyen de caméras thermiques FLIR<sup>1</sup> durant une période limitée. Le flux des véhicules qui est modélisé sur la maquette provient de ces mêmes mesures mais ne représente pas le trafic en temps réel. Des statistiques ont été déduites des données récoltées par les caméras thermiques et ont été utilisés afin de modéliser un trafic "type" sur la maquette qui tourne en boucle. Concrètement, les véhicules sont représentés par des points se déplaçant sur les axes routiers de la maquette.

Les données des personnes dans les bâtiments proviennent de statistiques fournies par la ville de Fribourg. Sur la maquette, le taux d'occupation d'un bâtiment est indiqué en rendant le bâtiment de plus en plus rouge au fur et à mesure qu'il se remplit.

---

<sup>1</sup> <https://www.flir.fr/>



Figure 8.2 : Maquette du prototype actuel. Le taux d'occupation des bâtiments d'habitation est représenté en les illuminant en rouge

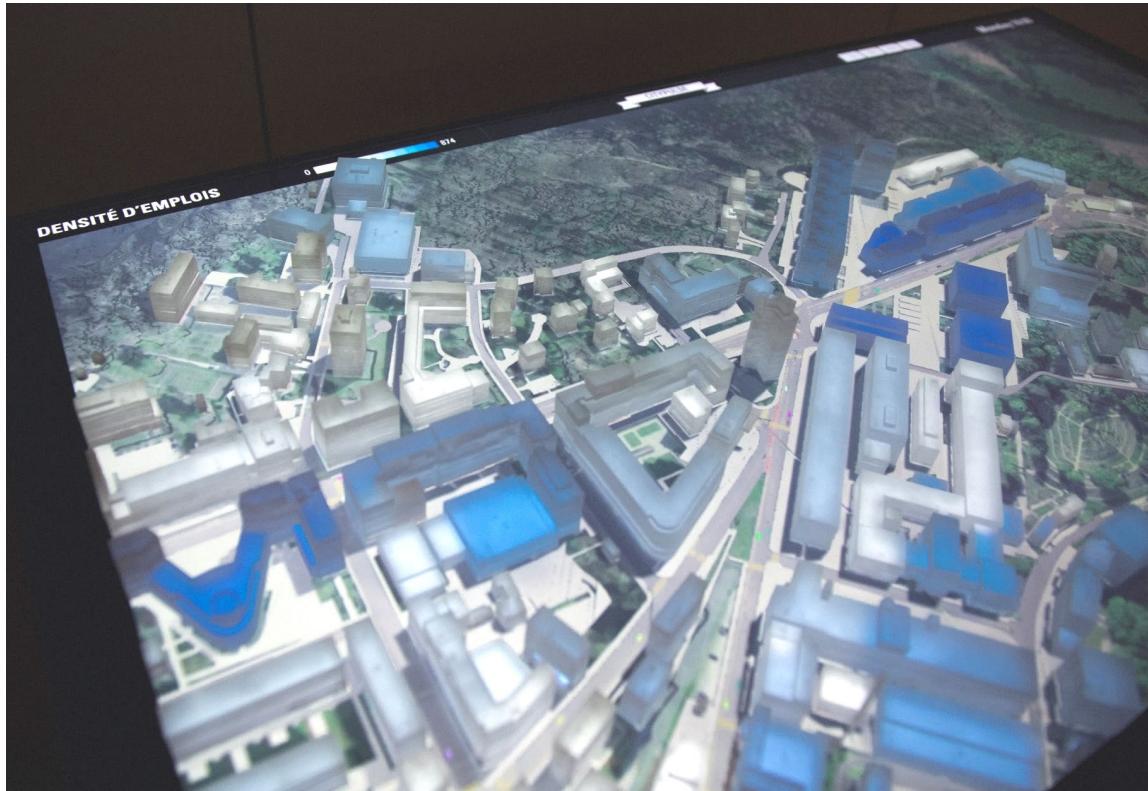


Figure 8.3 : Maquette du prototype actuel. Le taux de densité d'emplois par bâtiment est représenté en les illuminant en bleu. On peut également apercevoir des voitures modélisées sous forme de rectangles sur les axes routiers

## 8.3 Architecture

Une architecture complexe a dû être imaginée pour répondre aux objectifs du projet City Pulse. Elle se décompose en deux parties, la première est dédiée à la collecte de données :

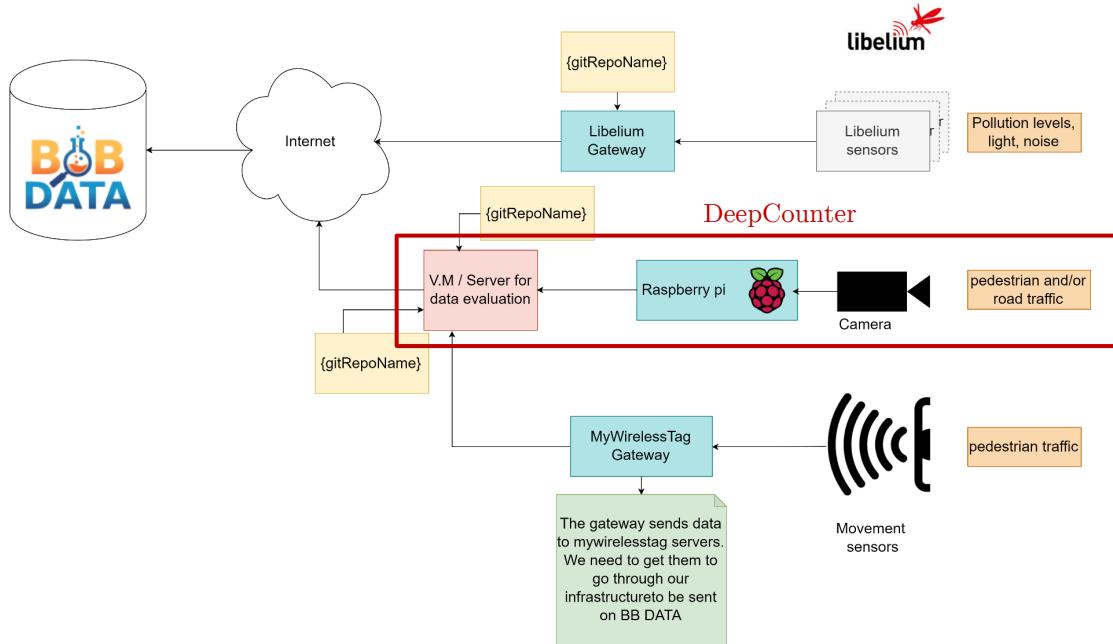


Figure 8.4 : Schéma de collecte de données (Crédit : Flavia Pittet)

On peut constater que trois types de capteurs vont être mis en place et ont pour vocation de mesurer plusieurs types de données :

- **Capteurs Libelium<sup>1</sup>** : niveaux de pollution, de lumière et de bruit
- **Caméras** : trafic routier et de piétons
- **Capteurs de mouvement** : trafic de piétons

Pour le moment, seule la partie concernant les données sur le trafic routier a été mis en place via des capteurs, comme indiqué dans le chapitre "Etat actuel du prototype" ci-dessus. Les données récoltées transitent à travers un serveur distant permettant de les évaluer et de les traiter avant de les envoyer sur une base de données (BBData) sous forme de données exploitables pour la visualisation sur la maquette.

Mon travail de bachelor va venir enrichir les capteurs de type "caméra" en proposant un moyen de quantifier les personnes se trouvant dans les bâtiments.

(2)

<sup>1</sup> <http://www.libelium.com/>

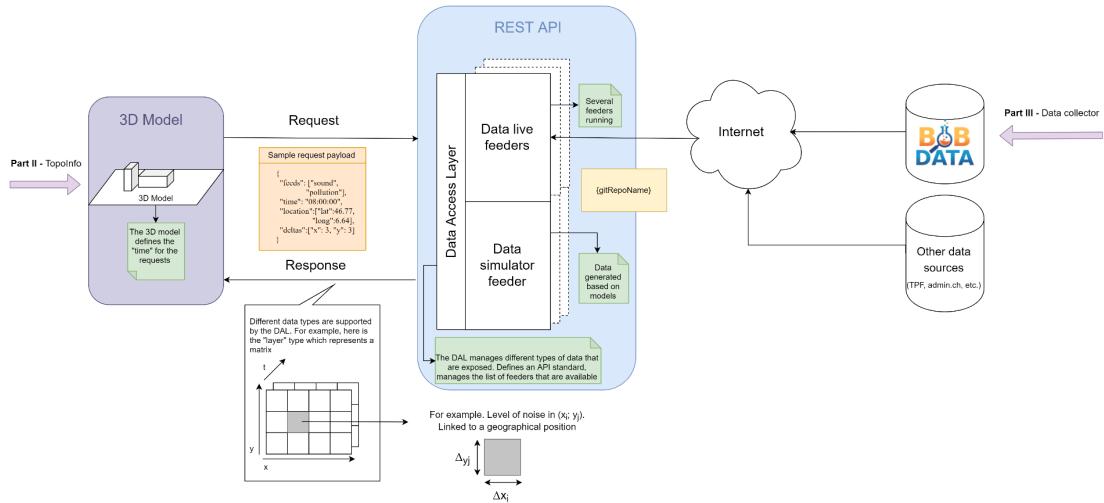


Figure 8.5 : Schéma de flux de données (Crédit : Flavia Pittet)

Comme indiqué ci-dessus, les données récoltées par les différents capteurs sont formatées et stockées sur la base de données BBData. Le système de modélisation de la maquette peut ensuite les utiliser afin de générer un modèle 3D via une API REST.

Il est important de souligner que mon projet ne touchera pas cette partie de l'architecture, je dois uniquement mettre à disposition des données exploitables par le système de modélisation sur la base de données. Tout le traitement se fera en amont de la base de données.

## 8.4 DeepCounter dans CityPulse

Nous avons constaté dans le chapitre "Etat actuel du prototype" ci-dessus que les données sur l'occupation des bâtiments de sont pas en temps réel et sont basées uniquement sur des statistiques fournies par la ville de Fribourg.

Le but du projet DeepCounter est donc de permettre un comptage en temps réel des personnes pour s'affranchir des données fournies par la ville et permettre une plus grande précision et une meilleure granularité dans la mesure de ce type de flux. Le système venant remplacer une partie existante du projet, il devra être conçu afin de s'intégrer parfaitement à l'architecture existante. Il est important de noter que cela reste un "Proof of Concept".

Les moyens qui seront mis en place pour y parvenir seront détaillés plus loin dans le chapitre "Analyse technologique" de ce travail.

## 9 Base de données

---

### 9.1 BBData<sup>1</sup>

Pour enregistrer toutes ces informations récoltées par les capteurs et pouvoir ensuite les utiliser en fonction de nos besoins, il est nécessaire d'une plateforme sécurisée de stockage de données. Cette plateforme se nomme **BBData** pour "Big Building Data".



Figure 9.1 : Logo de BBData

Dans les bâtiments connectés, le nombre de capteurs peut vite exploser. Encore plus si l'on souhaite faire des mesures sur un site entier plutôt que sur un seul bâtiment (tout le site de BlueFACTORY<sup>2</sup> par exemple). Ces capteurs récoltent un grand nombre d'informations hétérogènes qui doivent être interconnectées et mise à disposition de manière standardisée et transparente. C'est pour répondre à ce problème que le développement de BBData a été initié par le Smart Living Lab.

BBData n'a pas été développé spécifiquement pour City Pulse, mais plutôt pour servir de plateforme générale et d'outil permettant de stocker, de traiter, d'analyser et de partager des données de bâtiments intelligents de tout type. Cette plateforme est dans le cloud, "scalable" et utilise les dernières technologies du domaine de l'IoT et du Big Data afin d'offrir aux développeurs un environnement et des outils puissants, accessibles, standardisés et sécurisés.

---

<sup>1</sup> <https://daplab.gitlab.io/bbdata-docs/>

<sup>2</sup> <https://www.bluefactory.ch/>

### 9.1.1 Fonctionnement<sup>1</sup>

#### Objets virtuels<sup>2</sup>

Les objets virtuels sont un concept important de BBData, car c'est la forme sous laquelle les données des capteurs sont récupérées. Un objet virtuel peut appartenir à zéro, un ou plusieurs groupes d'objets. Des métadonnées sont stockées dans ceux-ci :

- ID (unique, généré à la création de l'objet)
- Date de création
- Nom
- Description
- Unité de la mesure
- Tags (pour faciliter la recherche)
- État (activé ou désactivé, un objet désactivé ne peut pas recevoir de mesure)
- Propriétaire (qui appartient à un groupe d'utilisateur, permet de gérer les accès)

Les données mesurées sont associées à un "timestamp". Le type de variable utilisé (String, int, ...) est défini par le type d'unité choisi (speed, degrees, ...).

```

1 [ {
2   "objectId": 6538,
3   "unit": {
4     "name": "tension DC",
5     "symbol": "VDC",
6     "type": "float"
7   },
8   "values": [
9     {
10       "timestamp": "2019-05-01T00:00:37.000",
11       "value": "48.779999"
12     },
13     {
14       "timestamp": "2019-05-01T00:05:36.000",
15       "value": "48.750000"
16     }
17   ]
}

```

Figure 9.2 : JSON retourné lorsque l'on effectue une requête GET sur un objet BBData

<sup>1</sup> <https://daplab.gitlab.io/bbdata-docs/components/>

<sup>2</sup> <https://daplab.gitlab.io/bbdata-docs/conception/>

## Parcours type d'une mesure

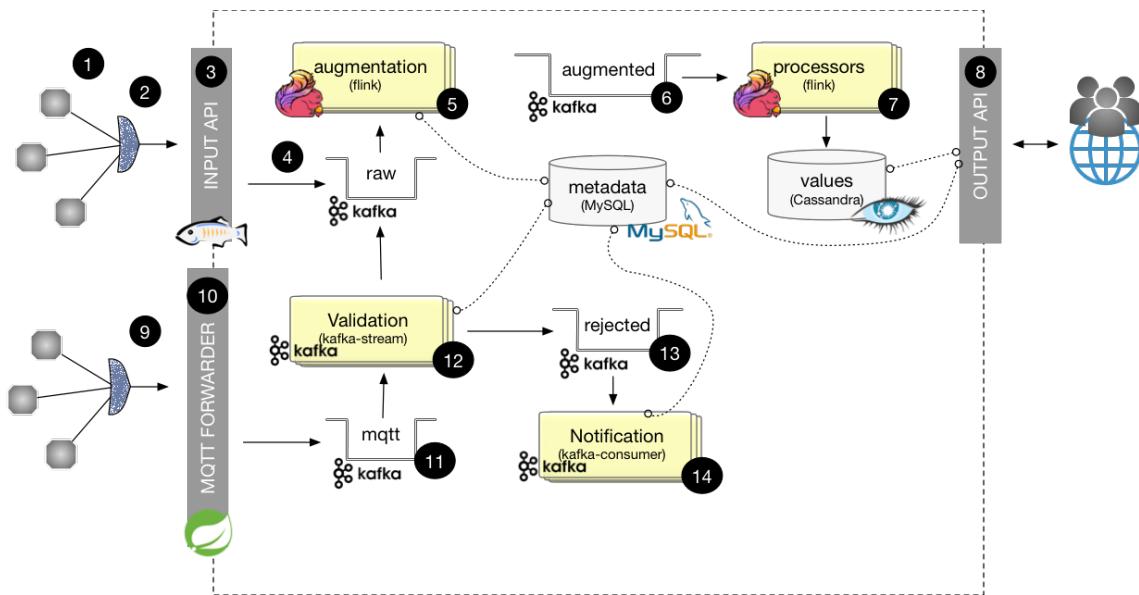


Figure 9.3 : Parcours d'une mesure dans le "pipeline" BBData<sup>1</sup>

Plusieurs types de capteurs font des mesures (1) qui sont ensuite envoyées dans un collecteur (2). Il permet d'assurer la compatibilité des données mesurées avec la plateforme en créant pour chacune un enregistrement dans la base de données sous forme d'objet virtuel. Ces objets sont ensuite envoyés en entrée à l'API REST en format JSON (3) qui se charge de vérifier la validité et l'authenticité de la mesure. Le message est ensuite mis dans une file (4) et y est "augmentée" de métadonnées récupérées d'une base de données SQL (5). Le résultat est ensuite compressé et envoyé dans une seconde file (6) depuis laquelle nous pourrons y faire du traitement (7). Les données traitées sont accessibles via une API REST ou via une application web (8).

Je précise que nous n'utiliserons pas le traitement offert dans BBData en (7). Tout le traitement (reconnaissance et comptage de personnes) se fera en amont, avant l'envoi des données sur la base de données. Nous utiliserons cette dernière uniquement afin de fournir des données utiles à la maquette afin de modéliser les flux de personnes dans les bâtiments.

<sup>1</sup> <https://daplab.gitlab.io/bbdata-docs/resources/components-big.png>

### Parcours type d'un utilisateur

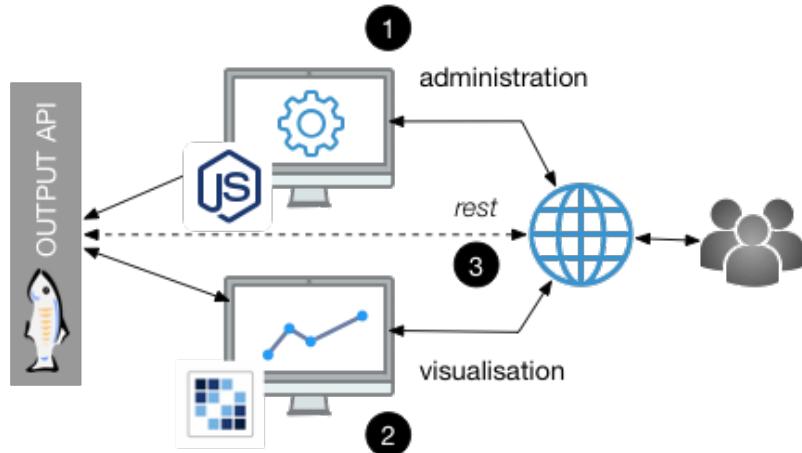


Figure 9.4 : Parcours d'un utilisateur souhaitant utiliser BBData<sup>1</sup>

Dans ce scénario, un utilisateur a un capteur et souhaite envoyer ces mesures sur BBData. Il doit en premier créer depuis l'interface d'administration de BBData (1) un objet virtuel correspondant à la donnée mesurée. Un ID unique sera assigné à l'objet ainsi qu'un "Token" pour la sécurité. L'utilisateur pourra ensuite envoyer, sous forme de requêtes HTTP POST, des données à l'API afin de les inscrire dans la base de données. L'ID de l'objet ainsi que le Token devront être spécifiés dans le corps de la requête.

L'utilisateur pourra visualiser ses données sous forme de graphiques via l'application web (2) ou pourra récupérer les données brutes via l'API REST (3).

<sup>1</sup> <https://daplab.gitlab.io/bbdata-docs/resources/user-apis-schema.png>

### 9.1.2 Stack technologique

L'écosystème BBData utilise les technologies open source suivantes :

- **Apache Hadoop** : Framework Java pour la création d'applications distribuées
- **GlassFish** : Serveur d'applications Java EE fournissant une API REST
- **Apache Cassandra** : SGBD NoSQL
- **Apache Flink** : Framework de traitement de flux
- **Apache Kafka** : Plateforme de diffusion distribuée
- **MySQL** : SGBD SQL
- **RAML** : Description d'API RESTful



Figure 9.5 : Stack technologique de BBData<sup>1</sup>

## Synthèse

Cette analyse nous a permis de constater que le projet City Pulse est construit sur une base solide, mais que beaucoup reste encore à faire afin de l'emmener au bout de ses ambitions. Nous avons cerné ses objectifs, compris le but qu'il souhaite atteindre et la plus-value qu'il va apporter.

Le projet étant très bien documenté, nous avons les clés qui nous permettent d'en comprendre le fonctionnement, les interactions ainsi que l'architecture et nous pourrons ainsi y intégrer au mieux le projet DeepCounter afin d'augmenter City Pulse d'une nouvelle fonctionnalité : celle de compter le flux de personnes dans les bâtiments.

<sup>1</sup> <https://daplab.gitlab.io/bbdata-docs/resources/technologies.png>

## III. ANALYSE TECHNOLOGIQUE

---

### Introduction

Nous avons quatre tâches principales à effectuer pour répondre au cahier des charges :

- Streaming d'un flux vidéo vers un ordinateur distant connecté au LAN
- Détection d'êtres humains en temps réel sur le flux vidéo (Bounding Boxes).
- Tracking de ces personnes au long de leur parcours dans le cadre de la caméra.
- Comptage des personnes depuis un flux vidéo en direct afin de définir combien se trouvent dans un bâtiment.

Dans ce projet qui est un "Proof of Concept", il a été suggéré par M. Hennebert d'utiliser les technologies suivantes :

- Capture de vidéo depuis un Raspberry Pi et envoie du flux sur le LAN
- Emploi de YOLO pour la détection de personnes
- Emploi de OpenCV pour les tâches de Computer Vision

Il est cependant nécessaire de faire un éventail des différentes alternatives afin de trouver la solution qui siéra au mieux au projet pour son déploiement à grande échelle. Il faut donc effectuer une analyse des différentes technologies possibles afin de réaliser les tâches voulues, il faut les comparer. Il faut également juger des risques auxquels chaque tâche peut être soumise, les recenser et trouver des solutions.

## 10 Fiabilité

---

La question de la fiabilité du système est secondaire pour le moment, car ce projet vise uniquement à proposer un "Proof of Concept". Il est cependant intéressant de se pencher sur les différents problèmes qui pourraient entraîner un dysfonctionnement du dispositif. En modélisant et prévoyant aussi tôt ces problèmes dans la phase de développement, il serait beaucoup plus aisément et moins chronophage de les intégrer dans le futur, contrairement à si cela avait été des fonctionnalités ajoutées en dernier recours tout à la fin. Cela est illustré par le schéma suivant :

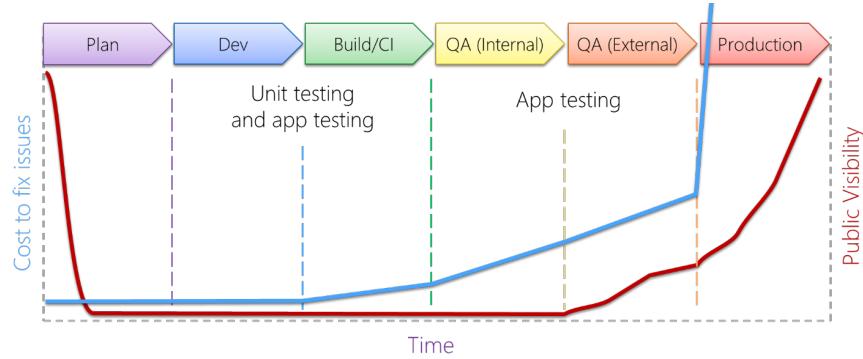


Figure 10.1 : Coût de la correction de bugs en fonction de l'avancée dans le cycle de vie<sup>1</sup>

Les problèmes que pourrait rencontrer notre dispositif de capture vidéo sont les suivants, ils peuvent entraîner des données faussées ou absentes ou une simple perte de celles-ci :

- Perte de la connexion au réseau
- Mise hors tension du dispositif
- Crash du dispositif ou redémarrage
- Freeze du dispositif

Certains de ces potentiels problèmes peuvent être résolus grâce à, par exemple, Cron<sup>2</sup> qui permet d'automatiser l'exécution de services ou grâce à Monit<sup>3</sup> qui permet de surveiller l'état de ceux-ci. Certains systèmes de streaming pour le RPI, comme UV4L<sup>4</sup>, ont également des fonctionnalités permettant de garantir la fiabilité directement "built-in". Reste encore le problème de la perte de données et l'absence de mesures durant un temps non défini.

## 11 Solutions hardware

Nous devons imaginer un système permettant de capturer une vidéo et d'effectuer un traitement sur celui-ci. Il a été choisi au début de ce travail une approche consistant à faire l'acquisition du flux vidéo sur un Raspberry Pi, de l'envoyer sur le réseau et de faire le traitement sur une workstation distante qui enverra les données sur BBData. Nous appellerons cette méthode "**traitement distant**". Ce choix a été fait afin de permettre une intégration aisée du système dans des installations existantes de vidéosurveillance.

<sup>1</sup> <https://docs.microsoft.com/en-us/visualstudio/cross-platform/tools-for-cordova/debug-test/unit-test-primer?view=toolsforcordova-2017>

<sup>2</sup> <https://doc.ubuntu-fr.org/cron>

<sup>3</sup> <https://mmonit.com/monit/>

<sup>4</sup> <https://www.linux-project.org/uv4l/installation/>

Il se peut cependant que cette approche ne soit pas satisfaisante pour plusieurs raisons telles qu'une latence trop grande, l'impossibilité de récupérer le flux vidéo en streaming, des problèmes de stabilité, de sécurité... C'est pourquoi je vais aussi m'intéresser à une autre méthode nommée : "**traitement intégré**" ou tout le traitement sur le flux vidéo est fait directement sur le dispositif de capture et depuis lequel on envoie directement les données.

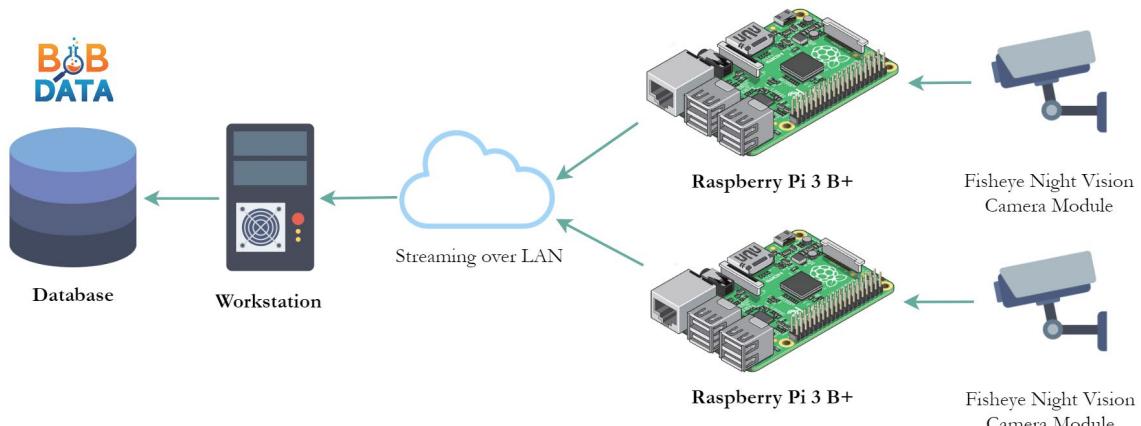
Il faut être conscient que la solution qui sera produite et qui permettra de faire la détection et le comptage de personnes sera utilisable tant pour la méthode distante que pour la méthode intégrée, car les deux seront basées sur un OS Linux. Seules la source du flux vidéo ainsi que la fluidité et la résolution changeront concrètement, hors des aspects pratiques.

## 11.1 Traitement distant

C'est la méthode qui a été choisie pour la réalisation du prototype pour les raisons susmentionnées. La qualité (en termes de FPS et de résolution) devrait être meilleure avec ce type d'approche. Cependant, une latence est introduite en envoyant le flux vidéo à travers le réseau. Cela nécessite plus de bande passante pour faire du streaming que pour simplement faire des requêtes à la base de données. Cette solution est également difficilement "scalable", car il faudra lancer une instance du système de CV pour chaque flux vidéo reçu par la workstation. On peut donc imaginer qu'on peut vite être limité par la puissance de calcul au cas où l'infrastructure devrait supporter des dizaines de dispositifs de capture de vidéo.

### 11.1.1 Architecture et hardware

Les données sont récoltées au moyen de caméra connectée à un Raspberry Pi :



*Figure 11.1 : Schéma de l'architecture du traitement distant*

Les caméras sont grand-angle (120°) et sont capables de voir durant la nuit grâce à des faisceaux infrarouges. Un serveur est exécuté sur le RPI, il récupère le flux de la caméra, défini sa résolution, ses FPS, l'encode et le rend disponible via une adresse IP sur le réseau LAN. Le flux vidéo est ensuite récupéré de l'autre côté du réseau sur une workstation qui a pour rôle d'y appliquer des techniques de computer vision afin de détecter et de compter les personnes qui y passent. Les informations sont finalement envoyées sur la base de données.

## 11.2 Traitement intégré

L'alternative au traitement distant est de faire un traitement intégré. Cela nous permet de nous affranchir d'une workstation, réduisant les coûts et permet également de ne pas avoir à gérer l'envoie et la sécurité d'un flux vidéo à travers un réseau. Ce concept consistant à traiter les données à proximité de là où elles sont capturées est nommé le "Edge Computing"<sup>1</sup>.

Chaque dispositif capturant un flux vidéo se charge d'effectuer son propre traitement directement en son sein. Ils sont connectés au réseau uniquement pour envoyer les données sur le nombre de personnes comptées sur la base de données. Cela permet une scalabilité bien meilleure, car il suffit de brancher un autre dispositif sur le réseau et de le configurer, sans avoir à s'inquiéter si la workstation est surchargée ou non. La contrepartie est une perte de qualité (de FPS et de résolution) comparé à un traitement intégré. Un mini-ordinateur, même spécialisé dans une tâche (ASIC) n'aura pas les mêmes performances qu'une workstation.

### 11.2.1 Architecture

La grosse différence est qu'il n'y a pas de workstation. Le système fonctionne autrement de manière très similaire :

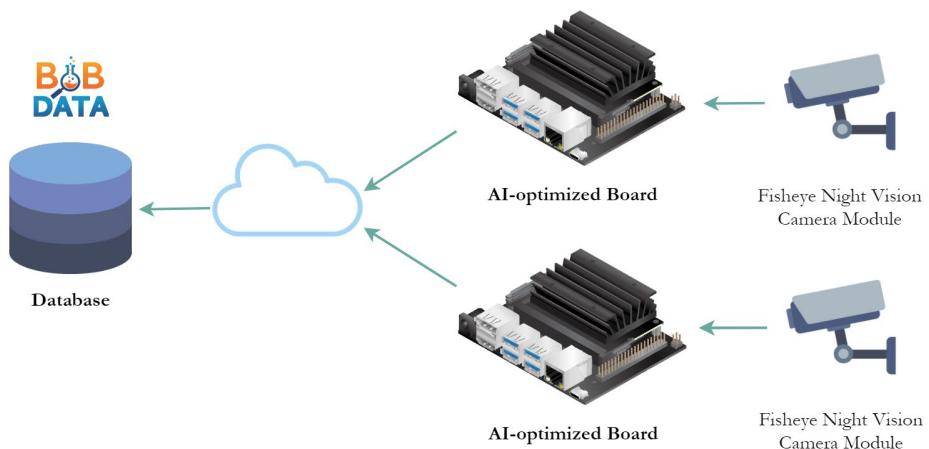


Figure 11.2 : Schéma de l'architecture du traitement intégré

<sup>1</sup> [https://en.wikipedia.org/wiki/Edge\\_computing](https://en.wikipedia.org/wiki/Edge_computing)

### 11.2.2 Hardware

Un nombre relativement important de constructeurs de matériel informatique proposent désormais du matériel permettant de faire du "Edge Computing" sous plusieurs formes :

- **Accélérateurs USB pour Raspberry Pi**
  - Google Coral USB Accelerator (Compatible qu'avec Tensorflow Lite)
  - Intel Neural Compute Stick<sup>1</sup> (Comptabilité limitée avec YOLO)
  - JeVois<sup>2</sup> (Avec caméra intégrée, mais très peu performante)
- **Micro-ordinateurs**
  - Nvidia Jetson Nano<sup>3</sup>
  - Google Coral Dev Board<sup>4</sup> (Compatible qu'avec Tensorflow Lite)
- **Systèmes "tout-en-un"**
  - Google AIY Vision Kit<sup>5</sup> (Compatible qu'avec Tensorflow Lite)
  - Amazon AWS DeepLens<sup>6</sup> (Dépendant des AWS)
  - Qualcomm Vision AI Dev Kit<sup>7</sup> (Pas en vente)

Celui qui propose les meilleures performances à un prix correct et qui offre une grande flexibilité dans les Librairies qui peuvent y être utilisées est le : **Jetson Nano**. Les autres solutions ne sont pas pertinentes dans notre cas d'utilisation, car soit trop limitées ou dépendantes de services externes ou soit proposant de moins bonnes performances. Il a également le gros avantage de venir préinstaller avec une installation de OpenCV qui supporte l'accélération GPU.

Le Jetson Nano permet, pour des applications de CV, un gain de performances de l'ordre d'environ 2500% par rapport à un Raspberry Pi seul et d'environ 250% par rapport à l'Intel Neural Compute Stick branché sur un RPI.<sup>8</sup>



<sup>1</sup> <https://software.intel.com/en-us/neural-compute-stick>

<sup>2</sup> <http://jevois.org/>

<sup>3</sup> <https://www.nvidia.com/fr-fr/autonomous-machines/embedded-systems/jetson-nano/>

<sup>4</sup> <https://coral.withgoogle.com/products>

<sup>5</sup> <https://aiyprojects.withgoogle.com/vision/>

<sup>6</sup> <https://aws.amazon.com/fr/deeplens/>

<sup>7</sup> <https://developer.qualcomm.com/hardware/vision-ai-development-kit>

<sup>8</sup> <https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks>

## 11.3 Choix pour ce projet

On écarte tous les systèmes autres que le Jetson Nano, car ils ne répondent pas aux critères.

### Critères (dans l'ordre d'importance) :

- **Scalabilité** : Capacité de la solution à être mis en œuvre à grande échelle
- **Intégration** : Facile à intégrer à un système existant
- **Utilisabilité** : Dois être simple de récupérer le flux vidéo pour l'utiliser
- **Qualité** : FPS et résolution doivent être le plus grand possible

Le comparatif ci-dessous montre que le système Jetson Nano est supérieur sur tous les aspects autres que l'intégration et que la qualité ("+" signifie qu'un système est mieux dans le critère donné et "-" qu'il performe moins bien)

Matériel	Scalab.	Intégr.	Utili.	Qual.
RPI + Workstation	--	++	-	+
Jetson Nano	++	-	+	-

Le reste de ce rapport considérera que nous avons choisi l'approche : **traitement distant** (donc RPI + Workstation). Nous avons constaté que ce n'était pas forcément la meilleure solution, mais cela suffit amplement pour l'optique de "Proof of Concept" de ce travail.

Le "Edge Computing" peut cependant être la solution qui est le plus propice à être mise en place à une grande échelle, spécialement pour City Pulse.

## 12 Streaming depuis le RPI

Étant donné que le traitement distant a été choisi, il faut à présent trouver un moyen d'envoyer le flux vidéo provenant du RPI sur le réseau local. Il faut le rendre accessible via une adresse IP et l'encoder dans un format qui permettra d'y appliquer un traitement. L'envoi du flux vidéo se doit d'être fiable, d'être compatible avec la librairie qui sera utilisée et d'avoir une latence faible.

On utilisera la librairie **OpenCV** pour ce projet, car il fait office de ténor dans le domaine du Computer Vision et offre toutes les fonctionnalités nécessaires afin de réaliser nos objectifs. J'en parlerai plus en détail plus tard afin de justifier ce choix, mais OpenCV offre spécifiquement une classe<sup>1</sup> permettant la capture des flux vidéo.

OpenCV supporte de nombreux encodages vidéo comme MJPEG, H264, MPEG ou AVI grâce à la librairie FFmpeg<sup>2</sup> et peut capturer des flux provenant de sources utilisant RTSP ou HTTP comme protocole de transmission. Nous avons donc un grand éventail de possibilité afin de récupérer un flux vidéo en direct, reste maintenant à savoir comment l'envoyer et comment le rendre disponible sur le réseau local via une adresse IP.

## 12.1 Systèmes de streaming

Voici les différentes solutions permettant de streamer un flux vidéo sur le réseau :

### 12.1.1 Netcat<sup>3</sup>

C'est certainement la façon la plus simple de faire du streaming depuis le RPI. La qualité est bonne et quasiment sans latence. Netcat est un utilitaire permettant d'ouvrir des connexions TCP ou UDP entre deux machines et d'y faire transiter des données. On peut donc l'utiliser pour ouvrir, sur un port, un socket entre notre RPI et notre machine de l'utiliser pour envoyer le flux vidéo. Son fonctionnement repose sur une logique client/serveur :

1. **Du côté client** (destination/workstation) : ouvrir une connexion sur un port qui va "écouter" le trafic entrant.
2. **Du côté serveur** (source/RPI) : se connecter au client grâce à son adresse IP et au port ouvert et envoyer les données souhaitées.

Raspivid est l'outil développé par RPI pour capturer et encoder des vidéos depuis le module caméra. On "pipe" donc son output dans netcat et on l'envoie sur la machine de destination via son port et son adresse IP. On reçoit le flux de l'autre côté (sur la workstation) qu'on peut ensuite lire avec un lecteur multimédia comme Mplayer<sup>4</sup>.

---

<sup>1</sup> [https://docs.opencv.org/4.1.0/d8/dfe/classcv\\_1\\_1VideoCapture.html#details](https://docs.opencv.org/4.1.0/d8/dfe/classcv_1_1VideoCapture.html#details)

<sup>2</sup> <http://ffmpeg.org/>

<sup>3</sup> <https://nmap.org/ncat/>

<sup>4</sup> <http://www.mplayerhq.hu/design7/news.html>

```
Sender :  
> raspivid -t 0 -w 1920 -h 1080 -fps 30 -n -o - | nc 160.98.30.133 5000  
  
Receiver :  
> nc -l -p 5000 | mplayer -fps 40 -cache 1024 -
```

Figure 12.1 : Commandes permettant de faire un stream avec Netcat

C'est une méthode facile et légère à mettre en place, mais on est obligé d'avoir un client netcat qui écoute pour recevoir le flux vidéo. Cela semble poser des problèmes pour le récupérer et l'utiliser sur OpenCV, car il faudrait ouvrir la connexion netcat depuis celui-ci.

### 12.1.2 GStreamer<sup>1</sup>

GStreamer est une alternative qui propose de bonnes performances et une latence faible. Il est cependant difficile à installer sur le RPI, le gestionnaire de paquets ne l'incluant pas. Le problème est le même qu'avec Netcat ou il est difficile de récupérer le flux envoyé avec GStreamer depuis le RPI sur OpenCV. Il faudrait mettre en place un pipe FIFO, mais cela serait beaucoup de travail supplémentaire. Autant utiliser une solution "plug & play".

### 12.1.3 MJPG-Streamer<sup>2</sup>

MJPG-Streamer offre une solution très intéressante. Il peut être utilisé pour diffuser un flux vidéo sur un réseau IP à partir d'une webcam vers différents types de visualiseurs tels qu'un navigateur internet, VLC, Mplayer et autres logiciels capables de recevoir des flux MJPG. On accède au stream via une adresse IP. Il est complexe à installer et mettre en place à cause de toutes les dépendances qu'il requiert.

### 12.1.4 Motion<sup>3</sup>

Motion est plus destiné aux systèmes de surveillance style "CCTV". Il intègre des fonctionnalités de détection de mouvement, mais n'est, à cause de cette spécialisation pour la vidéosurveillance, pas adapté à notre cas d'utilisation. Il a une latence excessive.

<sup>1</sup> <https://gstreamer.freedesktop.org/>

<sup>2</sup> <https://github.com/jacksonliam/mjpg-streamer>

<sup>3</sup> <https://motion-project.github.io/>

### 12.1.5 VLC<sup>1</sup>

VLC inclut des fonctionnalités permettant de faire du streaming. Le package est disponible sur le RPI et s'installe facilement. Il permet de mettre en place un flux vidéo utilisant le protocole RTSP de manière assez intuitive, mais possède la pire latence de toutes les solutions proposées. Le streaming ainsi créé est lisible que depuis un client et pas depuis un browser.

### 12.1.6 UV4L<sup>2</sup>

UV4L est considéré comme la solution "reine" pour faire du streaming depuis le RPI. Il possède une excellente documentation, s'installe facilement et met en place automatiquement un serveur de streaming lors de la mise sous tension. On peut visualiser le stream dans un navigateur via un site web offrant de nombreuses fonctionnalités et on peut également récupérer le flux vidéo sur HTTP (TCP) directement en MJPEG ou H264 depuis une adresse IP ce qui est parfait pour l'intégrer dans une application OpenCV. Ses points faibles sont qu'il consomme plus de bande passante et est assez lourd pour le CPU.



Figure 12.2 : Interface web de UV4L

<sup>1</sup> <https://www.videolan.org/vlc/streaming.html>

<sup>2</sup> <https://www.linux-project.org/uv4l/>

## 12.2 Comparaison des solutions<sup>1</sup>

On écarte les solutions qui ne sont pas ou peu compatibles avec OpenCV (Netcat et GStreamer) et celles qui sont faites pour une autre utilisation (Motion).

**Critères (dans l'ordre d'importance) :**

- **Facile à déployer** : Installation et automatisation
- **Images par seconde (FPS)** : le plus de FPS possibles
- **Fiabilité et robustesse** : peu gourmand en CPU et bande passante
- **Latence** : Le moins de latence possible

Méthode	Facile	FPS	Fiabilité	Latence
MJPG-Streamer	+	-	+	+
VLC	++	-	+	--
UV4L	++	++	+	++

("+" signifie qu'un système est mieux dans le critère et "-" qu'il performe moins bien)

Le comparatif ci-dessus dirige mon choix vers l'utilisation de **UV4L**. Ce sont principalement ses très bonnes performances ainsi que son fonctionnement "plug & play" qui ont fait pencher la balance en sa faveur.

## 12.3 Encodage du flux vidéo<sup>2</sup>

UV4L permet plusieurs types d'encodage de la vidéo dont : **H264** et **MJPEG**.

### MJPEG (Motion JPEG)<sup>3</sup>

C'est un format de compression vidéo, ou codec, dans lequel chaque image subit individuellement une compression JPEG. C'est un format qui est grandement utilisé par les dispositifs de caméra IP, car il offre une grande compatibilité.

---

<sup>1</sup> [http://stephane.lavirotte.com/perso/rov/video\\_streaming\\_pi2.html](http://stephane.lavirotte.com/perso/rov/video_streaming_pi2.html)

<sup>2</sup> <https://blog.angelcam.com/what-is-the-difference-between-mjpeg-and-h-264/>

<sup>3</sup> [https://en.wikipedia.org/wiki/Motion\\_JPEG](https://en.wikipedia.org/wiki/Motion_JPEG)

**Avantages :**

- Robuste, la perte d'une image n'affecte pas le flux vidéo
- Codec simple et peu gourmand en ressources CPU
- Supporté nativement par OpenCV

**Inconvénients :**

- Pas de son (pas important pour notre projet)
- Consomme plus de bande passante que H264

**H264<sup>1</sup>**

Comme MJPEG, c'est une norme d'encodage de vidéos. Son but est de diminuer grandement l'utilisation de la bande passante en appliquant une meilleure compression. Il le fait en compressant le flux vidéo dans son ensemble plutôt qu'image par image.

**Avantages :**

- Consomme beaucoup moins de bande passante grâce à une compression efficace
- Adapte sa qualité en fonction de la bande passante

**Inconvénients :**

- Encodage et décodage lourds pour le CPU
- Pose des problèmes de compatibilité avec OpenCV

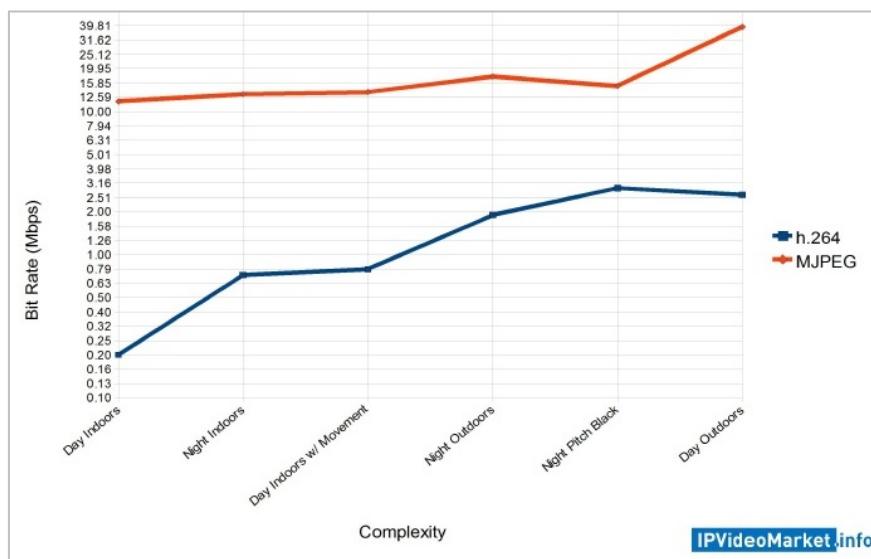


Figure 12.3 : Comparaison de l'utilisation de bande passante de MJPEG et H264 (à 30 fps) dans différents scénarios<sup>2</sup>

<sup>1</sup> [https://en.wikipedia.org/wiki/H.264/MPEG-4\\_AVC](https://en.wikipedia.org/wiki/H.264/MPEG-4_AVC)

<sup>2</sup> <https://ipvm.com/reports/h264-mjpeg-bandwidth-quality-test>

Pour ce projet, nous allons utiliser **MJPEG** pour sa simplicité d'utilisation grâce à une grande compatibilité et le fait que la bande passante disponible ne soit pas un aspect critique.

Cependant, si le temps me le permet, j'étudierai et comparerai les performances des deux codecs afin de juger ce qu'ils peuvent apporter à une mise en œuvre à grande échelle. H264 semble quand même être à favoriser pour ce genre d'application afin de diminuer la charge sur le réseau.

## 13 Détection et comptage

---

### 13.1 Ressources

Je reste abstrait dans ce chapitre, les domaines cités étant excessivement vastes et complexes. Ce travail de bachelor ne me laisse pas le temps d'explorer chaque publication scientifique en détail. Pour en savoir plus sur le fonctionnement de OpenCV, des CNN, des DNN, du computer vision, de l'Object Detection, de l'Object Tracking, du machine learning et du deep learning en général, je vous invite à aller lire les ressources suivantes :

- Le site : [https://vas3k.com/blog/machine\\_learning/](https://vas3k.com/blog/machine_learning/)
- Le site : <https://towardsdatascience.com/>
- Le site : <https://machinelearningmastery.com/>
- Le livre : *Fundamentals of Deep Learning - Nikhil Buduma* (3)
- Le livre : *Hands-On Machine Learning - Aurélien Géron* (4)
- Le livre : *Deep Learning With Python - François Chollet* (5)

### 13.2 Computer vision

*"At an abstract level, the goal of computer vision problems is to use the observed image data to infer something about the world."<sup>1</sup>*

Le projet nous demande de faire deux traitements sur le flux vidéo du RPI :

- Détection des êtres humains présents dans le cadre.
- Comptage des personnes qui entrent et sortent du cadre

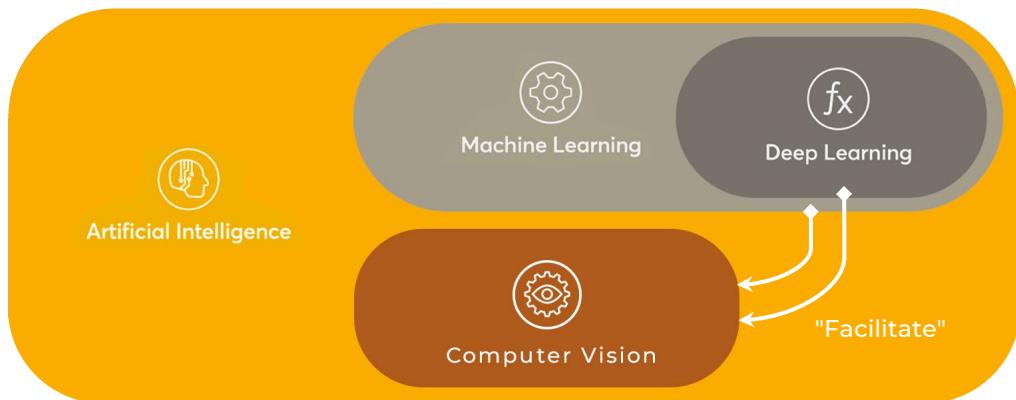
---

<sup>1</sup> Page 83, Computer Vision : Models, Learning, and Inference (Simon J. D. Prince), 2012.

Pour ce faire, je vais utiliser des méthodes dites de "Computer Vision" ou "Vision par Ordinateur" (abrégé CV). Ce domaine est une branche de l'intelligence artificielle et vise à traiter, comprendre et extraire des informations depuis des images digitales comme des photos ou des vidéos. Son but ultime, et comme son nom l'indique, est de permettre aux ordinateurs de "voir" et de comprendre le monde d'une manière qui se rapproche de celle des humains. C'est une tentative visant à automatiser des tâches que, jusqu'à présent, seul un être humain était capable d'exécuter.

Les tâches typiques qui peuvent profiter du Computer Vision sont :

- Reconnaissance (de caractères, de visages, d'objets, d'endroit...)
- Détection d'entités
- Reconstruction de scènes / Restauration d'images



*Figure 13.1 : Place du CV dans le domaine de l'intelligence artificielle*

Le domaine du Computer Vision a pu bénéficier de beaucoup des récentes avancées dans les domaines du Machine Learning et surtout du **Deep Learning** qui ont été permises par l'explosion des performances de GPU et l'abondance de données utiles. Les avantages offerts par le DL seront cités juste après.

L'utilisation du Deep Learning dans le Computer Vision est donc devenue nécessaire afin d'obtenir des solutions "state-of-the-art".

### 13.3 Deep Learning<sup>1</sup> (6)

*"Deep learning is a specific subfield of machine learning: a new take on learning representations from data that puts an emphasis on learning successive layers of increasingly meaningful representations. The deep in deep learning isn't a reference to any kind of deeper understanding achieved by the approach; rather, it stands for this idea of successive layers of representations."*<sup>2</sup>

Le Deep Learning est une sous-branche du Machine Learning et une évolution de celui-ci visant basiquement à permettre aux ordinateurs d'apprendre, de comprendre des données et d'en faire des déductions en s'inspirant du fonctionnement du cerveau humain. Un modèle de DL est capable de faire une prédiction seulement s'il a été entraîné ultérieurement. L'entraînement peut se faire de manière supervisée ou non.

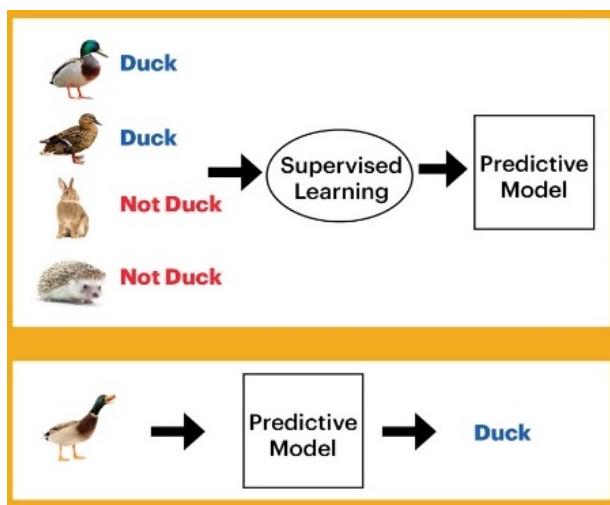


Figure 13.2 : Schéma d'apprentissage supervisé, phases d'entraînement et de prédiction<sup>3</sup>

Le fonctionnement est basé sur le concept de réseaux de neurones artificiels et est le suivant : apprendre au moyen de couches successives connectées entre elles. On extrait des "features" qui vont de très simples vers de moins en moins abstraits. On assigne une importance à ces "features" sous forme de poids et de biais qu'il définit au fur et à mesure de la phase d'entraînement (forward propagation) il les ajuste (back propagation) pour correspondre au mieux aux attentes.

<sup>1</sup> [https://en.wikipedia.org/wiki/Deep\\_learning](https://en.wikipedia.org/wiki/Deep_learning)

<sup>2</sup> Page 31, (5)

<sup>3</sup> [https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2018/09/WD\\_2.jpg](https://2s7gjr373w3x22jf92z99mgm5w-wpengine.netdna-ssl.com/wp-content/uploads/2018/09/WD_2.jpg)

La différence basique entre Machine Learning et Deep Learning est que ce dernier permet un apprentissage autonome des "features" d'une donnée, là où elles doivent être définies à la main dans le cas du Machine Learning. Cela rend le système plus proche d'une approche humaine et permet l'analyse de données très complexes sur lesquels il serait très difficile, voire impossible, pour un humain d'en extraire des features utilisables et pertinentes pour la prédiction.

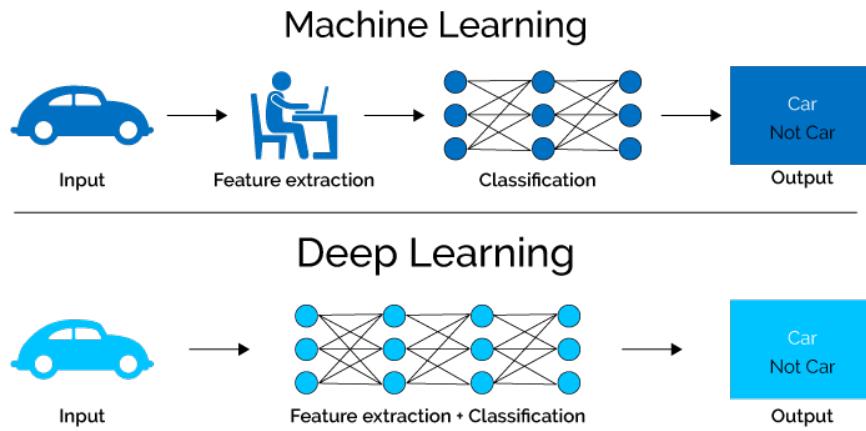


Figure 13.3 : Visualisation des différences entre ML et DL<sup>1</sup>

### 13.4 Convolutional Neural Network (CNN)<sup>2</sup> (6)

Une application du Deep Learning sont les **CNN** ou Convolutional Neural Network (réseau neural convolutif). Ce sont, de nos jours, les modèles de **classification d'images** les plus performants. Les méthodes que nous utiliserons pour faire de la détection d'objets puisent leur fonctionnement initial dans ce modèle de DL. Je vais bien sûr en parler, mais avant, voyons comment fonctionne un CNN de manière succincte :

Vu qu'un CNN est un modèle de Deep Learning, il utilise aussi des réseaux de neurones artificiels. Il est capable d'être entraîné et d'apprendre en fonction de données qui lui sont données. Un CNN est donc fait de couches de neurones (ou perceptrons<sup>3</sup>) reliées entre elles et auxquels sont attribués des poids et des biais. Le but est d'extraire automatiquement les fameuses "features" des images lors de la phase d'entraînement, d'ajuster les poids et les biais afin de faire les bonnes connexions entre les neurones et de permettre ensuite des résultats probables lors de la phase de prédiction.

<sup>1</sup> [https://cdn-images-1.medium.com/max/800/1\\*ZX05x1xYgaVoa4Vn2kKS9g.png](https://cdn-images-1.medium.com/max/800/1*ZX05x1xYgaVoa4Vn2kKS9g.png)

<sup>2</sup> [https://vas3k.com/blog/machine\\_learning/#scroll230](https://vas3k.com/blog/machine_learning/#scroll230)

<sup>3</sup> <https://fr.wikipedia.org/wiki/Perceptron>

Ce réseau de neurones est composé d'une couche d'entrée (ou l'on donne une image), d'une couche de sortie (rend une prédiction) et de couches cachées entre deux. Ces couches cachées sont généralement composées de : Convolutional layers, ReLU layers, pooling layers, et fully connected layers (Explications ici : <http://cs231n.github.io/convolutional-networks/> ).

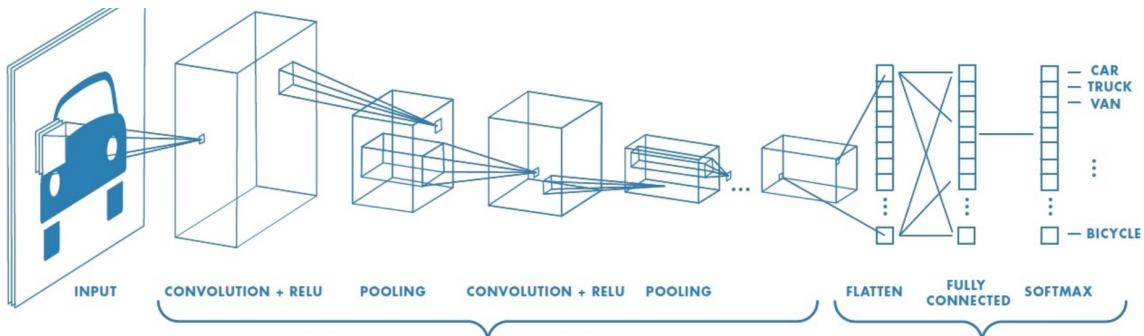


Figure 13.4 : Schématisation du fonctionnement d'un modèle CNN<sup>1</sup>

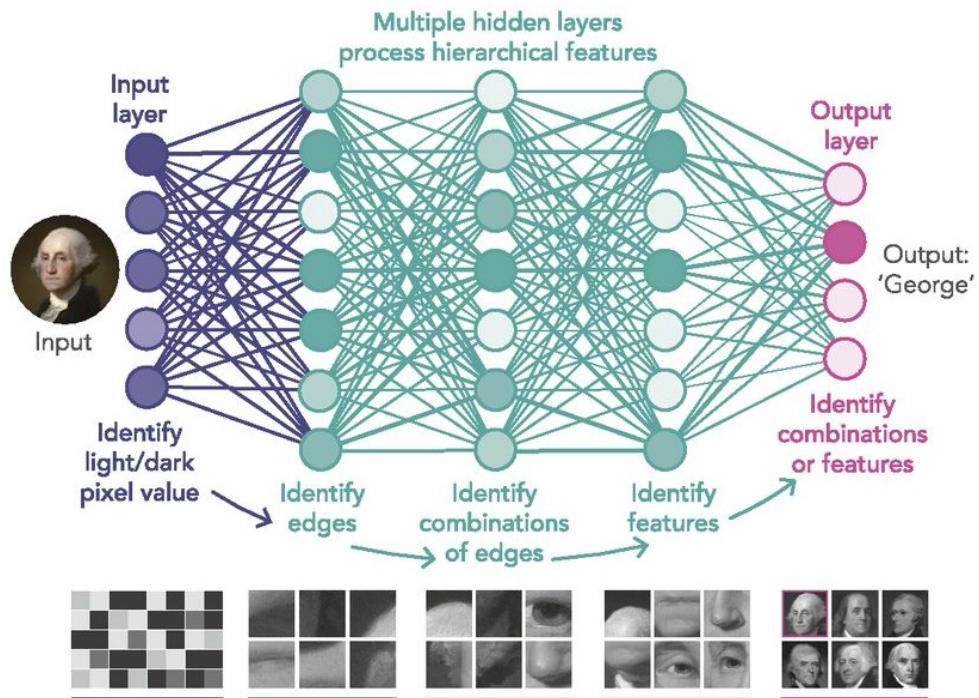


Figure 13.5 : DNN pour faire de la classification<sup>2</sup>

<sup>1</sup> <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

<sup>2</sup> <https://www.pnas.org/content/pnas/116/4/1074/F2.large.jpg?width=800&height=600&carouselSel=1>

### 13.5 Modèles pré-entraînés

Il est très complexe et long de développer soi-même un CNN et il est très couteux d'entrainer un CNN qui soit fiable et qui atteigne une efficacité de reconnaissance "state-of-the-art". Cela permet cependant de construire des modèles hautement spécifiques et permettant des résultats extrêmement fiables dans le domaine pour lequel ils ont été conçus.

C'est à cause des contraintes de temps et mon manque d'expérience dans le domaine, que nous allons utiliser des modèles **pré-entraînés**. Ce sont des modèles qui ont été précédemment entraînés sur des données similaires aux nôtres afin de résoudre un problème du même type. Nous n'aurons ainsi pas à faire la phase d'entraînement.

### 13.6 Architectures de CNN<sup>1</sup> (6)

Il existe des architectures de CNN bien connues, ce sont des modèles pré-entraînés. Elles sont toutes développées de base pour participer au challenge "ImageNet<sup>2</sup>" où le but est d'entraîner un modèle sur un grand set d'images et d'avoir la plus haute précision.

#### AlexNet

Premier système utilisant un CNN participant au challenge. Il est réputé pour avoir énormément augmenté le niveau de précision de ses prédictions grâce à cette architecture. Son succès fut le début d'une petite révolution dans le domaine, tous les participants se sont mis à utiliser des architectures CNN les années suivantes.

#### VGG by Oxford

En 2014, c'est le premier modèle qui a obtenu un taux d'erreurs inférieur à 10%. Il sert de base à de nombreux autres modèles. ResNet et Inception reprennent certaines de ses idées. Son problème est qu'il nécessite énormément de ressources, il est très lent.

#### ResNet by Microsoft

Encore une petite révolution., en 2015, ResNet introduit le "skip connection". Cela permet d'avoir des réseaux très profonds sans avoir d'énorme impact sur les performances.

---

<sup>1</sup> <https://towardsdatascience.com/neural-network-architectures-156e5bad51ba>

<sup>2</sup> <http://www.image-net.org/challenges/LSVRC/>

### Inception-v4 by Google

Gagnant en 2016, invaincu à ce jour. C'est une évolution de l'ancienne architecture de Google (GoogleLeNet, Inception-v3) qui a été mergé avec l'idée proposée par ResNet. Il est légèrement plus gourmand en ressources que RestNet, mais a une fiabilité sans égale.

### MobileNets by Google

Modèle créé spécifiquement avec une utilisation sur dispositifs mobiles et embarqués en tête. C'est un modèle plus petit et moins complexe qui lui permet d'avoir des performances acceptables sur du matériel peu puissant. Sa fiabilité reste acceptable, mais n'est clairement pas "state-of-the-art", cet aspect n'étant pas sa priorité.

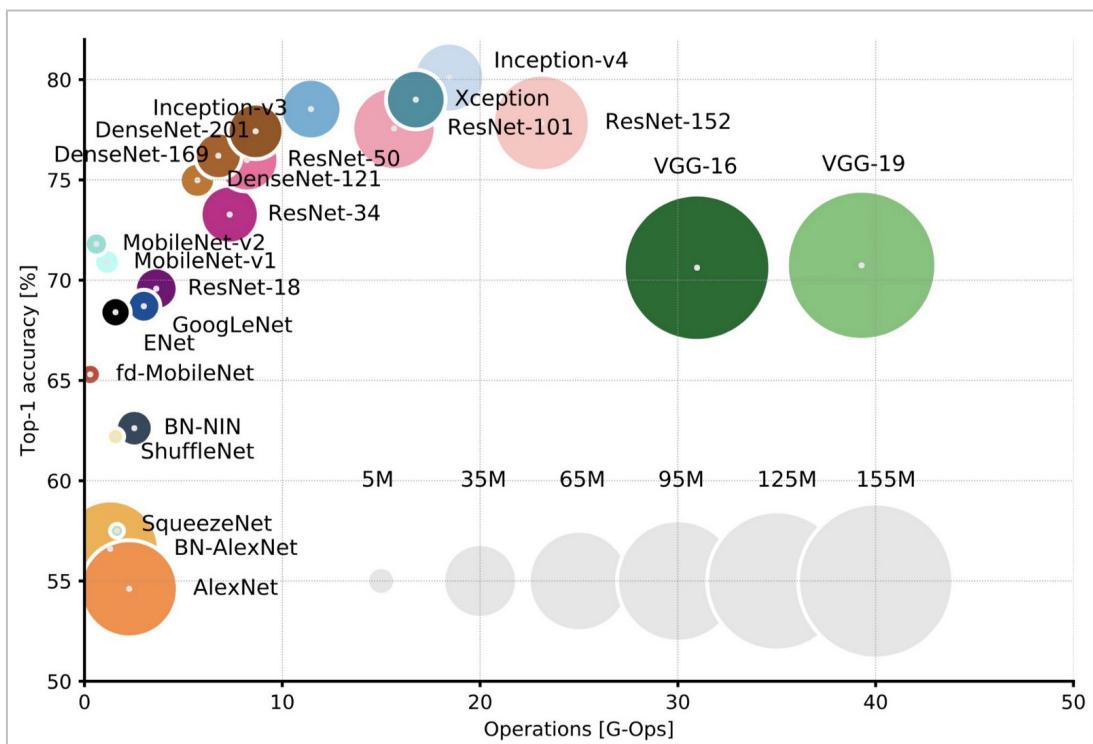


Figure 13.6 : Comparaison des différentes architectures CNN<sup>1</sup>

Nous pouvons constater que deux architectures sortent leur épingle du jeu : **ResNet et Inception**. Les différentes méthodes de détection d'objets que nous verrons ci-après les utilisent. À voir lequel peut permettre le meilleur équilibre fiabilité/vitesse.

<sup>1</sup> [https://cdn-images-1.medium.com/max/800/1\\*n16lj3lSkz2miMc\\_5cvkrA.jpeg](https://cdn-images-1.medium.com/max/800/1*n16lj3lSkz2miMc_5cvkrA.jpeg)

## 13.7 Object Detection

Nous devons, pour ce projet, mettre des "**bounding boxes**" autour d'êtres humains afin de les détecter et de les compter, ce problème fait partie du domaine de la détection d'objets. Voyons maintenant quel place ces notions de "CNN", d'"Architecture de CNN" précédemment développées ont dans le domaine de la détection d'objets.

Traditionnellement, les CNN sont utilisés pour faire de la classification d'images (attribution d'un label à une image donnée). Dans la détection d'objets, nous devons faire une classification, mais plusieurs fois sur la même image afin d'en extraire tous les objets qui y sont contenus et d'en définir l'emplacement.

Le principe succinct est de séparer l'image que nous souhaitons analyser en plusieurs zones ou "régions". Nous appliquons ensuite sur chacune de celles-ci notre algorithme de classification en mode "sliding-window".

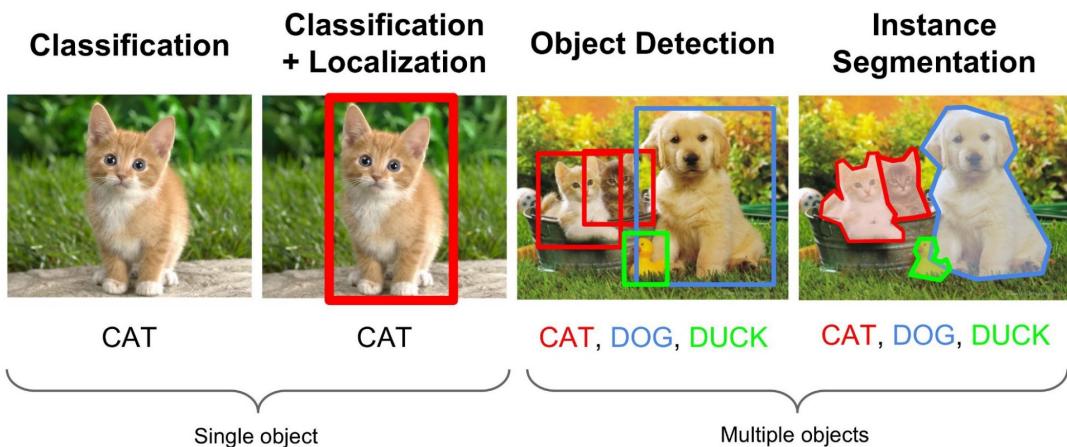


Figure 13.7 : Différence entre "classification" et "détection"<sup>1</sup>

Nous allons nous intéresser aux différentes méthodes développées autour de ce principe. Elles sont bien sûr bien plus élégantes que le principe cité ci-dessus qui serait extrêmement gourmand en ressources CPU.

<sup>1</sup> <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>

### 13.7.1 Méthodes anté-Deep Learning

Ces méthodes ont été rendues plus ou moins obsolètes par le Deep Learning. Je ne vais pas les étudier, car elles sont hors-scope pour ce projet.

- Background Subtraction
- Méthode Viola-Jones (HAAR cascade classifier)
- Histogram of Oriented Gradients (HOG)

### 13.7.2 Region-Based Object detection<sup>1</sup> (R-CNN, R-FCN)

Ce sont les premières méthodes rapides qui ont commencé à utiliser des CNN pour faire de la détection d'objets. Elles ne sont pas très adaptées à notre cas de figure, car très lentes.

#### R-CNN (Region-based CNN) (7)

Le principe est basé sur le "selective search<sup>2</sup>" et sur la propositions de "régions" :

1. Proposer de manière aléatoire des régions de différente taille sur l'image ( $\sim 2'000$ )
2. Essaye de regrouper les régions adjacentes en fonction de la couleur des pixels, de la texture, de l'intensité afin d'y trouver des objets
3. Les propositions de région sont redimensionnées pour matcher les dimensions d'entrée du CNN
4. Chaque région est passée à travers système de classification CNN
5. En sortie, une probabilité liée à un label est donnée pour chaque région au moyen d'une machine à vecteur de support (SVM<sup>3</sup>)

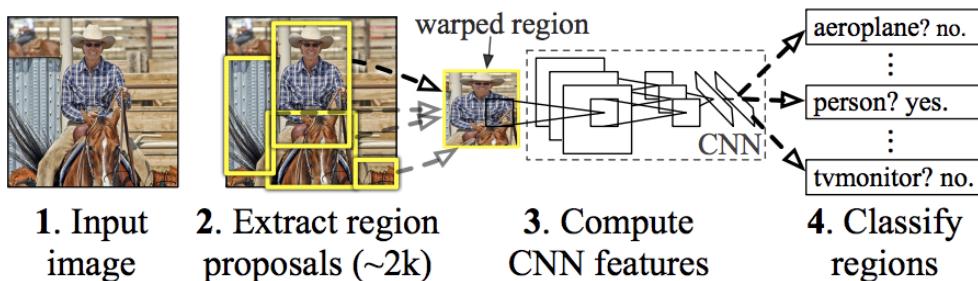


Figure 13.8 : Schéma de fonctionnement de R-CNN

<sup>1</sup> <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365>

<sup>2</sup> [https://www.researchgate.net/publication/262270555\\_Selective\\_Search\\_for\\_Object\\_Recognition](https://www.researchgate.net/publication/262270555_Selective_Search_for_Object_Recognition)

<sup>3</sup> [https://en.wikipedia.org/wiki/Support-vector\\_machine](https://en.wikipedia.org/wiki/Support-vector_machine)

La méthode fonctionne bien est cependant encore très lente et rend difficile la mise en place pour des applications "temps réel". La cause de cette lenteur est que nous devons lancer le CNN sur 2'000 régions pour chaque image.

### **Fast R-CNN, Faster R-CNN (8) (9)**

Évolutions et simplifications de R-CNN permettant d'améliorer les performances.

- **Fast R-CNN** : Utilise, après le Selective Search, un CNN sur toute l'image pour en extraire les features (Region of Interest Pooling). On utilise le CNN une fois au lieu de 2'000.
- **Faster R-CNN** : S'affranchit du Selective Search qui est lent au profit d'un algorithme de détection d'objets (Region Proposal Network).

### **R-FCN (Region-based Fully Convolutional Networks) (10)**

Détecteur d'objet basé, comme R-CNN, sur la proposition de région, mais modifie son architecture pour en proposer une composée uniquement de couches convolutives. R-FCN utilise une couche de convolution pour faire la proposition de zones.

Cela permet d'améliorer drastiquement la rapidité (2/3x plus rapide que Faster R-CNN) et on obtient la même fiabilité que Faster R-CNN.

### **13.7.3 Single Shot Object Detection (SSD, Focal Loss, YOLO)<sup>1</sup>**

Toutes les implémentations de R-CNN se basent sur un système de propositions de région suivie d'un CNN de classification. Ces approches sont fiables, mais sont encore assez limitées en termes de performances en "temps réel".

Il nous faut donc d'autres méthodes qui nous permettraient de le faire à une vitesse proche de la réalité (min. 10-30 FPS) en sacrifiant peu de fiabilité.

---

<sup>1</sup> <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852>

### SSD (Single-shot MultiBox Detector) (11)

Comme son nom l'indique, le Single Shot Detector prédit les "bounding boxes" ainsi que la classe en **une seule étape** contrairement à deux comme le fait R-CNN. Il était meilleur que YOLO tant en termes de performances que de fluidité avant que la version 2 de YOLO ne sorte. Il sacrifie un peu de performances de détection en faveur de la vitesse de détection.

SSD utilise la technique de MultiBox<sup>1</sup>. Il prend l'image en entier en input et la fait passer à travers de multiples couches de convolution de différente taille. Fonctionnement :

1. Extraction d'une grille (feature map) et
2. Application d'un filtre de convolution sur chaque case pour détecter les objets

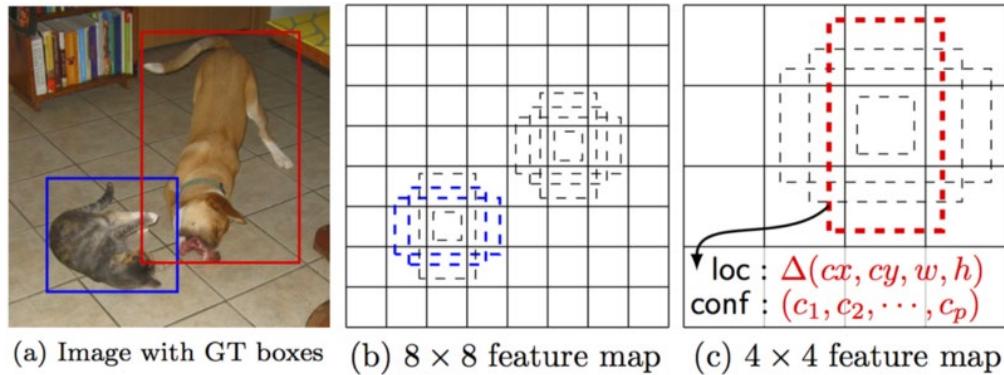


Figure 13.9 : Image originale et prédictions pour chaque case de la grille

### DSSD, FSSD, RSSD

Améliorations de SSD visant à améliorer la fiabilité au moyen de diverses techniques, le tout sans trop impacter la vitesse d'exécution.

### Focal Loss (RetinaNet) (12)

Offre une fiabilité très bonne dans la veine de détecteurs "deux phases" en ayant une approche "single shot". RetinaNet se base sur l'architecture "Feature Pyramid Network" (FPN). Ses performances en "temps réel" ne sont cependant pas à la hauteur de SSD.

<sup>1</sup> <https://arxiv.org/abs/1412.1441>

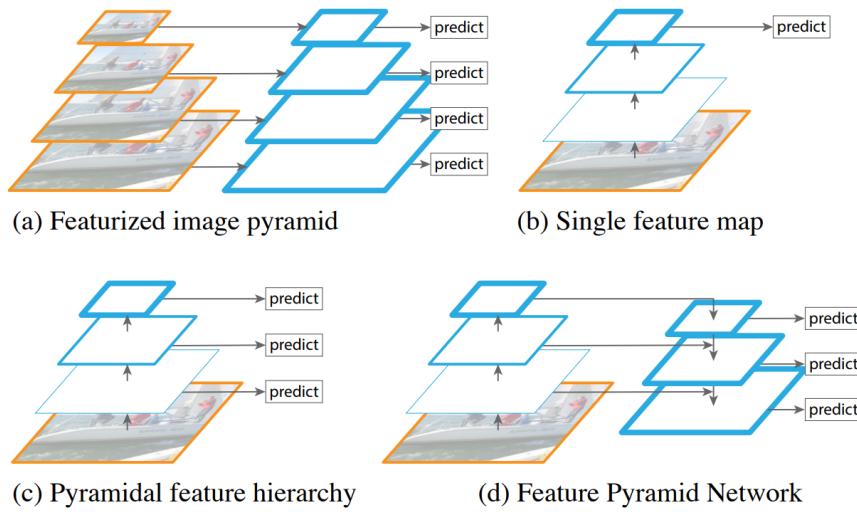


Figure 13.10 : Différents types de Feature Pyramid Network<sup>1</sup>

### 13.7.4 YOLO (You Only Look Once)<sup>2</sup> (13)

Système "Single Shot" et "State-of-the-art" pour la détection d'objets en termes de performances (FPS) sur GPU et en termes de fiabilité (au coude à coude avec RetinaNet, mais 3.8x plus rapide) et certainement le modèle le plus populaire. C'est également une méthode "single shot" très similaire à SSD. Il était initialement assez mauvais en fiabilité, misant tout sur les performances, mais il a été soigneusement mis à jour au fil des années jusqu'à en faire un système de détection d'objets à la pointe du marché.

Son fonctionnement<sup>3</sup> est le suivant :

1. Divise l'image en une grille de SxS cellules
2. Chaque cellule prédit cinq "bounding boxes" (rectangle qui entoure un objet)
3. La cellule fait pour chaque "bounding box" une prédiction sur son contenu et un score de confiance y est attribué
4. On élimine les "bounding boxes" qui n'ont pas une confiance suffisante

<sup>1</sup> [https://medium.com/@jonathan\\_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d](https://medium.com/@jonathan_hui/what-do-we-learn-from-single-shot-object-detectors-ssd-yolo-fpn-focal-loss-3888677c5f4d)

<sup>2</sup> <https://pjreddie.com/darknet/yolo/>

<sup>3</sup> [https://medium.com/@jonathan\\_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088](https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088)

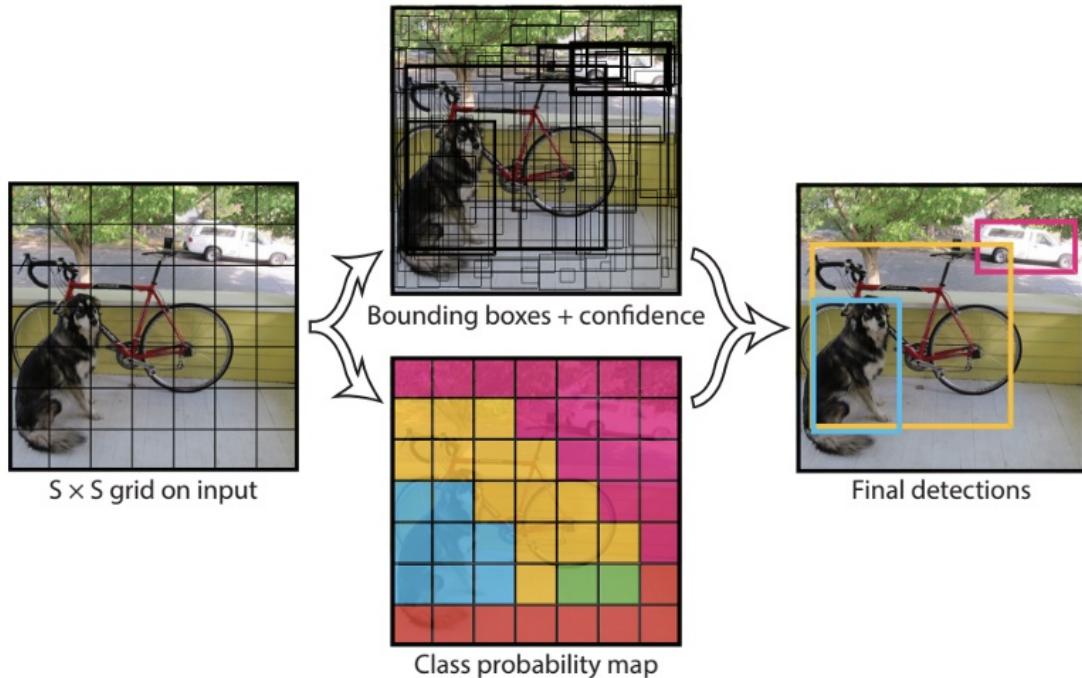


Figure 13.11 : Schéma du fonctionnement de YOLO

## YOLOv2, YOLOv3<sup>1</sup> (14) (15)

Évolutions et améliorations permettant d'améliorer de manière significative les performances ainsi que la fiabilité.

- **YOLOv2** : Mise à jour centrée sur la fiabilité au moyen de : Batch Normalization, High-Resolution Classifier, Convolutional with Anchor Boxes...
- **YOLOv3** : Mise à jour visant à améliorer la vitesse et la fiabilité. Utilise une nouvelle architecture de CNN (darknet-19).

### YOLOv3-tiny / YOLOv3-spp

Variantes de YOLO visant respectivement à :

- **YOLOv3-tiny** : proposer un modèle de petite taille ayant une fiabilité inférieure, mais étant 5x plus rapide.
- **YOLOv3-spp** : meilleure fiabilité au moyen de "spatial pyramid pooling"

<sup>1</sup> <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b>

### 13.7.5 Résumé

Le schéma ci-dessous résume bien l'évolution des systèmes de détection d'objets ainsi que les deux grandes "branches" soit les systèmes en deux phases (Region Proposal Based) et les systèmes en une phase (Regression/Classification Based)

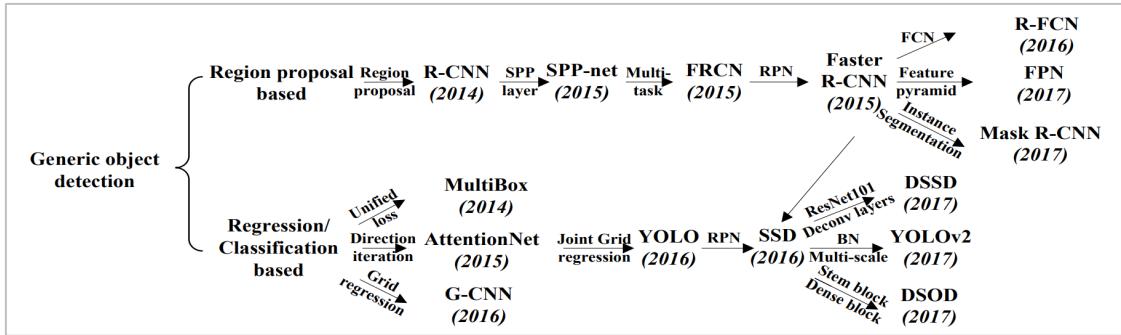


Figure 13.12 : Schéma résumant l'évolution des modèles de détection d'objets (16)

### 13.7.6 Comparaison des performances

Pour notre projet, il nous faut un système qui soit fiable et rapide. Fiable afin de ne pas enregistrer de mesures erronées dans la base de données et rapide afin d'avoir assez de FPS pour assurer le tracking des personnes et permettre l'utilisation sur un dispositif embarqué à puissance limitée.

Nous pouvons donc retirer les architectures RetinaNet ainsi que toutes les variantes de R-CNN des choix, car leurs performances en FPS ne sont pas suffisantes pour notre application "temps réel" et ce sont des modèles beaucoup trop lourds demandant une grande puissance de calcul afin d'être mis en œuvre correctement sur du hardware peut couteux. Il nous reste donc les algorithmes de tracking basés sur SSD ou YOLO.

Le graphique ci-dessous tiré d'une étude de Google montre bien la supériorité des architectures SSD quand il s'agit de proposer un système peu coûteux en performances. Il est également bon de noter que les performances sont modifiées en fonction de l'architecture CNN utilisée ("Feature Extractor"). MobileNet étant moins fiable, mais également plus léger comparé à Inception ou ResNet.

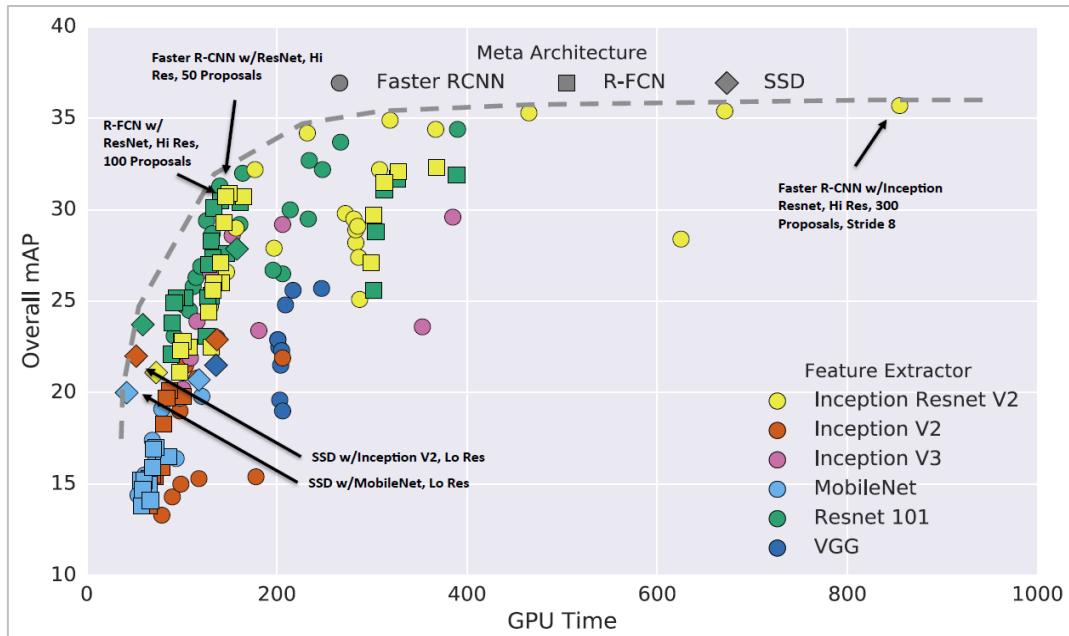


Figure 13.13 : Résumé des performances des différentes architectures (hors YOLO) (17)

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20

Figure 13.14 : Comparaison des performances des différents modèles sur GPU Titan X<sup>1</sup>

L'image ci-dessus compare les performances en termes de FPS et de fiabilité de différents systèmes de détection. Les tests étant effectués par les développeurs de Darknet sur un GPU, ils ne sont pas représentatifs des performances que nous aurons sur notre système. Nous pouvons cependant nous en servir pour comparer les modèles et nous aider à faire un choix pour notre implémentation.

<sup>1</sup> <https://pjreddie.com/darknet/yolo/>

Une autre étude a comparé ces performances et les résultats sont cohérents avec les résultats obtenus par l'équipe qui développe Darknet.

Method	Input	mAP (%)	FPS
Faster R-CNN [9] (VGG16)	-	72.70	11.23
YOLO [11]	448 × 448	62.52	42.34
YOLOv2 [24]	416 × 416	73.82	64.65
YOLOv2 544 × 544 [24]	544 × 544	75.96	39.14
SSD300 [12]	300 × 300	74.18	58.78
SSD512 [12]	512 × 512	76.83	27.75
DSSD (ResNet-101) [25]	321 × 321	76.03	8.36
R-SSD300 [13]	300 × 300	75.02	43.78
R-SSD512 [13]	512 × 512	77.73	24.19
RefineDet320 [26]	320 × 320	76.97	46.83
RefineDet512 [26]	512 × 512	77.68	29.45
SIN [27]	-	77.26	10.79
YOLOv3 [29]	416 × 416	88.09	51.26
Ours (DP-SSD300) *	300 × 300	75.43	50.47
Ours (DP-SSD512) *	512 × 512	77.94	25.12

Figure 13.15 : Étude comparant les différentes méthodes (taille de l'image en entrée, fiabilité et FPS) (18)

Maintenant que nous avons une meilleure idée des performances des différents systèmes, nous pouvons poser des critères et les noter en fonction des informations qui ont été récoltées. Le but est de faire un choix de la technologie à implémenter lors de la réalisation.

### Critères (dans l'ordre d'importance) :

- **Fiabilité** : Le moins d'erreurs possible
- **Vitesse** : Le plus de FPS possibles sur un hardware donné (CPU uniquement)

Architecture	Fiabilité	Vitesse (CPU)
SSD-MobileNet	-	++
SSD-Inception	+	+
SSD-ResNet	+	-
YOLOv3	++	--
YOLOv2	+	-
YOLOv3-tiny	--	++

("+" signifie qu'un système est mieux dans le critère et "-" qu'il performe moins bien)

Nous constatons que **YOLOv3** est le modèle de détection d'objets à favoriser dans notre approche, il est de loin de plus fiable. Si nous observons cependant qu'il n'offre pas une vitesse suffisante, nous pourrons nous reposer sur une architecture SSD pour parvenir à nos fins.

Dans la réalisation, les deux méthodes seront testées et évaluées (YOLO et SSD) afin de définir laquelle proposer le meilleure rapport fiabilité/vitesse et laquelle approche est la meilleure dans un cas donné.

## 13.8 Object Tracking<sup>1</sup> (19)

Nous avons délimité les personnes dans notre image au moyen de "bounding boxes" générées grâce à YOLO ou SSD qui font de la détection d'objet. Il faut récupérer les données en sortie et les utiliser pour suivre des personnes au fur et à mesure des trames et détecter si elles sortent ou entrent dans le cadre afin de tenir à jour un compteur.

### 13.8.1 Multiple Object Tracking<sup>2</sup>

Ce domaine s'appelle le "Multi-Object Tracking". Il vise à tracer le parcours de plusieurs objets de même type en mouvement en les identifiant indépendamment. C'est fondamentalement différent de l'Object Detection, car ici, le but est de détecter un élément et de le suivre sur tout le long de son parcours grâce à un ID qui lui est propre.

Il est important de prendre en compte que le flux vidéo IP que nous récupérons nous permet d'avoir uniquement l'image actuelle et l'image précédente afin de faire ce tracking, il faudra donc choisir un algorithme qui satisfait à cette contrainte. Ce type de tracking est appelé "Online" au contraire du tracking "Offline" où tous les frames de la vidéo sont disponibles.

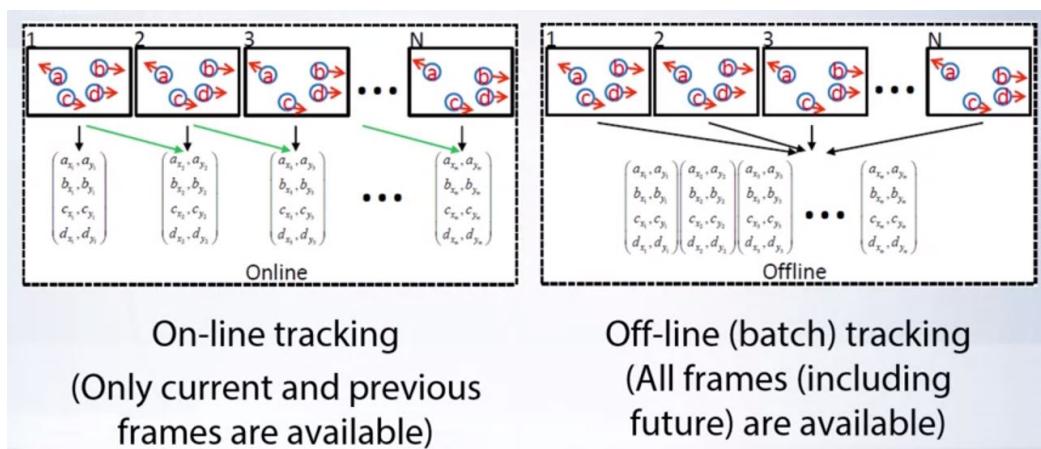


Figure 13.16 : Tracking online et offline<sup>3</sup>

<sup>1</sup> <https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/>

<sup>2</sup> <https://motchallenge.net/>

<sup>3</sup> <https://fr.coursera.org/lecture/deep-learning-in-computer-vision/examples-of-multiple-object-tracking-methods-VJZUW>

Le bon fonctionnement d'un tracker dépend intimement de la fiabilité d'un système de détection d'objets. Un bon tracker devrait ainsi être capable de gérer les défauts de la détection d'objets. Plusieurs challenges peuvent être rencontrés :

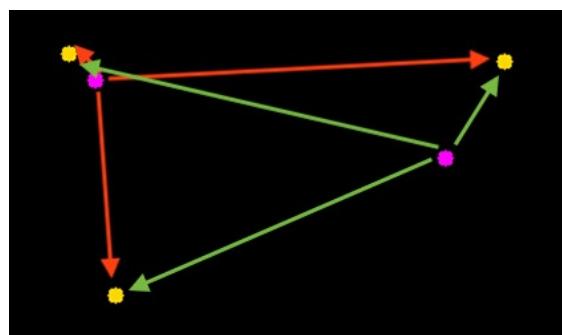
- Remplir les trous dans la détection
- Gérer l'occlusion lorsque des objets sont proches et se cachent
- Gérer les mouvements rapides (dépend des FPS de la détection)
- Gérer les changements de forme des objets non rigides
- Ignorer les faux positifs

Une méthode intuitive de tracker une personne en mode "online" est la suivante :

1. Sur une frame, identifier avec un ID la bounding box des personnes détectées.
2. Utiliser un algorithme pour que l'ID reste "attaché" à la bonne personne au fil du flux vidéo en utilisant que la frame actuelle et la précédente.
3. Une fois la personne arrivée sur le bord du cadre ou à une frontière, on peut l'identifier grâce à son ID et effectuer une action (comme incrémenter un compteur).

### 13.8.2 Euclidean Distance Algorithm<sup>1</sup>

C'est une méthode qui repose uniquement sur la distance qui sépare le centre des BBox ("centroid") d'une frame à l'autre. La distance Euclidienne est la distance en ligne droite ou à "vol d'oiseau" entre deux points d'un espace. L'algorithme calcule la distance Euclidienne entre toutes les paires (frame précédente et frame actuelle) d'éléments présents sur le plan et assigne l'objet possédant la plus petite distance comme étant le même objet. S'il ne trouve pas un objet assez prêt, il crée un nouveau label et l'assigne.



*Figure 13.17 : Illustration de la distance Euclidienne. Frame précédente (points violets) et frame actuelle (points jaunes)*

<sup>1</sup> [https://medium.com/@manivannan\\_data/object-tracking-referenced-with-the-previous-frame-using-euclidean-distance-49118730051a](https://medium.com/@manivannan_data/object-tracking-referenced-with-the-previous-frame-using-euclidean-distance-49118730051a)

C'est un algorithme simple à mettre en place, mais qui est très basique. Son fonctionnement est assez "bête", dans le sens que si un objet est perdu, on l'oublie et on lui assigne un nouveau label. Il n'y a aucun mécanisme pour, par exemple, prendre en compte l'occlusion ce qui peut amener à un système qui ne présente pas une fiabilité suffisante.

### 13.8.3 IoU (Intersection over Union)<sup>1</sup>

Cette méthode est une métrique d'évaluation qui consiste à calculer le pourcentage de chevauchement entre deux bounding boxes de deux frames pour définir si elles sont liées ou non (grâce à un threshold). Présente les mêmes limitations que la distance Euclidienne.

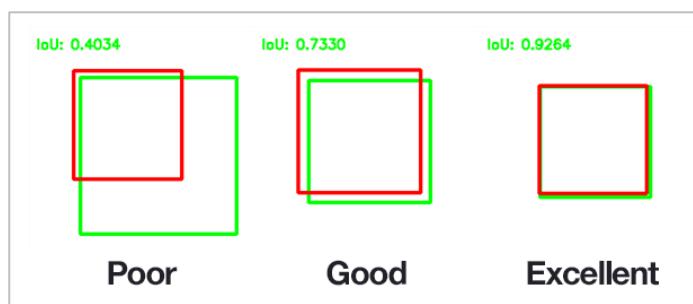


Figure 13.18 : Exemples de "score" IoU pour différentes bounding boxes

### 13.8.4 Correlation Filters

La corrélation est la mesure de similarité entre deux images, on peut utiliser un système pareil afin de mesurer les similarités entre deux Bounding Boxes de deux frames et les associer ou non en fonction du résultat. Pas de CNN ici, seulement des opérations mathématiques. Des algorithmes populaires utilisant cette mesure sont : MOSSE, KFC, CSRT.

### 13.8.5 SORT (Simple Online and Realtime Tracking)<sup>2</sup> (20)

Système de tracking d'objet visant à fournir des performances meilleures et à régler le problème de l'occlusion en combinant deux algorithmes :

- **Hungarian Algorithm** : Permet de matcher un objet de la frame actuelle avec un objet de la frame précédente au moyen d'un score (IoU est utilisé).
- **Kalman Filter** : Permet de prédire la position future en fonction de la position actuelle. Utilise le sens et la vitesse du déplacement d'un objet.

<sup>1</sup> <https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/>

<sup>2</sup> <https://github.com/abewley/sort>

En cas de perte de détection d'un objet, on utilise le filtre de Kalman pour continuer à le tracker de manière fictive sur plusieurs frames en espérant qu'il réapparaîsse et qu'on puisse le réattraper. La combinaison de ces deux algorithmes fait sa vraie force.

### 13.8.6 Deep SORT<sup>1</sup> (21)

Amélioration de SORT qui ajoute un système d'évaluation de similarités de l'apparence des objets pour aider à faire le tracking.

On lance un CNN sur les bounding boxes afin d'en calculer les "Deep Features". On les utilise ensuite pour évaluer les similarités entre les différents objets détectés et les lier au fil des frames. Il est important de noter que l'efficacité de ce système décroît grandement si les personnes sont habillées de la même manière.



Figure 13.19 : Visualisation du fonctionnement du système de similarités d'apparence de Deep SORT<sup>2</sup>



Figure 13.20 : Tracking et identification individuelle des personnes avec Deep Sort<sup>3</sup>

<sup>1</sup> [https://github.com/nwojke/deep\\_sort](https://github.com/nwojke/deep_sort)

<sup>2</sup> <https://fr.coursera.org/lecture/deep-learning-in-computer-vision/examples-of-multiple-object-tracking-methods-VJZUW>

<sup>3</sup> <https://www.youtube.com/watch?v=bkn6M4LAoHk>

### 13.8.7 Comparaison des performances

Malheureusement pas de benchmark des performances pour la distance Euclidienne ou pour les Correlation Filters. Dans l'ordre du meilleur au moins bon : Deep SORT, SORT puis IoU. Il est intéressant de noter que plus l'algorithme de tracking est performant en termes de fiabilité et de précision, plus il nécessite de puissance de calcul et donc le moins de FPS il produira sur un hardware donné. Deep SORT produit par contre un nombre important de faux positifs comparé à ses rivaux.

	<b>MOTA</b>	<b>MOTP</b>	<b>MT</b>	<b>ML</b>	<b>FP</b>	<b>FN</b>	<b>ID sw</b>	<b>Frag</b>	<b>Hz</b>
<b>SORT</b>	59.8	79.6	25.4%	22.7%	<b>8698</b>	63245	1423	<b>1835</b>	<b>59.5</b>
<b>Deep SORT</b>	<b>61.4</b>	79.1	<b>32.8%</b>	<b>18.2%</b>	12852	<b>56668</b>	<b>781</b>	2008	40
<b>IOU</b>	57.1	77.1	23.6%	32.9%	<b>5702</b>	70278	2167	3028	<b>3004</b>

Figure 13.21 : Comparaison des performances<sup>1</sup>

Légende :

- **MOTA** : Object Tracking Accuracy (Perfect : 100%)
- **MOTP** : Object Tracking Precision (Perfect : 100%)
- **MT** : Mostly tracket targets (Perfect : 100%)
- **ML** : Mostly lost target (Perfect : 0%)
- **FP** : False postitives (Perfect : 0)
- **FN** : False négatives (missed targets) (Perfect : 0)
- **ID sw** : Identity switches (Perfect : 0)
- **Frag** : Number of time a trajectory in interrupted during tracking (Perfect : 0)
- **Hz** : Processing speed or FPS (Perfect :  $\infty$ )

Dans la réalisation, nous allons en premier mettre en place une solution basique et peu gourmande basée sur la distance Euclidienne ou sur IoU, évaluer les performances et ensuite améliorer le système en fonction des besoins. C'est-à-dire, tester les performances que pourraient offrir SORT, Deep SORT ou Correlation Filters.

<sup>1</sup> <https://motchallenge.net/results/MOT16/>

### 13.9 Système de comptage

Maintenant que nous possédons un système de suivi, nous pouvons utiliser les données de celui-ci afin de créer une logique de comptage basée sur les déplacements.

Plusieurs logiques de comptage peuvent être imaginées. La méthode qui est préconisée pour ce genre de tâche est de dessiner une ou deux "lignes" (ou frontière) sur l'image et de surveiller si un objet la traverse ou non et dans quel sens. On incrémente ou décrémente ensuite un compteur en fonction du sens de traverse de la frontière.

L'algorithme qui sera utilisé pour le comptage sera plus développé dans la partie Conception et Réalisation.

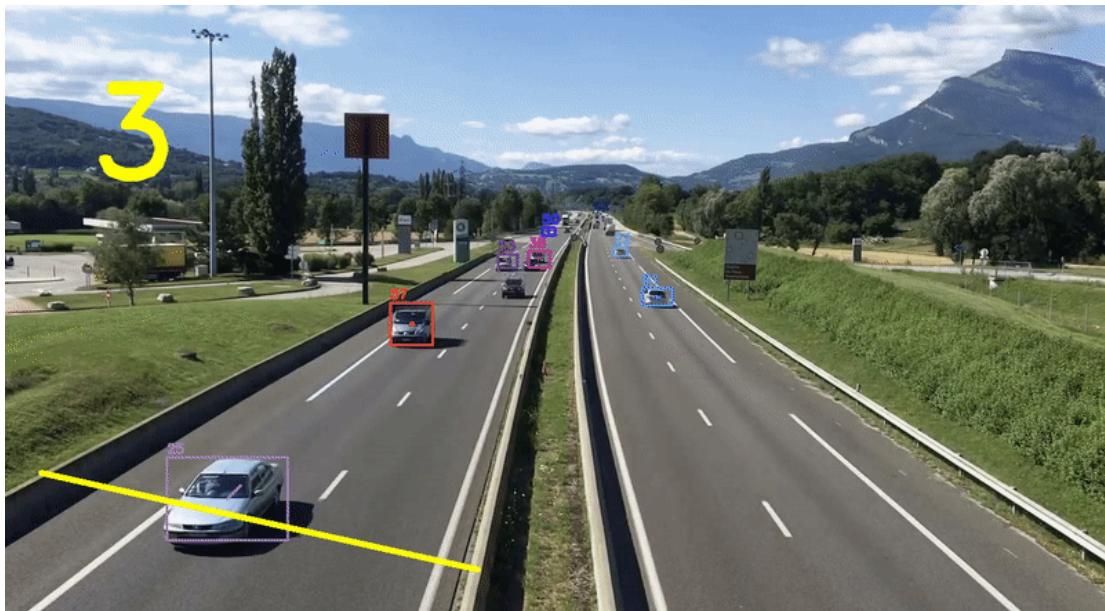


Figure 13.22 : Exemple de mise en pratique de comptage unidirectionnel utilisant une frontière<sup>1</sup>.

<sup>1</sup> <https://github.com/guillemlopez/python-traffic-counter-with-yolo-and-sort>

## 13.10 Datasets d'entraînement

Les modèles pré-entraînés que nous utiliserons ont été entraînés sur des sets de données. Ces Dataset contiennent un grand nombre d'images d'objets quotidiens qui ont été labellisées afin de faire de l'entraînement supervisé. Les Datasets principaux utilisés pour entraîner les modèles de détection d'objets sont les suivants :

### 13.10.1 COCO<sup>1</sup>

Développé par Microsoft, Dataset de grande échelle. Est destiné à l'entraînement et l'évaluation de modèles faisant de la classification, de la détection et de la segmentation. Il contient plus de 200'000 images labellisées, 80 catégories d'objets, 250'000 images de personnes... C'est le Dataset utilisé pour les modèles pré-entraînés de **YOLO** et l'implémentation par Tensorflow de **SSD**.

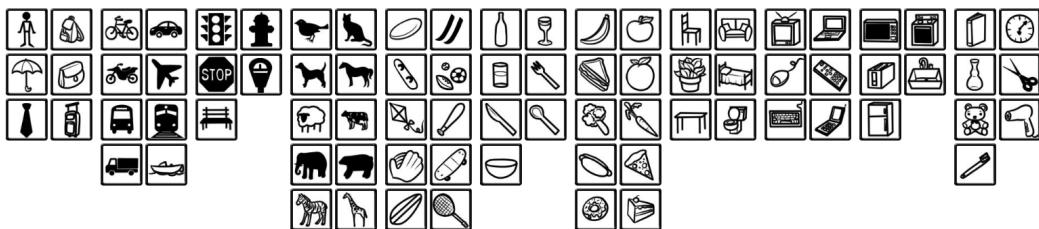


Figure 13.23 : Visualisation des différentes catégories que COCO propose

### 13.10.2 Pascal VOC<sup>2</sup>

Fournit un ensemble d'images normalisées pour la reconnaissance de classes d'objets, la détection et la segmentation. Contient 20 classes d'objets communs. Il y a environ 10'000 images pour l'entraînement et la validation avec des bounding boxes autour des objets.

### 13.10.3 ImageNet Bounding Boxes<sup>3</sup>

Set de données de 500'000 images contenant un objet et des bounding boxes. 200 Catégories sont disponibles. Est rarement utilisé à cause de sa taille déraisonnée nécessitant une puissance de calcul très importante.

<sup>1</sup> <http://cocodataset.org/#home>

<sup>2</sup> <http://host.robots.ox.ac.uk/pascal/VOC/>

<sup>3</sup> <http://image-net.org/download-bboxes>

## 14 Problèmes potentiels

---

Il faut mentionner les risques encourus, cela afin de supporter nos décisions et d'être conscient des problèmes potentiels que nous pourrons rencontrer lors des phases de réalisation et d'exploitation.

### **Le streaming du flux vidéo depuis le Raspberry Pi pose un problème**

Une image qui n'a pas une résolution suffisante ne permettra pas de faire une détection d'objet correctement sur celle-ci et une vidéo qui n'a pas assez de FPS rendra le tracking difficile, les "sauts" des personnes entre les images étant plus grands, on les perd plus facilement. Un flux instable ou qui fluctue beaucoup en qualité est aussi problématique.

Je suis cependant confiant que UV4L nous permet d'avoir un streaming d'assez bonne qualité pour notre système. Dans le cas contraire, il faudra se tourner vers un autre système de streaming voire même vers de l'Edge Computing.

### **Le matériel n'est pas assez puissant pour faire les tâches de détection**

Il faudra, soit réduire la résolution dans la limite de l'acceptable pour YOLO, soit se fournir une workstation plus puissante. Il est également envisageable de passer sur un autre système de détection tel que SSD en acceptant une baisse de fiabilité.

J'ai pris soin de choisir, dans cette analyse technologique, des solutions qui ont un bon rapport performances/fiabilité dans le seul but de prévenir ce genre de problème. Il ne devrait donc pas survenir et le système pourrait même tourner sur du matériel moins puissant en Edge Computing.

### **Le système de tracking n'a pas des performances suffisantes**

L'occlusion est peut-être trop importante, des personnes sont cachées par d'autres et elles ne sont donc pas suivies. Il faudrait changer la caméra de place. Le tracking dépend également beaucoup de la qualité et de la consistance du système de détection d'objets. Il faudrait songer à le changer s'il ne satisfait pas nos attentes.

## **Le système de détection de personnes YOLO n'a pas une fiabilité suffisante et manque beaucoup de personnes**

Il faudrait songer à peut-être ré-entrainer YOLO avec un dataset correspondant plus au cas d'application ou limiter les classes reconnaissables. Il faudrait aussi essayer de changer l'angle de la caméra qui capture la scène, le tout en essayant quand même de la positionner afin d'éviter l'occlusion de certaines personnes.

---

## **Synthèse**

Nous avons maintenant un large éventail des technologies existantes afin de mener à bien notre projet. Nous savons vers lesquels nous tourner et avons des alternatives en cas de problème.

Le workflow hypothétique qui ressort de cette analyse sera donc le suivant :

- Capture de vidéo sur un RPI
- Envoie du flux sur le LAN avec UV4L
- Récupération du flux vidéo via une adresse IP avec OpenCV
- Object Detection avec YOLO ou SSD sur les frames du flux vidéo
- Object Tracking avec Euclidean Distance sur les bounding boxes de la détection
- Object counting avec les données de la détection et du suivi sur OpenCV avec la méthode de "frontière"
- Envoie des données de comptage sur BBData avec une requête HTTP Post depuis Python

Pour ce travail, je vais surtout utiliser des librairies et des méthodes existantes afin de gagner du temps, ne pas réinventer la roue. Tout réimplanter moi-même me permettrait d'apprendre énormément de choses, mais le temps est compté et il faut absolument fournir un "Proof of Concept" sur lequel itérer.

## IV. ANALYSE ÉCONOMIQUE

---

### Introduction

Une analyse économique figure au cahier des charges afin de juger du potentiel du produit qui sera développé en dehors du contexte de City Pulse. Il y a plusieurs aspects auxquels il faut s'intéresser afin de positionner un projet sur le marché, d'exposer les problèmes qu'il résout et de démontrer ou non sa nécessité et sa réponse à une demande.

Les aspects suivants vont être au cœur de cette partie :

- Quel est le cadre légal de la protection des données et de la vidéosurveillance ?
- Quelles offrent similaires existent sur le marché, quel serait notre concurrence ?
- Quelle est la clientèle possible, qu'elle est la proposition de valeur du produit ?

### 15 Réglementation

---

Avant de se plonger dans le cas d'application spécifique du projet, il est nécessaire de comprendre le cadre légal général dans lequel nous évoluons. En Suisse, l'utilisation de dispositifs de vidéosurveillance tombe sous la loi fédérale sur la protection des données<sup>1</sup> et sous la RGPD en Europe. Je vais en étudier les objectifs et voir leurs implications dans mon projet.

Afin de me permettre de juger au mieux les exigences pour ce projet sur le cadre légal, j'ai contacté différents experts afin de diriger mes réflexions :

- Monsieur Bacher Jean-Philippe, professeur à la HEIA-FR
- Madame Dougoud Maya, juriste de la HES-SO
- Monsieur Michael Koch, Regional Account Manager chez XOVIS
- Monsieur Fabian Sipp, CTO de Morphean

---

<sup>1</sup> <https://www.edoeb.admin.ch/edoeb/fr/home/protection-des-donnees/dokumentation/feuilles-thematiques/videosurveillance-effectuee-par-des-particuliers.html>

## 15.1 Protection des données<sup>1</sup>

À quoi servent les données et pourquoi les protéger ? Déjà, que pouvons-nous qualifier de donnée personnelle ? Ce sont simplement toutes les informations qui se rapportent à une personne identifiée ou identifiable, qu'elle soit physique ou morale. Ces données sont diverses et variées : adresse du logement, historique d'achats, musique et contenu consommé en ligne, historique des positions (géolocalisation), etc... Ces données forment une "persona" de l'individu auxquels elles sont attachées créant une identité virtuelle de celui-ci.

Dans notre société numérique, les dispositifs de captures sont partout. Dans la plupart des cas, le consommateur n'est même pas au courant de la collecte. Les données peuvent être récoltées de plusieurs manières : cookies sur un site web, carte de membre, compte d'utilisateur, etc... Les dispositifs sont plus ou moins intrusifs et plus ou moins "fourbes" dans le sens où, par exemple, un consommateur ne se doute pas que sa carte "Cumulus" récolte des informations sur ses habitudes de consommation.

Il est important de protéger ces données, elles représentent un bien précieux dans un cadre économique. Les entreprises s'y intéressent. Les données sont une source précieuse d'informations sur leurs clients, permettant de juger précisément leurs comportements et d'adapter et cibler leur offre ou leur marketing en fonction de ceux-ci. L'appât du gain et la concurrence entraînent des débordements dans le respect de la vie privée. C'est pour cela qu'un cadre légal est nécessaire afin de protéger les citoyens des abus.

Les intérêts économiques ne sont pas les seuls. L'état et les autorités sanitaires s'intéressent aussi aux données dans le but de protéger la population. On peut citer dans ces cas de figure : la lutte contre le terrorisme, la criminalité ou la réduction des coûts de la santé. La frontière entre surveillance pour le bien de l'individu et surveillance démesurée est mince.

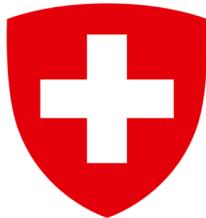
Dans tous les cas, ces données font donc partie intégrante de notre personne. Il est essentiel que nous ayons le contrôle de celles-ci dans notre société démocratique fondée sur le respect du droit humain. Le choix doit nous appartenir.

---

<sup>1</sup> <https://www.edoeb.admin.ch/edoeb/fr/home/protection-des-donnees/generalites/protection-des-donnees.html>

## 15.2 Cadre légal suisse

En Suisse, c'est la **LPD** (Loi fédérale sur la protection des données, 235.1, entrée en vigueur



le 1 juillet 1993) qui s'occupe de cadrer toute opération en relation avec les données personnelles. Cette loi fédérale repose sur plusieurs principes<sup>1</sup> qui doivent être respectés. : *"La présente loi vise à protéger la personnalité et les droits fondamentaux des personnes qui font l'objet d'un traitement de données."*<sup>2</sup>

L'article 13 de la Constitution fédérale fixe le principe de la protection de la sphère privée qui s'est naturellement étendue à la LPD : *"Toute personne a droit au respect de sa vie privée et familiale, de son domicile, de sa correspondance et des relations qu'elle établit par la poste et les télécommunications. Toute personne a le droit d'être protégée contre l'emploi abusif des données qui la concernent"*<sup>3</sup>.

### 15.2.1 La LPD et la vidéosurveillance<sup>4</sup>

La LPD est t à appliquer dans le cas de vidéosurveillance effectuée par des particuliers. Elle est valable du moment que l'on filme des êtres humains identifiables (indépendamment du fait que les images soient conservées ou non). Dans notre cas, nous allons filmer des entrées de bâtiments privés (écoles, commerces, ...), mais accessibles au public. La confédération met à disposition un aide-mémoire pour nous aider :

- Les systèmes de vidéosurveillance sont autorisés uniquement si l'atteinte à la personnalité qu'ils représentent est **justifiée par un intérêt** prépondérant.
- Ce doit être la **méthode adéquate** afin de réaliser le but souhaité. Si d'autres méthodes moins invasives ne permettent pas de l'atteindre.
- Les images doivent être **utilisées uniquement pour remplir l'objectif** visé. Ne peut pas les utiliser à des fins de marketing des images visant à assurer la sécurité.
- Le responsable du système doit prendre toutes les mesures afin de **protéger les données et le flux contre l'accès** par des personnes non autorisées.

---

<sup>1</sup> <https://www.admin.ch/opc/fr/classified-compilation/19920153/index.html#a4>

<sup>2</sup> <https://www.admin.ch/opc/fr/classified-compilation/19920153/index.html#a1>

<sup>3</sup> <https://www.admin.ch/opc/fr/classified-compilation/19995395/index.html#a13>

<sup>4</sup> <https://www.edoeb.admin.ch/edoeb/fr/home/protection-des-donnees/dokumentation/feuilles-thematiques/videosurveillance-effectuée-par-des-particuliers.html>

- Nous devons **filmer uniquement ce qui est nécessaire** pour remplir l'objectif visé. Il n'est pas permis de filmer l'espace public (rue, trottoir, ...)
- Les **principes de bonne foi, de proportionnalité et de transparence** doivent être respectés.
- Il faut un **avis indiquant préalablement la présence d'une caméra** de surveillance. Il faut également mentionner une personne de contact responsable.
- La surveillance doit être limitée aux **lieux qui sont la propriété de la personne** mettant en œuvre la surveillance. Un consentement est autrement nécessaire.
- Le nombre des personnes qui ont accès aux images doit être aussi restreint que possible

Maintenant que nous connaissons les exigences de cette loi, nous pouvons les appliquer à notre projet.

### 15.2.2 Le cas DeepCounter

Intéressons-nous maintenant à notre projet en particulier. Contrairement à un système de vidéosurveillance traditionnel, DeepCounter ne permet pas d'accéder au flux vidéo. Seul le l'algorithme de détection et de comptage l'utilise en temps réel afin d'en extraire des métadonnées anonymes. Nous ne stockons également pas d'extraits lors de l'exploitation.

#### Traitements distants

Cependant, malgré le fait que le flux ne soit pas accessible, notre système est quand même considéré comme un système de vidéosurveillance dès lors que vous faisons un "traitement" sur le flux vidéo. Dans le cas du traitement distant, nous envoyons ce flux sur une workstation.

Selon la LPD<sup>1</sup>, le simple envoi du flux à distance est considéré comme un traitement. Il faut donc, dans ce cas, se plier aux exigences mentionnées ci-dessus dans le chapitre "La LPD et la vidéosurveillance". M. Koch, Mme Dougoud et M. Bacher m'ont tous confirmé cette affirmation.

---

<sup>1</sup> <https://www.admin.ch/opc/fr/classified-compilation/19920153/index.html#a3>

### Traitemet intégré (Edge Computing)

Dans cette approche, seul un stream de données indiquant des coordonnées de BBox sera envoyé sur le LAN. Dans le cas où le tout le processing est fait au niveau du dispositif de capture (Edge Computing), que l'image ne le quitte jamais et que seul sont envoyées sur le réseau des métadonnées entièrement anonymisées, le dispositif est alors en règle. Il ne sera pas nécessaire d'en indiquer la présence et de suivre les exigences liées aux systèmes de vidéosurveillance. Les principes de la LPD doivent dans tous les cas être respectés.

C'est ce genre d'approche qu'a choisi d'implémenter la société XOVIS<sup>1</sup>. Elle permet depuis sa plateforme en ligne de paramétriser le niveau de privacité souhaité. À la responsabilité de la personne mettant en œuvre le dispositif d'ensuite le paramétrier en accord avec les lois en vigueur dans sa zone géographique.



Figure 15.1 : Niveaux de privacité proposés par XOVIS (22)

Il sera dans les deux cas du devoir de la personne mettant en place le système de garantir une implémentation SOTA en termes de cyber sécurité afin d'empêcher tout accès au flux vidéo, que ce soit directement depuis le capteur ou via le réseau.

Nous connaissons maintenant le cadre légal à respecter pour les deux architectures imaginées dans partie Analyse Technologique. Nous pouvons constater qu'un traitement intégré pour faire le comptage est plus avantageux, car il nous permet de mettre en place un système sans trop se poser de questions.

<sup>1</sup> <https://www.xovis.com/home/>

### 15.3 Europe (RGPD)<sup>1</sup>

La LDP Suisse est considérée comme dépassée par les évolutions technologiques<sup>2</sup>. En effet, sa rédaction date de 1992. En 27 ans, notre société s'est grandement numérisée et il est donc nécessaire d'en faire une révision afin de continuer à protéger les citoyens et leur permettre de disposer de leurs propres données. Ce projet est actuellement en cours.

L'Europe a été plus rapide que la Suisse. Elle a mis en vigueur la RGPD<sup>3</sup> depuis mai 2018 (Règlement général sur la protection des données, 2016/679, "relatif à la protection des personnes physiques à l'égard du traitement des données à caractère personnel et à la libre circulation de ces données"). Les Suisses sont concernés, car le règlement s'applique à toutes les données personnelles d'habitants de l'UE, que le traitement ait lieu à l'intérieur ou en dehors de celui-ci. Un français voyageant en Suisse sera donc toujours protégé par le RGPD.

Les points les plus importants de ce règlement sont :

- Récolte minimale de données
- Le droit à l'oubli
- Un consentement mieux demandé

			
<b>The Right to Be Informed</b> Individuals have a right to know who is processing their personal data	<b>The Right to Access</b> Individuals have the right to access any personal data that has been collected about them	<b>The Right to Rectifications</b> Individuals have the right to require organizations to correct inaccurate personal data	<b>The Right to Be Forgotten</b> Individuals have the right to have their personal data deleted and to prevent further collection
<b>The Right to Restrict Processing</b> Individuals have the right to require organizations to restrict the processing of specific categories of personal data	 <b>The Right to Data Portability</b> Individuals have the right to require organizations to transfer personal data to a recipient of their choice	 <b>The Right to Object</b> Individuals have the right to consent, or withdraw consent, to the processing of their personal data	 <b>Rights in Relation to Automated Decision Making and Profiling</b> Individuals have the right to opt out of the use of their personal data by automated systems, such as artificial intelligence

Figure 15.2 : Infographie résumant les points importants du RGPD<sup>4</sup>

<sup>1</sup> <https://smetille.ch/2017/12/14/la-suisse-et-le-rgpd/>

<sup>2</sup> <https://www.bj.admin.ch/bj/fr/home/staat/gesetzgebung/datenschutzstaerkung.html>

<sup>3</sup> <https://bit.ly/2wE26vL>

<sup>4</sup> <https://www.braze.com/perspectives/article/gdpr-compliance-need-to-know>

Ce nouveau règlement en vigueur en Europe ne change en rien les mesures que nous devons prendre pour nous conformer à la LPD. Il vise plutôt à protéger la population de la récolte et l'emploi abusif de leurs données privées. Nous ne récoltons rien de tout cela et aucun lien biométrique n'est fait par notre système, nous n'avons donc aucun souci à nous faire.

## 16 Étude de marché

---

Le but d'une étude de marché est de juger du potentiel d'un nouveau produit et d'analyser les forces à l'œuvre sur un marché. Le projet DeepCounter peut ne pas être réservé qu'à une utilisation dans CityPulse, bien d'autres entités peuvent en dégager des bénéfices. Nous allons voir quelles sont les choses auxquelles penser si nous souhaitions déployer le projet à une échelle industrielle, quels sont la demande du marché, les cas d'utilisation et la concurrence.

### 16.1 Offre

Notre produit offre une utilisation qui diffère des systèmes de vidéosurveillance traditionnels qui ont pour objectif principal la sécurité et la prévention. Notre système utilise certes une caméra, mais son but est d'utiliser les informations du capteur afin d'en extraire des données statistiques. Nous offrons basiquement un système permettant de compter les flux de personnes. Il indique le nombre de personnes qui ont circulé dans un sens et dans l'autre. Nous exposerons l'utilité du système dans le chapitre "Cas d'utilisation"

Deux implémentations sont possibles :

- Intégrer la solution de comptage dans du hardware existant (caméras IP déjà sur le terrain)
  - Traitement distant à favoriser
- Installation d'un système dans un environnement non équipé
  - Traitement intégré à favoriser

#### 16.1.1 Proposition de valeur

La proposition de valeur est une promesse que nous nous engageons à tenir envers nos clients. C'est ce qui fait qu'ils choisissent notre produit plutôt que celui d'un concurrent. Son but est de montrer le côté unique de notre solution, d'en prêcher les avantages et de prouver qu'il résout un problème. La proposition de valeur est également là pour donner une idée claire au client des services que nous proposons et des bénéfices qu'ils pourront en tirer.

Nous pouvons la modéliser grâce au canevas de proposition de valeur :

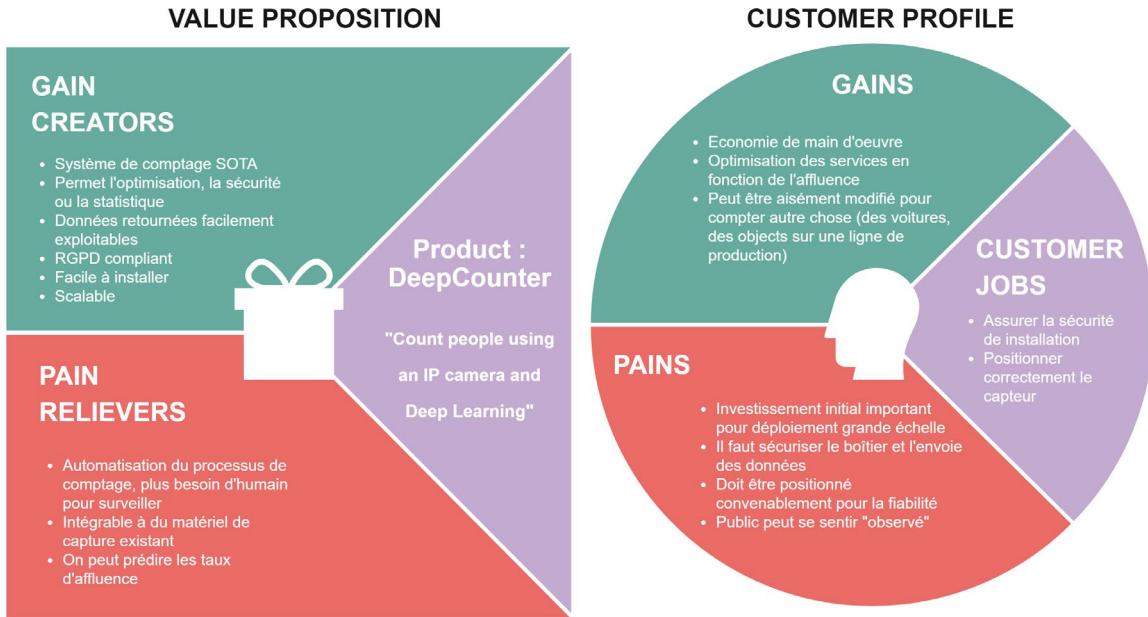


Figure 16.1 : Canvas de proposition de valeur du projet DeepCounter

Notre produit veut donc offrir aux clients un moyen de compter des objets mouvants (de haut en bas ou inversement). Le but est de savoir combien sont passés dans un sens et combien dans l'autre. On peut ainsi déduire combien de personnes se trouvent à un endroit donné. Le système se veut facilement intégrable (traitement distant) et scalable (traitement intégrer). On peut également modifier le système pour qu'il suive d'autres entités que des humains (des voitures par exemple).

## 16.2 Demande

Nous avons pour le moment modélisé notre produit et connaissons ses avantages. Il nous faut maintenant regarder du côté des consommateurs si un tel produit intéresse et quel sont ses champs d'application. Rien ne sert de mettre sur le marché un produit qui n'a aucune demande potentielle.

### 16.2.1 Ciblage de la clientèle et cas d'utilisation

Il nous faut des clients pour acheter notre produit. Il faut savoir à qui il s'adresse afin de diriger notre communication et concorder l'offre et la demande. Si on espère toucher tout le monde, on risque surtout de ne toucher personne. Une clientèle se doit d'être : Mesurable, Accessible, Exploitable et Profitable afin d'assurer la pérennité du produit.

## **Commerces**

Les services de monitoring de personnes sont en grande majorité destinés aux commerces pour qu'ils puissent améliorer leurs performances et leurs ventes. Pour qu'ils puissent optimiser leur espace intérieur et connaître le taux d'affluence au fil des heures. Ceci permettrait par exemple d'élargir un rayon où il y a beaucoup de passage ou bien d'ouvrir une caisse en plus en cas de pic d'affluence.

C'est le cas d'utilisation et la cible sur laquelle se focalisent Morphéan et XOVIS (plus de détails dans la partie "Concurrence").

## **Aéroports, gares**

Ce genre d'établissements "publics" peut bénéficier d'un système de comptage de personnes afin de mettre en évidence des zones de congestion et régler le trafic des passagers en conséquent. Compter les passagers présents dans un train et sur le quai afin d'ajouter ou supprimer des wagons au besoin ou allonger une file d'attente ou ouvrir un nouveau guichet en cas de congestion.

## **Bibliothèques (salles d'études), restaurants**

Permettraient de compter les personnes se trouvant dans le bâtiment et pourraient ainsi indiquer aux potentiels visiteurs s'il reste des places disponibles à l'intérieur ou non, par exemple au travers d'une plateforme web. Cela leur éviterait des déplacements inutiles et des déconvenues.

## **Trafic routier**

Notre système compte les personnes, mais de menues modifications lui permettraient de détecter des voitures à la place. Cela remplacerait les boudins qui sont mis sur la route actuellement afin de mesurer la densité de passage du véhicule à un endroit donné. Le système est bien moins intrusif et facilement déployable en hauteur (lampadaires, feu de signalisation)

## **Sécurité (incendie)**

Si le système est suffisamment fiable, il permettrait de dire en cas d'incendie s'il reste des personnes dans le bâtiment ou non. Aidant alors grandement le travail des pompiers et sauvant potentiellement une personne se retrouvant coincée.

## Boites de nuit

Les salles de spectacle et les boites de nuit ne peuvent pas accepter un nombre de personnes infinies en leur sein. Le vigile est en général responsable de faire respecter la législation en limitant les personnes qui entrent. Si notre système fonctionne dans l'obscurité, le vigile pourrait s'affranchir de sa tâche de comptage.

Nous avons mis en évidence que les cas d'utilisation sont nombreux et qu'une vraie utilité et un vrai bénéfice se dégagent de notre produit, permettant des économies d'argent, d'effectif humain et permettant l'optimisation. Avec un peu d'imagination, nous pourrions trouver bien d'autres champs d'applications. Le produit a donc un bon potentiel commercialisable.

## 16.3 Concurrence

Il est essentiel de connaître un marché avant de s'y lancer. Il faut savoir quels acteurs offrent un service similaire au nôtre afin de nous aider à mieux le positionner. Dans notre cas, nous n'avons pas un système spécialement novateur, des entreprises proposent des systèmes de comptage intégré à de la vidéosurveillance depuis déjà quelques années. Nous n'offrons pas un produit révolutionnaire, mais il est construit sur des technologies open-source, est relativement peu cher à l'unité et garanti le respect de la vie privée en ne diffusant pas le flux de la caméra IP.

### 16.3.1 Morphean<sup>1</sup>

Selon Morphean : "Nous offrons la meilleure solution pour une vidéosurveillance proactive, un contrôle d'accès facile d'utilisation et des analyses de business intelligence inédites dans les domaines du retail, de la banque, de la restauration, de l'hébergement, du transport, du Smart City, des gouvernements, de l'éducation, de la santé et des industries."

Leur business modèle est principalement basé sur leur cloud, ils offrent une solution de VSAAS (Vidéosurveillance as a Service). Pour la vidéosurveillance, ils utilisent les solutions de AXIS pour le hardware et le software.<sup>2</sup> Leur plus-value est dans leur plateforme web et dans tous les services qu'elle peut offrir pour optimiser des commerces.

---

<sup>1</sup> <https://morphean.com/fr/>

<sup>2</sup> <https://www.axis.com/fr-ch/products/axis-people-counter>



Figure 16.2 : Interface web du service proposé par Morphean

### 16.3.2 XOVIS<sup>1</sup>

XOVIS utilise une technologie propriétaire utilisant des caméras 3D calculant la différence de hauteur afin de détecter les personnes et les compter. Tout le processing se fait directement au sein du capteur. Leur service est dirigé envers les commerces pour optimiser leur agencement et surveiller des informations comme le taux de conversion et envers les aéroports pour gérer les files d'attente et détecter des zones de congestion.

## 16.4 Ambitions industrielles

Afin de rendre le système commercialisable, il est important d'assurer des coûts raisonnables nous permettant de dégager un chiffre d'affaires. Il faut également que le produit soit déployable de manière massive sans difficulté.

### 16.4.1 Traitement distant

Comme mentionné dans la partie Analyse Technologique : "Solutions hardware", l'approche du traitement distant n'est pas envisageable à grande échelle pour cause de limitations dues au matériel. Une instance du système de comptage devrait être lancée sur un cluster de workstation pour chaque caméra, cela rend la mise en place d'un réseau de comptage à grande échelle (tout un quartier par exemple) difficilement réalisable. Nous avons également constaté que cela nous impose diverses contraintes au niveau législatif.

<sup>1</sup> <https://www.xovis.com/solutions/detail/people-counting-dwell-time-measurement/>

Il faudrait soit avoir un datacenter avec pleins de GPU afin de traiter les centaines de flux vidéo entrants et renvoyer les métadonnées ou alors installer directement chez les clients des workstations qui se chargeront de faire le processing. Dans les deux cas, c'est difficilement réalisable pour des questions de prix, de praticité et d'entretien.

Cette solution avait initialement été imaginée afin de rendre le système facilement "pluggable" sur des installations de caméra IP existante, réduisant ainsi les travaux de déploiement et les coûts en nouveau matériel.

### 16.4.2 Traitement intégré (Edge Computing)

Le traitement intégré s'avère donc être dans ce cas, l'approche à favoriser. Elle permet un déploiement massif en ayant chaque capteur qui fait le traitement en son sein et envoie sur le réseau uniquement les métadonnées. Cette approche Edge Computing combinée à un cloud pour accéder aux données est la plus cohérente dans un cadre de commercialisation en offrant une bien meilleure scalabilité.

Il est cependant difficile d'intégrer ce système dans des installations de caméra IP existant, car le processing doit se faire au même endroit que la capture. Il faudrait donc remplacer tout le hardware déjà déployé engendrant des coûts supplémentaires en matériel et en main-d'œuvre pour le client. Nous n'avons cependant pas de workstations à acheter ce qui réduit grandement le coût.

### 16.4.3 Coûts

#### Traitement distant, coûts pour un seul capteur

Le hardware permettant la capture de données est relativement bon marché, c'est la workstation qui s'occupe du traitement qui fait décoller les prix. C'est une solution très chère pour avoir qu'une seule caméra. À tester quelle est la charge simultanée de flux vidéo que peut endurer la workstation avant de devoir en ajouter une autre. Pour cette analyse, nous considérons qu'une workstation peut avoir dix instances de OpenCV en parallèle au maximum, donc elle peut gérer jusqu'à dix capteurs.

Matériel	Prix
Raspberry Pi 3 B+	40.-
Carte micro SD 16 Go	10.-
Adaptateur POE	25.-
Caméra Fisheye Night Vision	25.-
CV GPU Workstation	2'500.-
<b>TOTAL</b>	<b>2'600.-</b>

### **Traitements intégrés (Edge Computing), coûts pour un seul capteur**

Le gain est flagrant face à l'autre solution. Même si le prix des boards Jetson Nano est trois fois plus cher qu'un RPI.

Matériel	Prix
Nvidia Jetson Nano	130.-
Carte micro SD 16 Go	10.-
Caméra Fisheye Night Vision	25.-
<b>TOTAL</b>	<b>165.-</b>

### **Comparaison des coûts**

Nous pouvons aisément visualiser les coûts en fonction du nombre de capteurs mis en service. Le traitement intégré coûte 165.- l'unité et le traitement distant coûte 100.- l'unité et tous les 10 capteurs, on assume qu'on doit ajouter une workstation à 2500.-

Coûts pour 30 capteurs :

- **Système intégré** : 4'950.-
- **Système distant** : 10'300.-

Les gains en performances de la solution distante ne sont pas suffisants pour la justifier face à la solution intégrée dans un contexte économique où il faut rendre le système rentable.

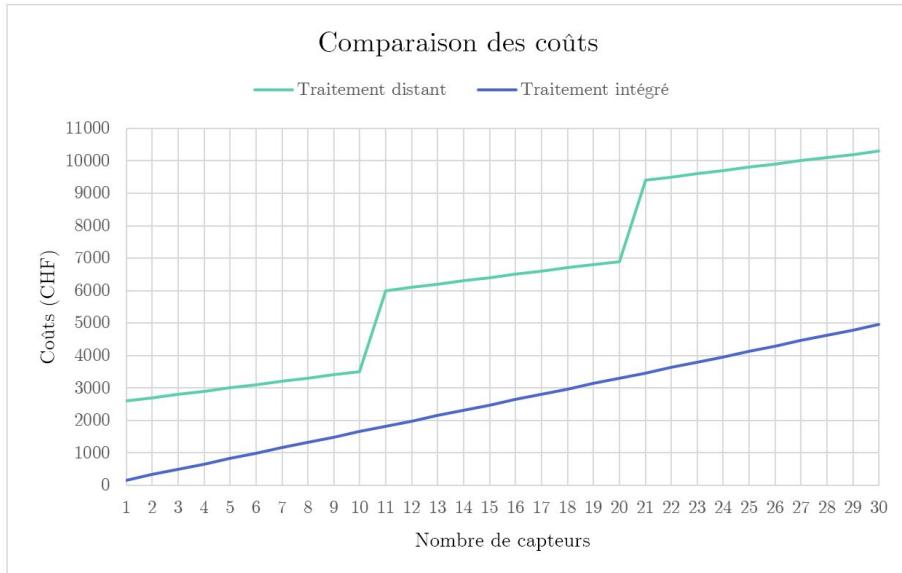


Figure 16.3 : Comparaison des coûts en fonction du nombre de capteurs

Ceci est une estimation de coûts, elle ne prend bien sûr pas en compte tous les frais comme l'installation, le cloud etc... Nous pouvons néanmoins déjà nous faire une idée claire des coûts des deux types d'architecture. Et pouvons mettre en évidence les coûts absurdes engendrés par le traitement intégré.

#### 16.4.4 Comparaison avec une offre concurrente

Pour justifier le besoin d'avoir un système peu cher, il est intéressant d'analyser les coûts d'une installation pour l'entreprise Morphean et de savoir comment ils facturent leur service. Morphean utilise des caméras AXIS avec la solution AXIS People Counter afin de proposer un produit similaire au nôtre. Les prix du hardware et des licences sont les suivants :

- **License AXIS People Counter** : 200.-/années, utilisation illimitée
- **Caméra IP AXIS Dôme** : 200.-

Durant mon entretien avec M. Sipp, CTO de Morphean, il m'a indiqué que le service est facturé au client **10.-/mois** pour une seule caméra reliée au cloud de Morphean. Cela requerrait que le client reste abonné 20 mois au service afin de générer du profit rien que pour rembourser le hardware, sans compter tous leurs serveurs et le cout de la main-d'œuvre. Il est évident qu'ils doivent avoir un autre moyen de revenue ou que les chiffres qu'ils m'ont communiqués sont que partiellement vrais.

Les coûts estimés de Morphean sont sans compter l'infrastructure Cloud qui est la vraie plus-value de Morphean) Le prix du service de 10.-/mois est cependant apparemment possible. Notre système ne peut donc absolument pas se permettre de nécessiter du hardware cher si nous souhaitons être compétitifs sur ce marché.

## 16.5 Potentiel commercial

Je n'ai pas trouvé d'études accessibles gratuitement sur le sujet, mais grâce au résumé d'une étude s'intéressant à l'évolution du marché des caméras "intelligentes" on constate que le potentiel haussier est important. Cette étude de 2019<sup>1</sup> prédit que d'ici 2024, le marché global des systèmes de surveillance intelligents en intérieur va augmenter significativement.

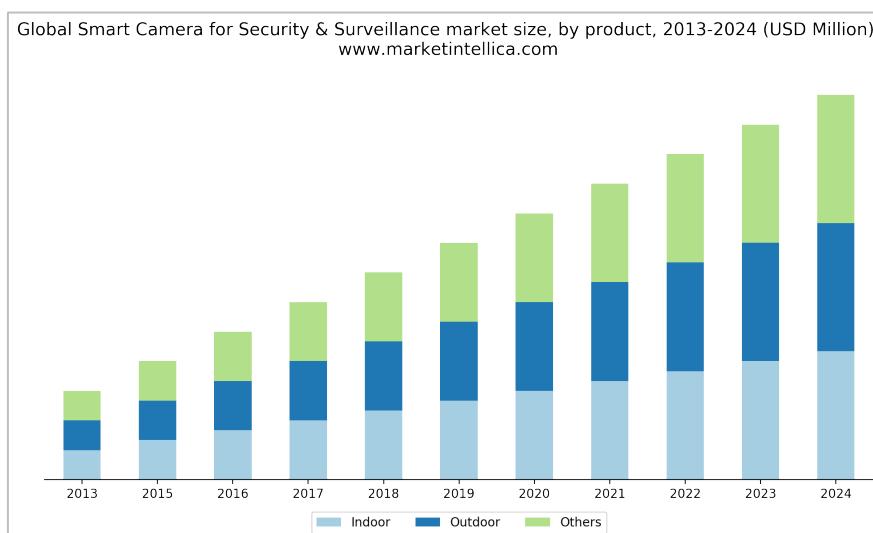


Figure 16.4 : Prédiction de la taille du marché global des caméras intelligentes

Je ne peux pas donner de chiffres exacts, l'étude nécessitant de payer pour accéder à ceux-ci. Je me base donc uniquement sur les graphiques ce qui n'est évidemment pas idéal et peut s'avérer être complètement erroné, car on ne connaît pas l'unité des axes. À prendre donc avec des pincettes. Le consensus est cependant d'accord que le virage vers des installations de vidéosurveillance intelligentes est un marché juteux dans lequel plusieurs millions de dollars vont être injectés dans les années qui suivent.

<sup>1</sup> <https://www.marketintellica.com/report/MI38837-global-smart-camera-for-security-surveillance>

## Synthèse

Cette analyse nous a permis de comprendre le cadre légal de la protection des données en Suisse et quelles directives nous devons suivre afin que le système DeepCounter soit conforme à la législation. Nous avons modélisé le but et l'offre du produit afin d'estimer son potentiel et justifier sa mise en œuvre.

L'analyse nous a également permis de pointer que le projet n'est pas spécialement innovant dans ce qu'il propose, plusieurs entreprises offrent un service similaire avec une fiabilité honorable. Nous ne disrupterons donc pas le marché avec notre produit.

## V. CONCEPTION

---

### Introduction

Nous avons pu constater dans le chapitre précédent qu'il est nécessaire de faire fonctionner plusieurs systèmes en harmonie afin d'arriver à une solution répondant au cahier des charges.

Je vais, dans ce chapitre, modéliser une architecture imaginable ainsi que les flux de données afin de m'assister lors de la réalisation de ce travail. Le but de cette partie est de comprendre avec quoi je vais travailler et comment je vais mettre le système en place afin qu'il réponde aux attentes.

## 17 Installations physiques

---

J'ai déjà expliqué de manière détaillée l'architecture et le hardware qui seront mis en place afin de capturer une vidéo et de l'envoyer sur une machine distante dans le sous-chapitre "**Architecture et hardware**" du chapitre "Solutions hardware, Traitement distant".

### 17.1 Boîtier Raspberry Pi

Le RPI est dans un boîtier qui possède une monture VESA où 1/4" qui rend sa fixation et son positionnement aisés. Le module caméra Fisheye est sur un pivot qui permet de le diriger l'endroit voulu. Le RPI a un module PoE qui fait qu'il n'a pas besoin d'être raccordé par un câble d'alimentation, seulement par un câble Ethernet supportant la technologie. Le RPI étant branché au LAN, l'accès et la configuration de celui-ci se fait par SSH donc aucunement besoin d'avoir un écran branché dessus.

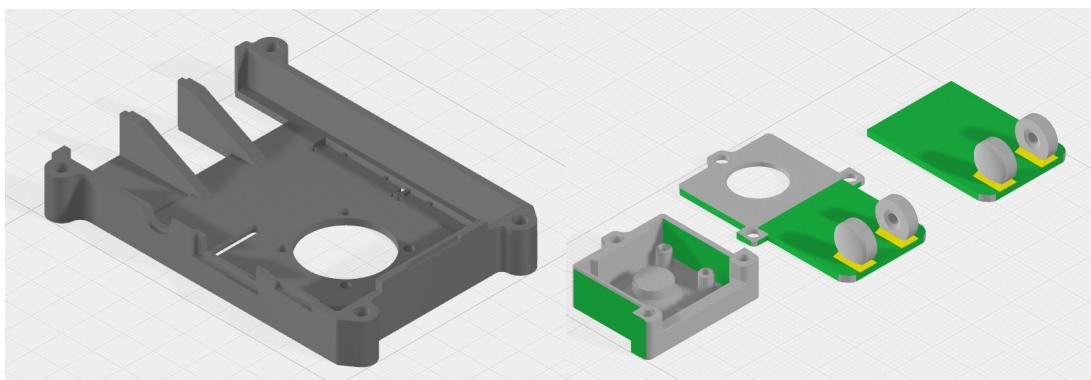


Figure 17.1 : Pièces du boîtier à imprimer en 3D



Figure 17.2 : RPI dans son boîtier<sup>1</sup> avec le module PoE et le module caméra sur pivot<sup>2</sup>

## 17.2 Positionnement du boîtier

Le boîtier et, par extension, la caméra du RPI pointe sur une entrée de bâtiment pour monitorer les personnes qui entrent et qui sortent. Elle est placée en hauteur afin d'éviter au maximum l'occlusion qui pourrait se produire en la plaçant à hauteur d'œil. On peut ainsi voir le flot de personnes de manière optimale sans rien en perdre. Le service de streaming est exécuté sur le RPI et fournit le flux vidéo de l'entrée du bâtiment sur une adresse IP.

Deux positions de caméra seront testées :

- **Deepcounterpi1** : À ~45 degrés de la porte à 3 mètres de hauteur
- **Deepcounterpi2** : En face de la porte, à 3 mètres de hauteur

Des tests seront faits pour voir quelle position est la plus avantageuse et propose la meilleure fiabilité. Si le temps me le permet, d'autres tests seront également faits pour constater les performances du système avec une caméra positionnée à hauteur d'œil. À voir également si le système de détection est capable de reconnaître des personnes depuis dessus, si ce n'est pas le cas, il faudrait songer à le ré-entrainer ou à déplacer la caméra.

---

<sup>1</sup> <https://www.thingiverse.com/thing:922740>

<sup>2</sup> <https://www.thingiverse.com/thing:262061>



Figure 17.3 : Emplacement et point de vue des caméras dans l'entrée

## 18 Diagramme flux de données

Comme expliqué en introduction de ce chapitre, la solution consistera à lier plusieurs "briques" ayant chacune une tâche définie. Chaque "brique" prendra en entrée le résultat de la brique précédente et passera à la "brique" suivante le résultat de ses calculs.

### 18.1 Détection sur chaque frame

La première approche qui est la plus intuitive est la suivante :

- Capturer le flux vidéo et en extraire chaque frame
- Y appliquer sur chacune la détection d'objets pour trouver les êtres humains
- Dessiner et enregistrer des "Bounding Boxes" autour des personnes détectées
- Appliquer l'algorithme de tracking sur les B-Boxes de la détection afin d'identifier les personnes individuellement
- Les suivre de leur entrée à leur sortie du cadre tout en maintenant leur ID (il faut gérer si possible les disparitions, les occlusions, les erreurs de détection...)
- S'il traverse une frontière, on considère qu'il est entré ou sorti du bâtiment et on met à jour un compteur
- Envoyer le compteur mis à jour sur BBData au moyen d'une requête HTTP POST

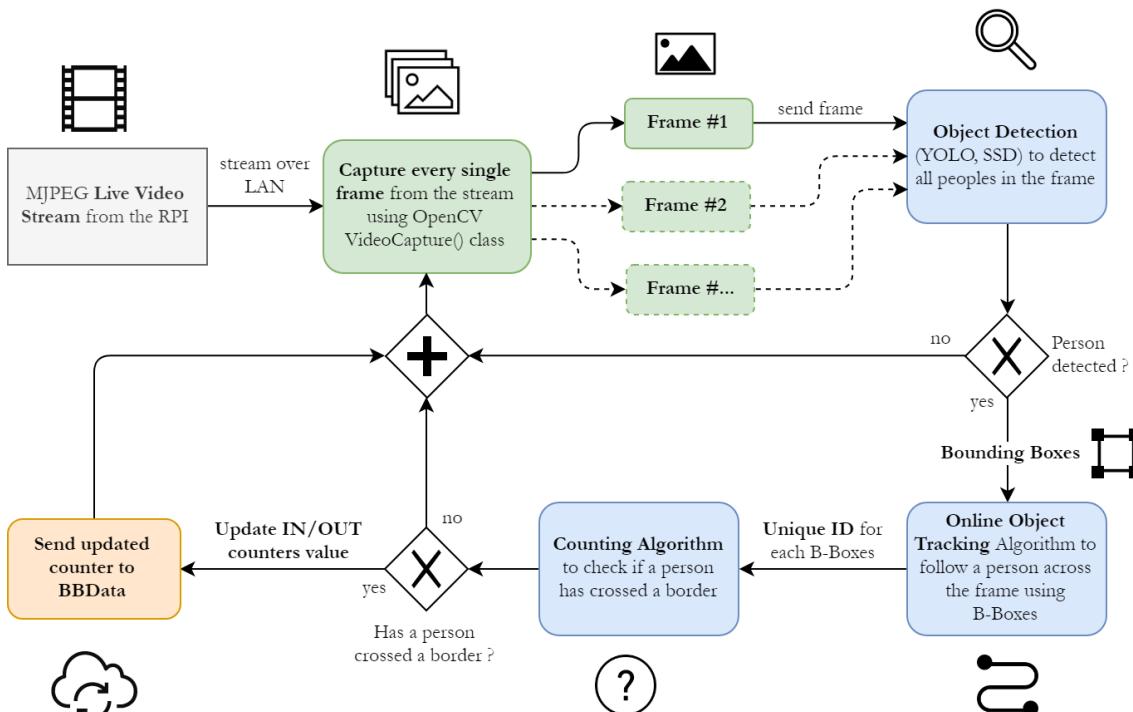


Figure 18.1 : Schéma du flux des données du système effectuant la détection et le tracking sur chaque frame

Cette solution fonctionne dans les faits, mais est couteuse en temps de calcul. En effet, le système de détection d'objets utilise un CNN et doit être appliqué sur chaque frame (jusqu'à 30 par secondes). Le temps d'inférence peut monter à plusieurs centaines de millisecondes ce qui fait qu'un système utilisant une détection d'objet lourde comme YOLOv3 par exemple, pourra être exécuté à maximum 2/3 fps sur du hardware relativement bon marché.

## 18.2 Détection chaque n frame

Le tracking est plus rapide que la détection. La solution à ce problème de performances est donc d'effectuer la détection que sur certaines de ces frames et ensuite faire confiance au système de tracking pour suivre une personne entre les détections et combler le "vide". Un simple modulo peut être utilisé ici et en fonction de sa valeur sur la frame actuelle, on passe par le "chemin" de la détection ou du tracking. On pourrait doubler les performances rien qu'en sautant une frame sur deux.

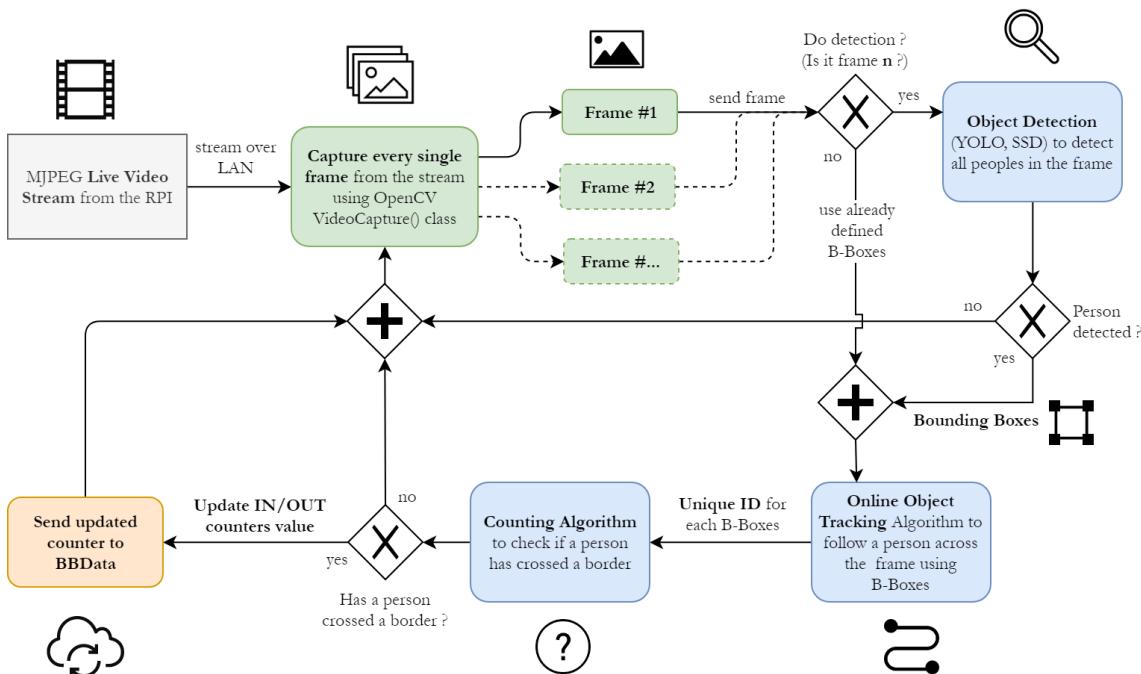


Figure 18.2 : Schéma du flux des données du système effectuant la détection et le tracking chaque n frame

Cette seconde approche nous permet d'utiliser un système de détection plus lourd afin de garantir la fiabilité de celle-ci et assister ainsi le système de tracking. À voir quelle approche permet le meilleur rapport fiabilité/vitesse. Les deux implémentations seront testées afin de définir laquelle est à préconiser pour notre projet spécifiquement.

## 19 Dataset de test

---

Nous devons construire un set de données afin de tester et d'éprouver notre système de comptage de personnes de manière reproductible.

Une capture vidéo de plusieurs heures doit être faite dans l'environnement sur lequel sera appliqué le comptage (voir "Positionnement du boîtier" ci-dessus) et avec le même hardware. On pourra ainsi compter à la main les personnes qui sortent et qui entrent pour établir le "Ground Truth" (informations fournies par observation direct). Une fois celui-ci établit, nous pourrons passer notre système de comptage sur les extraits préalablement enregistrés et jauger les performances et la fiabilité en comparant les résultats obtenus avec le GT.

Les extraits font 2 heures et 24 minutes et sont en 480p. Ils ont été capturés des deux points de vue en simultané entre 11h et 13h30. Ils font office de cas d'utilisation "typique", étant des captures effectuées dans des conditions réelles. Ils contiennent des groupes, des individus seuls, des croisements, des personnes portant des éléments volumineux, des gens qui courent...

### 19.1 Scénarios problématiques

En plus du comptage dans des conditions "normales", il faut lister plusieurs cas qui peuvent potentiellement poser un problème au système. Il faut jouer ces scénarios et les soumettre au système de comptage pour valider, ou non, son comportement. Les scénarios imaginés :

- Personne en chaise roulante
- Personne qui tourne en rond, qui sort par le côté du cadre, retourne en arrière
- Personne qui passe derrière un objet
- Personne poussant/portant un objet volumineux
- Personnes étant collées ou qui se tiennent la main
- Personne qui court (à plusieurs vitesses)
- Gros groupe (lent et rapide)
- Personne qui a le visage caché (capuchon, voile...)
- Enfant/s

Ces scénarios vont donc être enregistrées dans la mesure du possible en plus du dataset de test de base. Nous avons à présent tout ce qu'il nous faut afin d'évaluer les performances du système qui sera mis en place.

## 20 Environnement

---

### 20.1 Jupyter Notebook (6)

Un Notebook Jupyter<sup>1</sup> est un environnement de développement. C'est un "coteau suisse" du développement Python qui contient toutes les librairies couramment utilisées pour faire du machine learning (pandas, cv2, PIL, Keras, Tensorflow, numpy, matplotlib...). Il nous permet de créer des documents interactifs composés de blocs de code et de texte formaté. Il est bien sûr possible d'en exporter soit un fichier .ipynb ou un script Python.



C'est l'outil que je vais utiliser pour coder la récupération du flux vidéo avec OpenCV le système de détection d'objets, le tracking, le comptage des personnes et l'envoi des données sur BBData.

### 20.2 OpenCV<sup>2</sup>

Nous allons utiliser la librairie OpenCV. C'est une librairie spécialisée qui est faite pour le Computer Vision en temps réel, pour le Processing d'images et pour le Machine Learning. OpenCV a été conçu pour offrir des performances dignes de produits commerciaux, pour fournir une infrastructure commune aux applications de Computer Vision le tout en étant sous licence BSD.



C'est un outil puissant, complet, très bien documenté, très populaire et utilisé par les ténors de l'industrie. OpenCV est multiplateforme et supporte les frameworks de Deep Learning TensorFlow, PyTorch, Darknet et Caffee. C'est l'outil parfait pour ce travail de bachelier.

---

<sup>1</sup> <https://jupyter.org/>

<sup>2</sup> <https://opencv.org/>

OpenCV sera au centre de notre système, ce sera le ciment qui permettra au système de fonctionner en harmonie et qui permettra aux différents algorithmes de fonctionner entre eux en s'envoyant des données utiles. Ses tâches sont variées :

- Récupérer le flux vidéo du RPI sous forme utilisable grâce à la classe VideoCapture<sup>1</sup>
- Implémenter YOLO et l'utilise sur ce flux vidéo grâce au module DNN<sup>2</sup>
- Permettre de récupérer les données des bounding boxes générées grâce à YOLO
- Utiliser les bounding boxes afin de faire du tracking de personnes
- Ajouter un overlay sur la vidéo afin de visualiser les frontières
- Faire le décompte des personnes trackées lorsque celles-ci traversent les frontières

### 20.2.1 Python / C++<sup>3</sup>

OpenCV a été écrit en C++. Il existe cependant une version Python qui est essentiellement un "wrapper" autour du code original qui nous permet d'appeler des méthodes écrites en C++ depuis du code Python. Moyennant une petite baisse de performance, il est donc possible d'utiliser le langage Python et son intuitivité pour construire des projets avec OpenCV. Je vais utiliser la version Python de OpenCV, car je suis à l'aise avec ce dernier, contrairement au C++ que je n'ai jamais utilisé. Le temps imparti ne me permet pas de me former à l'emploi d'un nouveau langage.

---

## 21 Object Detection

---

### 21.1 OpenCV DNN Module<sup>4</sup>

C'est une librairie qui permet l'utilisation (inférence) de modèles de Deep Learning pré-entraînés. Il ne permet pas l'entraînement, mais uniquement la prédiction. Il supporte les frameworks de Deep Learning suivants, sont listées seulement ceux qui font la détection d'objets (avec les modèles de détection d'objets pré-entraînés supportés par OpenCV) :

- **Darknet** : YOLOv2, YOLOv3, YOLOv3-tiny
- **Caffe** : SSD-VGG, SSD-MobileNet, Faster-RCNN
- **Tensorflow** : SSD-Inception, SSD-MobileNet, Faster-RCNN

---

<sup>1</sup> [https://docs.opencv.org/3.1.0/d8/dfe/classcv\\_1\\_1VideoCapture.html](https://docs.opencv.org/3.1.0/d8/dfe/classcv_1_1VideoCapture.html)

<sup>2</sup> [https://docs.opencv.org/master/d2/d58/tutorial\\_table\\_of\\_content\\_dnn.html](https://docs.opencv.org/master/d2/d58/tutorial_table_of_content_dnn.html)

<sup>3</sup> [https://docs.opencv.org/3.4/da/d49/tutorial\\_py\\_bindings\\_basics.html](https://docs.opencv.org/3.4/da/d49/tutorial_py_bindings_basics.html)

<sup>4</sup> <https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV>

Model	Version		
MobileNet-SSD v1	2017_11_17	<a href="#">weights</a>	<a href="#">config</a>
MobileNet-SSD v1 PPN	2018_07_03	<a href="#">weights</a>	<a href="#">config</a>
MobileNet-SSD v2	2018_03_29	<a href="#">weights</a>	<a href="#">config</a>
Inception-SSD v2	2017_11_17	<a href="#">weights</a>	<a href="#">config</a>
Faster-RCNN Inception v2	2018_01_28	<a href="#">weights</a>	<a href="#">config</a>
Faster-RCNN ResNet-50	2018_01_28	<a href="#">weights</a>	<a href="#">config</a>

Figure 21.1 : Exemple de modèles mis à disposition pour Tensorflow sur OpenCV<sup>1</sup>

J'ai choisi d'utiliser le module DNN de OpenCV pour faire la détection d'objet par soucis de simplicité de mise en place. Les modèles étant déjà existants, il suffit de télécharger leurs poids et leur topologie afin de les utiliser directement dans OpenCV. On peut ensuite aisément récupérer les données qu'ils produisent afin de les utiliser plus loin dans la chaîne.

On sacrifie peut-être des performances en passant à travers OpenCV pour faire l'inférence, mais une fois la logique construite, il ne sera pas difficile d'utiliser un implémentation Python de YOLO par exemple, qui offrira une vitesse meilleure avec un GPU. Le but de ce projet étant de fournir un "Proof-of-concept", cette approche est cohérente.

Également, dans le cas où YOLO ne nous fournit pas de performances suffisantes, nous pourrons ainsi assez aisément implémenter d'autres systèmes de détection d'objet.

## 22 Tracking

Pour le tracking, il faudra commencer par implémenter une solution basique comme la distance Euclidienne et aviser ensuite s'il est nécessaire ou non de fournir un système plus évolué. Le tracking s'occupera de créer un **objet** unique pour chaque Bounding Box, de lui attribuer un ID et permettra de le suivre le long de son chemin dans le cadre en enregistrant l'historique de sa position. Il est capable de créer et supprimer des objets lorsqu'ils apparaissent et disparaissent du cadre.

---

<sup>1</sup> <https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API>

## 22.1 OpenCV MultiTracker<sup>1</sup>

Huit algorithmes de tracking sont proposés par OpenCV : BOOSTING, MIL, KCF, TLD, MEDIANFLOW, GOTURN, MOSSE et CSRT. Je ne vais pas les présenter en détail, la ressource suivante le fait : *Object Tracking using OpenCV*<sup>2</sup>

## 22.2 Dlib Correlation Filters<sup>3</sup>

Dlib est une librairie offrant des algorithmes de machine learning. Elle met à notre disposition un algorithme de tracking qui basé sur la corrélation. Il est facile à implémenter et à utiliser : il faut lui fournir les coordonnées de la bounding box de l'objet que nous souhaitons suivre et il va définir sa position dans les frames suivantes. Il se met à jour de lui-même et ne nécessite pas l'utilisation du système de détection d'objets sur chaque frame.

## 22.3 Centroid Tracking

Voir chapitre : "Euclidean Distance Algorithm". Nécessite de faire la détection sur chaque frame. Prends le centre d'une Bounding Box (le centroid) et utilise la distance d'Euclide pour le matcher ou non avec un ID d'objet d'une frame précédente. Une implémentation en a été faite ici : *Simple object tracking with OpenCV*<sup>4</sup>

---

# 23 Algorithme de comptage

---

Pour le comptage des personnes, il y a aussi plusieurs approches qui sont possibles. Nous devons utiliser les données fournies par l'algorithme de tracking. Nous avons donc à disposition les objets créés par celui-ci qui contiennent les informations suivantes :

- ID unique
- Historiques sous forme de tableau des coordonnées du centroid

Il nous faut développer un système de comptage au moyen de ces informations uniquement.

---

<sup>1</sup> [https://docs.opencv.org/3.4/d9/df8/group\\_\\_tracking.html](https://docs.opencv.org/3.4/d9/df8/group__tracking.html)

<sup>2</sup> <https://www.learnopencv.com/object-tracking-using-opencv-cpp-python/>

<sup>3</sup> <http://blog.dlib.net/2015/02/dlib-1813-released.html>

<sup>4</sup> <https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/>

### 23.1 Approche basée sur la direction de déplacement

Nous pouvons avoir un système qui utilise l'historique de l'emplacement du centroid d'un objet et calculer le sens du mouvement de la personne au moyen de celles-ci.

- On fait la moyenne de l'historique des coordonnées (X ou Y) du centroid
- On soustrait à la coordonnée actuelle (X ou Y) du centroid la moyenne ci-dessus.
- Si le résultat est positif, on va dans un sens s'il est négatif, on va dans l'autre
- On divise le cadre en deux zones : une du côté IN, une du côté OUT
- On regarde ensuite la zone dans laquelle il se trouve au moyen des coordonnées du centroid actuel. S'il va dans le sens IN et qu'il se trouve dans la zone côté intérieur, on incrémente le conteur des personnes entrées et inversement.

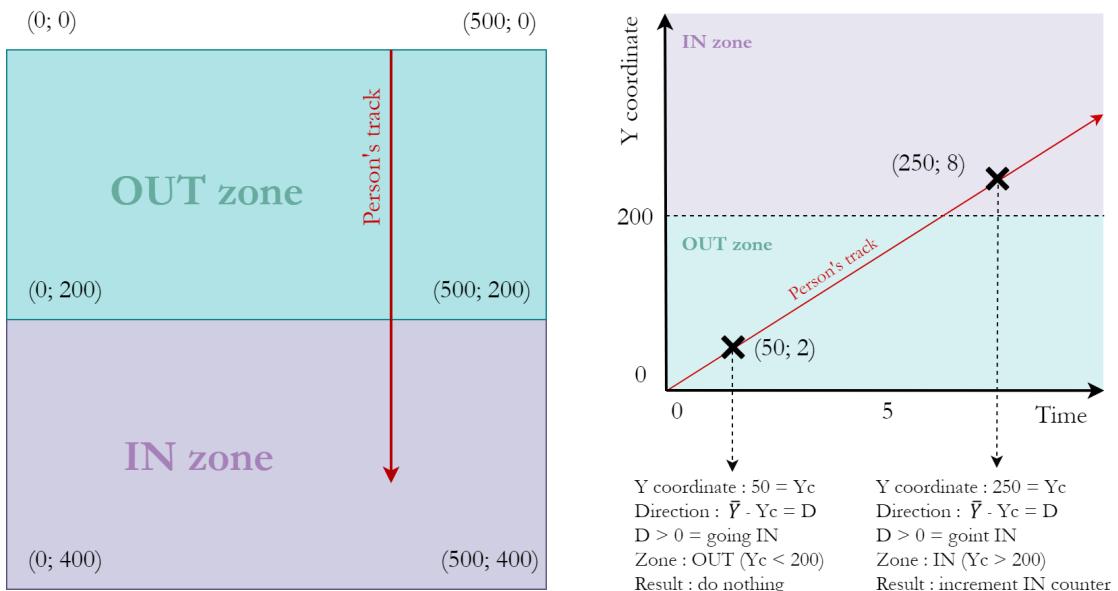


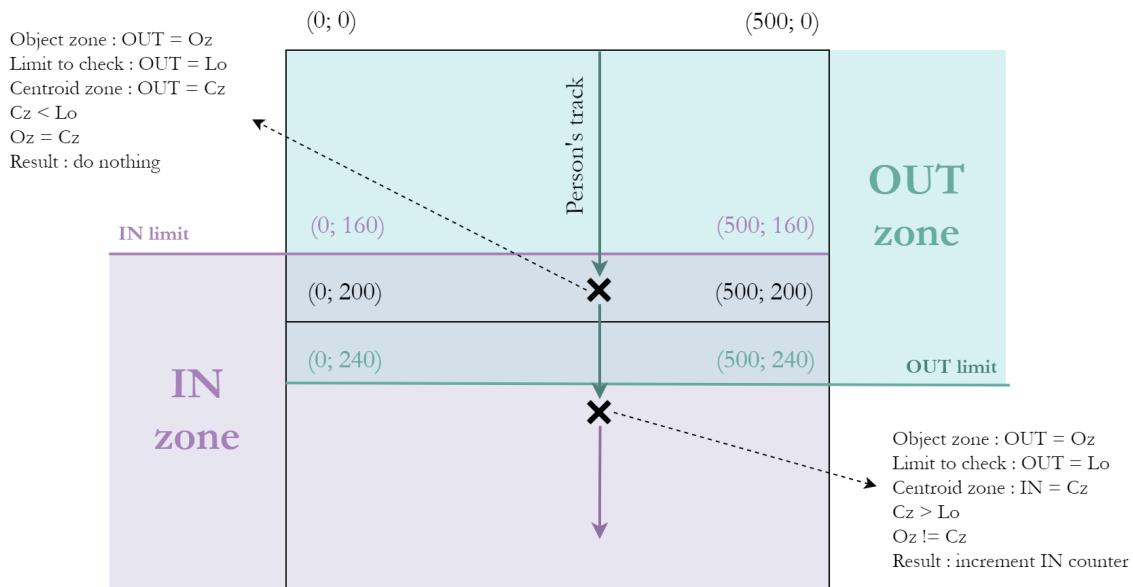
Figure 23.1 : Schéma du fonctionnement de l'algorithme de comptage

### 23.2 Approche basée sur des zones

Ici, plus besoin de tenir un historique des positions du centroid. On ajoute cependant à l'objet un attribut "Zone" qui contient la zone dans laquelle la personne se trouve actuellement.

- Une fois la personne entrée dans le cadre, on assigne une valeur à son attribut "Zone" en fonction des coordonnées de son centroid (IN ou OUT)
- On met à jour la zone dans laquelle la personne se trouve à chacun de ses mouvements

- On place deux limites, une du côté IN et une du côté OUT. Les limites ont un espace entre-elles pour éviter les erreurs si une personne reste sur celles-ci.
  - Si l'objet personne est en zone IN et le centroid se trouve dans la zone, on change sa zone d'appartenance et on incrémente le conteur correspondant



*Figure 23.2 : Schéma du fonctionnement de l'algorithme de comptage*

### 23.3 Choix de l'algorithme de comptage

À ce stade, rien ne nous permet de choisir un algorithme de comptage. Il faudra les implémenter et les tester indépendamment afin de voir lequel offre la plus grande fiabilité dans une utilisation commune et dans des cas spéciaux.

Synthèse

Cette phase de conception nous a permis de modéliser en entier notre solution. De définir les outils qui seront utilisées et la logique qui sera implémentée. Nous avons donc à présent toutes les clés qui nous permettront de mettre en place un prototype de solution que nous pourrons tester, éprouver et améliorer.

Rien n'est cependant gravé dans le marbre, en fonction des résultats et des performances des différentes implémentations, il faudra peut-être revenir sur ces réflexions afin de fournir une solution qui soit en accord avec les attentes du projet.

## VI. RÉALISATION

---

### Introduction

Je vais expliquer dans ce chapitre le développement des différents "modules" ou les différentes "briques" qui composent la solution. Chacune a une utilité bien précise et elles ont été modélisées dans le chapitre "Conception".

Les ressources suivantes m'ont aidé afin de faire la réalisation :

- Le site : <https://www.pyimagesearch.com/>
- Le site : <https://www.learnopencv.com/>
- Le livre : *Mastering OpenCV 4 with Python* - Alberto Fernandez Villan (23)
- Le livre : *Building Computer Vision Projects with OpenCV 4 and C++* (24)

### Etapes :

1. Mettre en place et configurer le RPI
2. Lire et capturer le flux vidéo
3. Object Detection
4. Object Tracking
5. Algorithme de comptage
6. Envoie des données sur BBData

## 24 Raspberry Pi

---

### 24.1 Hardware

- Raspberry Pi 3B+
- Carte Micro SD 16 GB
- Module Caméra : Fisheye NightVision 5 MP
- Module PoE
- Boîtier imprimé en 3D

### 24.2 Installations préalables

#### Operating System

On installe Raspbian sur la carte SD : <https://www.raspberrypi.org/downloads/noobs/>

## Module PoE<sup>1</sup>

Le module PoE nous permet d'avoir que un câble Ethernet branché au RPI, le courant passant à travers celui-ci. On économise donc un câble et la mise en place sera plus facile.

## Raspi-Config<sup>2</sup>

Il faut autoriser le SSH en configurant le RPI pour y accéder à distance via son adresse IP. Il faut aussi autoriser le module caméra afin que nous puissions accéder à son flux.

## Adresse IP statique<sup>3</sup>

On configure l'adresse IP du RPI pour qu'elle soit statique. Ainsi, on pourra facilement le trouver sur notre réseau local, car il aura toujours la même adresse IP.

### 24.2.1 UV4L (stream)

Suivre le tutoriel suivant pour installer UV4L sur le RPI et permettre son lancement automatique en tant que service dès la mise sous tension de celui-ci : <https://www.linux-project.org/uv4l/installation/>

#### Stream

Le stream se trouve à : *adresseIPduRPI:8080/stream*

#### Configuration

On peut configurer le stream (FPS, résolution, codec) ici : *adresseIPduRPI:8080/panel* ou en modifiant le fichier `/etc/uv4l/uv4l-raspicam.conf`

Une fois tout ceci fait, il nous suffit de mettre le boîtier en position, d'orienter la caméra au bon endroit et de brancher un câble Ethernet (lui-même branché sur un réseau compatible PoE) et le tour est joué ! Le RPI stream sur l'adresse IP.

---

<sup>1</sup> <https://www.raspberrypi.org/products/poe-hat/>

<sup>2</sup> <https://www.raspberrypi.org/documentation/configuration/raspi-config.md>

<sup>3</sup> <https://thepihut.com/blogs/raspberry-pi-tutorials/how-to-give-your-raspberry-pi-a-static-ip-address-update>

## 25 Récupérer le flux vidéo

---

Avant de pouvoir faire du traitement sur une image, il faut la récupérer depuis un flux d'une manière ou une autre. Dans notre cas, le flux est envoyé sur un adresse IP. OpenCV met à disposition une classe pour le faire : VideoCapture.

### 25.1 OpenCV VideoCapture Class<sup>1</sup>

VideoCapture est une classe qui permet de capturer la vidéo à partir de fichiers vidéo, de séquences d'images, de webcam ou d'adresse IP (flux live). C'est parfait dans notre cas d'application. On a seulement à lui indiquer la source du flux vidéo en paramètres (dans notre cas c'est l'adresse IP) et il se charge de créer un objet correspondant depuis lequel on pourra lire chaque frame

Nous créons également des variables enregistrant la hauteur et la largeur de l'image du stream. Cela nous sera utile plus tard.

```
streamIP = "http://160.98.31.185:8080/stream/video.mjpeg"

vs = cv2.VideoCapture(streamIP)

H = int(vs.get(cv2.CAP_PROP_FRAME_HEIGHT))
W = int(vs.get(cv2.CAP_PROP_FRAME_WIDTH))
```

Figure 25.1 : Création d'un objet VideoCapture

### 25.2 Lire le flux vidéo<sup>2</sup>

Pour lire le flux vidéo, on crée une boucle toujours "True" qui va lire chacune des frames qui sont retournées par l'objet VideoCapture jusqu'à ce qu'on lui dise explicitement d'arrêter. Cela nous permet de récupérer chaque image de la vidéo de manière indépendant et donc d'y faire sur chacune un traitement indépendant.

---

<sup>1</sup> [https://docs.opencv.org/4.1.0/d8/dfe/classcv\\_1\\_1VideoCapture.html](https://docs.opencv.org/4.1.0/d8/dfe/classcv_1_1VideoCapture.html)

<sup>2</sup> [https://docs.opencv.org/2.4/modules/highgui/doc/reading\\_and\\_writing\\_images\\_and\\_video.html#videocapture-read](https://docs.opencv.org/2.4/modules/highgui/doc/reading_and_writing_images_and_video.html#videocapture-read)

```
# loop over frames from the video stream
while True:
    # read the next frame
    (grabbed, frame) = vs.read()

    # if the frame was not grabbed, end of the stream
    if not grabbed:
        print("Done processing !!!")
        break

    #####
    # Do processing here
    #####

    # show the video (useless in our usecase)
    cv2.imshow('RPI', frame)

    # quit when key is pressed
    if cv2.waitKey(1) == ord('q'):
        break

    # release the stream pointers
    vs.release()
```

Figure 25.2 : Boucle sur les images du flux vidéo

## 26 Object Detection

---

Maintenant que nous avons des images provenant du RPI, nous pouvons y appliquer de la détection d'objets pour détecter les humains présents dans celles-ci.

### Etapes :

1. Définir le modèle de détection à utiliser
2. Faire le pré-processing, créer un "blob"
3. Donner en input l'image au modèle pré-entraîné pour obtenir la prédiction
4. Filtrer les prédictions en fonction de leurs classes et de leur fiabilité
5. Les afficher sur l'image

## 26.1 OpenCV DNN Module<sup>1</sup>

Pour ce faire, nous allons utiliser le module DNN de OpenCV (25). Ce module nous permet de faire de l'inférence au sein de OpenCV au moyen de réseaux profonds (Deep Networks) pré-entraînés. Il supporte plusieurs framework populaires, nous allons utiliser Tensorflow (SSD) et Darknet (YOLO), car ils contiennent tout ce dont nous avons besoin. Il est important de noter que le module DNN est à utiliser uniquement pour faire une détection, on ne peut pas l'utiliser afin de réentraîner un modèle.

Les poids et les fichiers de configurations/topologie peuvent être téléchargé depuis :

- Tensorflow : <https://github.com/opencv/opencv/wiki/TensorFlow-Object-Detection-API>
- Darknet : <https://pjreddie.com/darknet/yolo/>

Nous devons faire deux codes, car on n'appelle pas les deux framework de la même manière et ils ne retournent pas les résultats de l'inférence sous la même forme :

	Tensorflow	Darknet
<b>Object Detection Model</b>	SSD	YOLOv3, YOLOv2
<b>Pretrained Dataset</b>	COCO	COCO
<b>CNN Architecture</b>	MobileNet, Inception	Darknet-19, Darknet-53
<b>Read Network call</b>	readNetFromTensorflow	readNetFromDarknet
<b>Config File</b>	.pb	.cfg
<b>Weight File</b>	.pbtxt	.weights
<b>Forward Function Return</b>	(1, 1, 100, 7)	(845, 85)
<b>Representation of BBox</b>	(left, top, right, bottom)	(left, top, width, height)

---

<sup>1</sup> <https://github.com/opencv/opencv/wiki/Deep-Learning-in-OpenCV>

## 26.2 Chargement du modèle

Pour ce faire, on utilise la fonction : OpenCV readNetFrom...<sup>1</sup>

### 26.2.1 Darknet (YOLO)

```
# Width and Height of network's input image
inputWidth = 416
inputHeight = 416

cfg_file = "yolo/yolov3.cfg"
weights_file = "yolo/yolov3.weights"

# Give the configuration and weight files for the model and load the
# network using them.
net = cv2.dnn.readNetFromDarknet(cfg_file, weights_file)
```

Figure 26.1 : Définition du modèle à utiliser et charge les poids et la configuration

### 26.2.2 Tensorflow

```
# Width and Height of network's input image
inputWidth = 300
inputHeight = 300

pbFile = "tf/ssd_inception_frozen_inference_graph.pb"
pbtxtFile = "tf/ssd_inception_v2_coco_2017_11_17.pbtxt"

# Give the configuration and weight files for the model and load the
# network using them
net = cv2.dnn.readNetFromTensorflow(pbFile, pbtxtFile)
```

Figure 26.2 : Définition du modèle à utiliser et charge les poids et la configuration

## 26.3 Faire la prédiction (inférence)

Une fois le modèle chargé, nous pouvons l'utiliser pour faire l'inférence (la prédiction). On construit un "blob" depuis l'image et on la passe dans le réseau en entrée. On obtient en sortie une liste de prédictions, leur fiabilité et leurs coordonnées dans la frame.

---

<sup>1</sup> [https://docs.opencv.org/4.1.0/d6/d0f/group\\_\\_dnn.html](https://docs.opencv.org/4.1.0/d6/d0f/group__dnn.html)

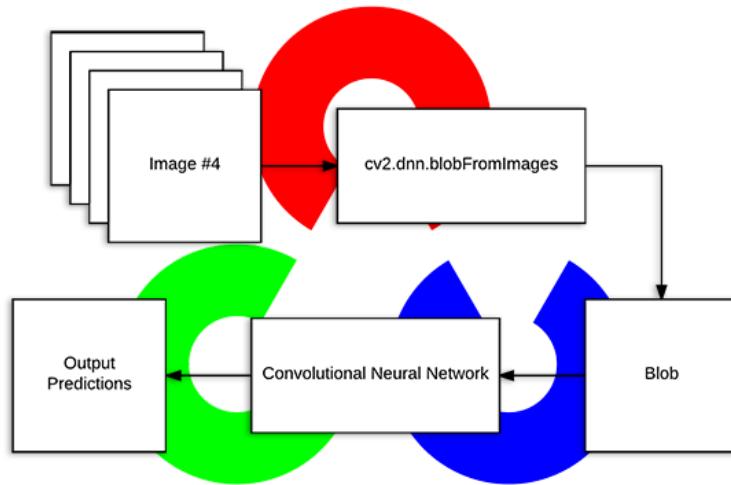


Figure 26.3 : Schéma montrant les étapes pour faire une prédiction dans OpenCV<sup>1</sup>

Le "blob" est une étape de pré-processing, il faut donner au réseau une image dans le bon format afin qu'il l'accepte. La fonction retourne une matrice en 4 dimensions. Plusieurs choses peuvent/doivent être faites en fonction du modèle :

- Redimensionnement de l'image en fonction du modèle
- Mean Subtraction
- Scaling
- Inverser les canaux rouges et bleus
- Rognage

Plus d'informations ici : <https://www.pyimagesearch.com/2017/11/06/deep-learning-opencv-blobfromimage-works/>

```
# Create the blob with a size of (300, 300)
blob = cv2.dnn.blobFromImage(frame, size=(inputWidth, inputHeight),
    swapRB=True, crop=False)

# Feed the input blob to the network, perform inference and get the output:
# Set the input for the network
net.setInput(blob)

detections = net.forward()

# call function to process detections
detect(frame, detections)
```

Figure 26.4 : Création du blob et envoi dans le réseau afin d'avoir une prédiction

<sup>1</sup> <https://www.pyimagesearch.com/2017/11/06/deep-learning-opencv-blobfromimage-works/>

## 26.4 Traitement des prédictions

Une fois qu'une liste de prédiction a été obtenue, nous pouvons les filtrer afin de garder uniquement celles qui ont un niveau de fiabilité suffisant et nous pouvons aussi filtrer tout ce que nous ne souhaitons pas (garder que les humains dans notre cas). On finit par dessiner un cadre autour de la prédiction (Bounding Box).

### 26.4.1 SSD

```
# minimum probability to filter weak detections
minConfidence = 0.3

# process detections
def detect(frame, detections):
    # loop over the detections
    for detection in detections[0, 0, :, :]:
        confidence = float(detection[2])

        # if the confidence is above a threshold
        if (confidence > minConfidence):
            classID = detection[1]

            # proceed only if the object detected is indeed a human
            if(classID == 1):
                # get coordinates of the bbox
                left = detection[3] * W
                top = detection[4] * H
                right = detection[5] * W
                bottom = detection[6] * H

                # create BBox
                box = [left, top, right, bottom]
                # Draw BBox around detections
                drawBoundingBox(frame, box, color=(0, 0, 255))
```

Figure 26.5 : Fonction permettant de filtrer les prédictions et de les afficher

### 26.4.2 YOLO

Le code commenté en détail est sur Gitlab<sup>1</sup>, je ne le mettrai pas ici par soucis de place. Le traitement et le filtrage des prédictions est fait différemment, car le réseau ne donne pas la même sortie. Il est également plus long que SSD, car certaines étapes spécifiques doivent être faites en plus. La logique est cependant identique dans les grandes lignes.

---

<sup>1</sup> <https://gitlab.forge.hefr.ch/patrick.audriaz/tb-audriaz/tree/master/Realisation/OpenCV%20YOLO>

### 26.4.3 Résultat des détections

Une fois la détection faite, nous obtenons une image avec les humains qui ont été détectées mises en évidence. Ci-dessous, les résultats de différents modèles sur une image de test.

#### YOLOv3

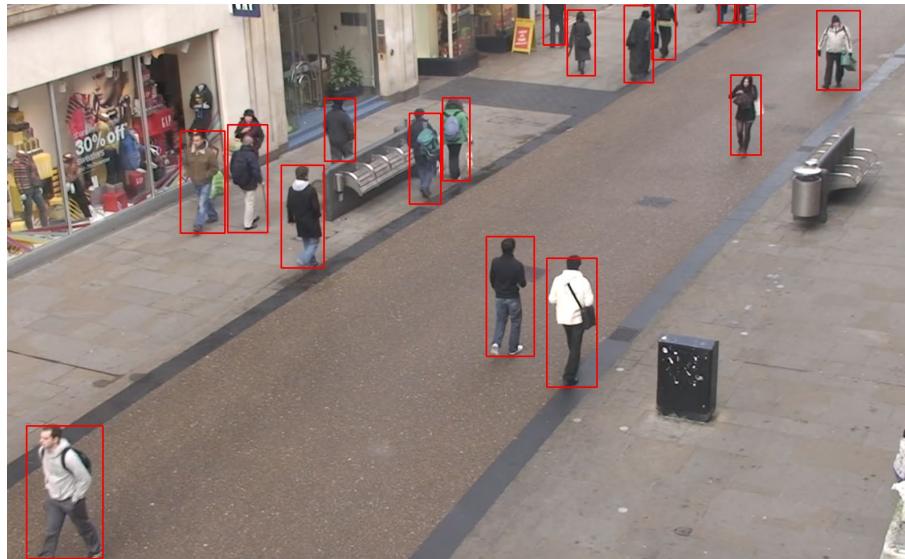


Figure 26.6 : Object Detection affichant que les humains avec YOLOv3

#### SSD Inception V2



Figure 26.7 : Object Detection affichant que les humains avec SSD Inception V2

On constate des différences de fiabilité entre les deux modèles. Nous allons quantifier la fiabilité que chacun dans la partie "Tests" de ce rapport.

## 27 Object tracking

Maintenant la détection faite, nous pouvons utiliser les données retournées par celle-ci afin d'attribuer un ID unique à chaque BBox. Cela nous permettra de les suivre le long de leur parcours et donc de définir s'ils entrent ou sortent de la zone.

Le tracking utilise et combine deux méthodes afin d'améliorer la robustesse du système :

- Centroid Tracker (basé sur la distance d'Euclide)<sup>1</sup>
- Correlation Filter (de la librairie Dlib)<sup>2</sup>

Les ressources suivantes ont été utilisées pour l'implémentation :

- Simple Object Tracking : <https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/>
- Object tracking with Dlib : <https://www.pyimagesearch.com/2018/10/22/object-tracking-with-dlib/>
- Multi-object tracking with Dlib : <https://www.pyimagesearch.com/2018/10/29/multi-object-tracking-with-dlib/>
- OpenCV People Counter : <https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/>

### 27.1 Objet "suivable"

Pour chaque BBox détectée, un objet est créé. Il contient un ID, un historique (une liste) de la position de son centroid et la zone dans laquelle il se trouve.

```
# for every bounding box detected, a trackable object is created
# to follow his path through the frame
class TrackableObject:
    def __init__(self, objectID, centroid, zone):
        # store the object ID, then initialize a list of centroids
        # using the current centroid and store current zone
        self.objectID = objectID
        self.centroids = [centroid]
        self.zone = zone
```

Figure 27.1 : Objet permettant de suivre une personne

<sup>1</sup> <https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/>

<sup>2</sup> <http://blog.dlib.net/2015/02/dlib-1813-released.html>

## 27.2 Tracking

A chaque fois qu'une détection est faite, on envoie les BBox au tracking afin de créer des objets et les suivre.

Tous les objets sont stockés dans une liste et on les met à jour leurs attributs à chaque frame. Chaque objet a un ID unique ce qui permet de les différencier. On leur en attribut un à leur entrée dans le cadre (dès qu'ils sont détectés pour la première fois) et on les supprime une fois qu'ils disparaissent de celui-ci.

```
trackers = []

# construct a dlib rectangle object from the bounding
# box coordinates and then start the dlib correlation tracker
tracker = dlib.correlation_tracker()
rect = dlib.rectangle(left, top, right, bottom)

tracker.start_track(frame, rect)

trackers.append(tracker)

rects.append(box)
```

*Figure 27.2 : On envoie les données de la détection au tracking Dlib*

On appelle le CentroidTracker une fois la détection faite : `objects = ct.update(rects)`

Pour le Centroid Tracker, voir ici : <https://github.com/balbatross/centroid-tracker>

### 27.2.1 Résultats du tracking

Une fois le tracking fait, nous obtenons une image avec les humains qui ont été détectées mises en évidence et un ID unique leur est associé.

#### YOLOv3



Figure 27.3 : Tracking en utilisant la détection de YOLOv3

#### SSD Inception V2



Figure 27.4 : Tracking en utilisant la détection de SSD Inception V2

Sur SSD Inception, on peut constater que la personne de droite à l'ID 3 au lieu de 1, cela signifie qu'on a perdu son tracking et qu'on lui a réattribué un nouvel ID.

## 28 Algorithme de comptage

Nous avons la détection d'êtres humains et nous sommes capable de les suivre individuellement tout le long de leur chemin à travers le cadre de la vidéo.

Nous pouvons utiliser ces données afin de définir si une personne entre ou sort du bâtiment. La logique de comptage a été expliquée dans la partie "Conception".

```
global totalIn
global totalOut

# loop over the tracked objects
for (objectID, centroid) in objects.items():
    # check to see if a trackable object exists for the current object
    to = trackableObjects.get(objectID, None)

    # if there is no existing trackable object, create one
    if to is None:
        # define starting zone of the trackable object (IN or OUT)
        if centroid[1] ≥ H/2:
            zone = "in"
        else :
            zone = "out"

        to = TrackableObject(objectID, centroid, zone)

    # otherwise an object exist so we can utilize it to know direction
    else:
        if to.zone == "in" :
            if centroid[1] < limitOut:
                totalOut += 1
                to.zone = "out"

        elif to.zone == "out" :
            if centroid[1] > limitIn:
                totalIn += 1
                to.zone = "in"

    # add current centroid to object centroids list
    to.centroids.append(centroid)

# store the trackable object in our dictionary
trackableObjects[objectID] = to
```

Figure 28.1 : Algorithme de comptage



Figure 28.2 : Deux personnes qui sortent du bâtiment, on peut voir le compteur "OUT" en bas à gauche qui a été incrémenté au moment où ils ont traversé la limite

## 29 Détection chaque n frame

Comme indiqué dans le chapitre : "Détection chaque n frame", la détection est très couteuse en ressources. On la fait donc seulement toutes les n frames et on utilise entre deux le tracking pour assurer le suivi. Pour ce faire, on utilise simplement un modulo sur le numéro de la frame.

```
# object detection only every n frames to improve performances
if elapsedFrames % skipFrames == 0:
    # .....

    # do detection
    detect(frame, detections)

# do tracking using detection data on every other frame
else:
    track(frame, trackers)

# increment the total number of frames processed thus far
elapsedFrames += 1
```

Figure 29.1 : Détection chaque n frame

## 30 Envoie sur BBData

L'envoi des données sur BBData se fait simplement avec un appel à l'API Rest. Il faut lui envoyer une requête HTTP Post contenant les données utiles. Nous le faisons au moyen de la librairie Requests<sup>1</sup> pour Python.

Il faut au préalable créer un objet sur BBData auquel nous pourrons envoyer les données. Nous créons deux objets par caméra. Un objet "IN" qui est un compteur des gens qui sont rentrés et un compteur "OUT" qui est pour les gens qui sont sortis.

Nous pouvons leur envoyer des données en spécifiant l'ID et le Token de l'objet souhaité. L'appel se fait à chaque évènement "IN +1" ou "OUT +1".

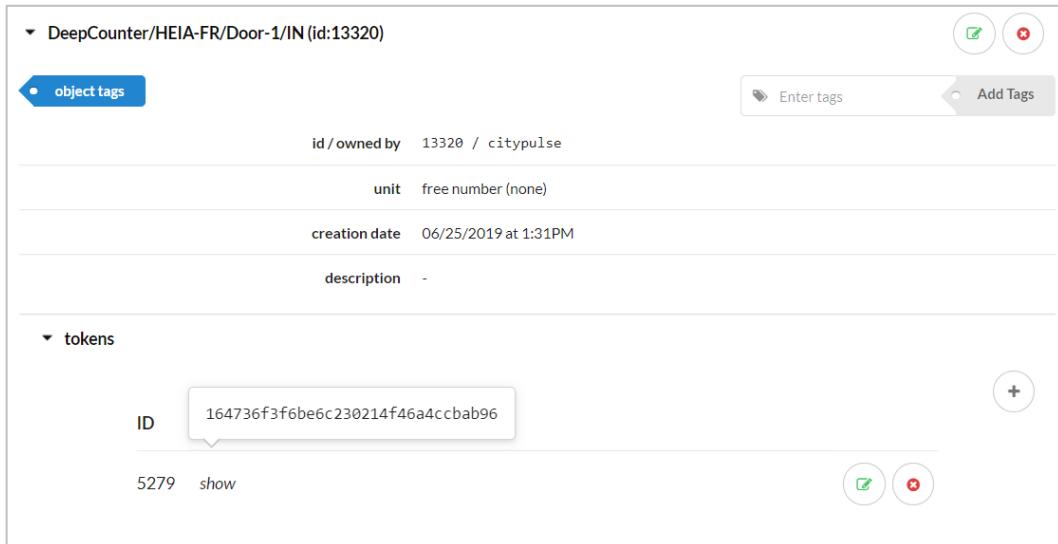


Figure 30.1 : Objet BBData pour un capteur "IN", son unité est un entier

<sup>1</sup> <https://2.python-requests.org/en/master/>

```
# constants used to push new measure in BBData
url = 'https://bbdata.daplab.ch/input/measures'
headers = {'content-type': 'application/json', 'accept': 'application/json'}
```

```
push_to_bbdata(13320, "164736f3f6be6c230214f46a4ccbab96", totalIn)
```

```
# send data to BBData
def push_to_bbdata(object_id, object_token, value):
    # get current timestamp and format to ISO
    now = datetime.datetime.utcnow().replace(microsecond=0).isoformat()
```

```
    # form the payload with parameters
    payload = {"objectId": object_id, "token": object_token, "timestamp": now,
               "value": value}
```

```
    response = requests.post(url, data=json.dumps(payload), headers=headers)
```

```
    # 0 if success and -1 if there is an error
    if response.status_code == 200:
        print("Success")
        print(response.json())
        return 0
    else:
        print("Error")
        return -1
```

Figure 30.2 : Envoie des données sur BBData pour un compteur "IN"

---

## Synthèse

Nous avons pu voir dans ce chapitre un bref tour d'horizon des différents "modules" qui ont été développé afin de parvenir à effectuer le comptage. Ceci en accord avec ce qui a été modélisé dans la Conception. Le code commenté en détail se trouve sur Gitlab. La solution étant maintenant fonctionnelle, nous pouvons la tester en profondeur afin d'en évaluer la fiabilité.

## VII. TESTS ET VALIDATION

---

### Introduction

Afin de valider que la solution développée est cohérente et utilisable dans le monde réel, il faut évaluer sa fiabilité de manière scientifique. Savoir les performances offertes dans les différents cas d'utilisation afin de connaître les avantages et les limitations de la solution,

Un datasets ont été construits comme indiqué dans le chapitre "Dataset de test", il contient plusieurs extraits vidéo :

- Un extrait long de 2h filmé depuis les deux points de vue en simultané
- Des scénarios courts spécifiques enregistrés de manière indépendante

Les tests vont être effectuées sur chacun de ses extraits avec chacune des solutions développées et ses différentes variantes (Skip Detection Rate, FPS, ...). Les modèles testés ont été choisis grâce à l'analyse technologique et sont les suivants :

- Darknet YOLOv3
- Darknet YOLOv3 Tiny
- Darknet YOLOv2
- Tensorflow SSD MobileNet V2
- Tensorflow SSD Inception V2

Les tests sont effectués en "off-line" sur des fichiers pré-enregistrés et non en direct afin de pourvoir avoir des tests reproductibles et comparables. Tous les résultats suivants représentent donc les performances dans un cas de figure où il n'y a aucun "bottleneck", où le hardware a une puissance illimitée et que le flux vidéo est envoyé à la vitesse maximum. Dans la réalité, il faudra choisir une solution qui correspond au hardware donné en fonction de nos besoins en fiabilité et en vitesse.

### 31 Vitesse

---

L'évaluation de la vitesse d'un système nous sert à définir sur quel hardware il peut être utilisée. Si on peut envisager du Edge Computing ou si une workstation est nécessaire.

### 31.1 OpenCV vs Darknet sur CPU

Nous utilisons OpenCV pour les tâches de détection, nous allons rapidement comparer sa vitesse d'inférence face à Darknet sur CPU. Cela nous indiquera ainsi si notre solution est la bonne ou si nous pouvons encore gagner des performances en se tournant un autre framework.

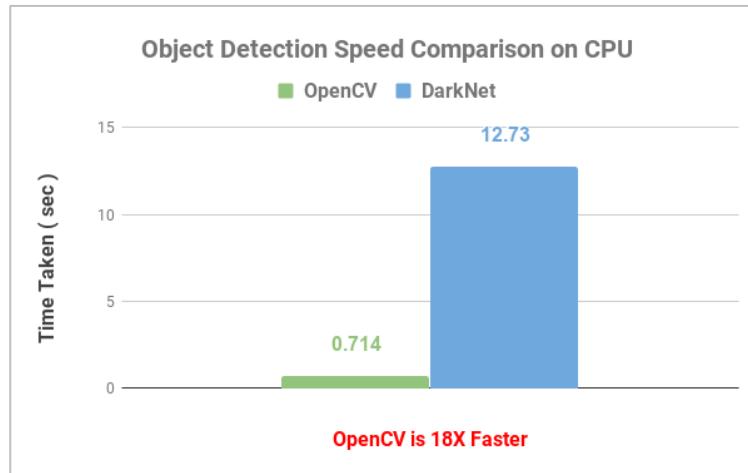


Figure 31.1 : Comparaison de la vitesse de OpenCV et Darknet sur YOLOv3<sup>1</sup>

Nous constatons que l'implémentation de YOLOv3 sur CPU de OpenCV est bien plus performante que celle de Darknet. La raison est que OpenCV est de base un produit de Intel et ils ont donc optimisé la vitesse d'inférence sur CPU Intel, cela suffit, car OpenCV a pour objectif de faire que de l'inférence et non pas de l'entraînement. Darknet a été pensé pour être utilisé avec un GPU et dans ce cas de figure, il est bien plus rapide. Le support du GPU pour le module DNN de OpenCV est cependant prévu<sup>2</sup>, cela nous donnera un bon boost de performances sur hardware compatible.

### 31.2 FPS théoriques sur CPU Intel i7-4770

La machine utilisée a **un CPU Intel i7-4770**. Cette partie vise à nous indiquer quels sont les FPS que nous pouvons espérer atteindre sur ce hardware en considérant que le stream n'est pas limité FPS qu'il peut fournir. On pourra ainsi trouver les modèles qui sont les plus et les moins gourmands en ressources et donc faire un choix en fonction du hardware sur lequel on fera l'implémentation.

<sup>1</sup> <https://www.learnopencv.com/cpu-performance-comparison-of-opencv-and-other-deep-learning-frameworks/>

<sup>2</sup> [https://github.com/opencv/opencv/wiki/GSoC\\_2019#idea-allow-the-opencv-deep-neural-net-module-dnn-to-work-with-gpus](https://github.com/opencv/opencv/wiki/GSoC_2019#idea-allow-the-opencv-deep-neural-net-module-dnn-to-work-with-gpus)

L'axe des X indique le "**Detection Rate**". C'est simplement : toutes les combien de frames effectuons-nous la détection ? Un "1" indique qu'on fait la détection chaque frame et un "10" qu'on fait la détection tous les dix frames et que le reste du temps on fait le tracking. Voir chapitre : "Détection chaque n frame".

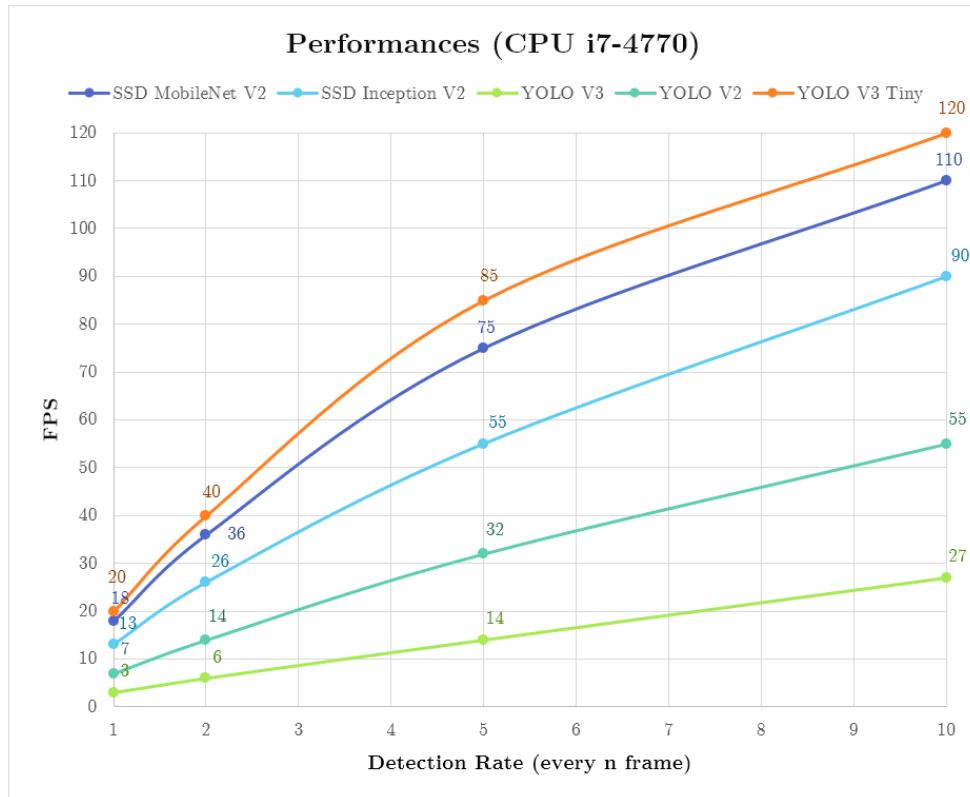


Figure 31.2 : FPS des différents modèles en fonction du Detection Rate

Les résultats sont clairs, YOLOv3 et YOLOv2 sont les plus gourmands en ressources et les modèles de détection SSD sont bien gourmands. Nous pouvons également voir l'influence du taux de détection sur les FPS. Cela confirme ce qui a été dit dans la Conception : faire la détection est plus couteux en performances que de faire du tracking. En faisant donc la détection chaque n frame, nous augmentons les performances.

La vitesse n'est cependant pas tout, il faut mettre ces résultats en relation avec la fiabilité qu'offrent les modèles. Un système de détection ultra rapide, mais ayant une fiabilité médiocre ne sert à rien. Idem pour le Detection Rate, rien ne sert de l'augmenter si on perd la fiabilité.

### 31.3 Temps d'inférence

Le temps d'inférence est le temps que met le modèle de Deep Learning à faire une prédiction. C'est lui qui a la plus grosse influence sur le nombre de FPS qui ont été testés ci-dessus. Un modèle plus lourd aura un temps d'inférence plus long et inversement.

Dans nos tests en offline, cela n'a pas d'impact, mais sur un flux en live oui. Durant l'inférence, on bloque sur une frame durant un certain temps, mais le live continue de son côté sans nous attendre, des frames seront donc dropées. Si le temps d'inférence est trop long, cela pose un problème au tracking qui devra reprendre sur une personne qui aura déjà eu le temps de faire un mouvement conséquent dans le cadre. Il faut donc s'assurer que le temps d'inférence soit assez faible pour garantir le tracking. Une solution pourrait être d'implémenter du multithreading afin de ne rien perdre.

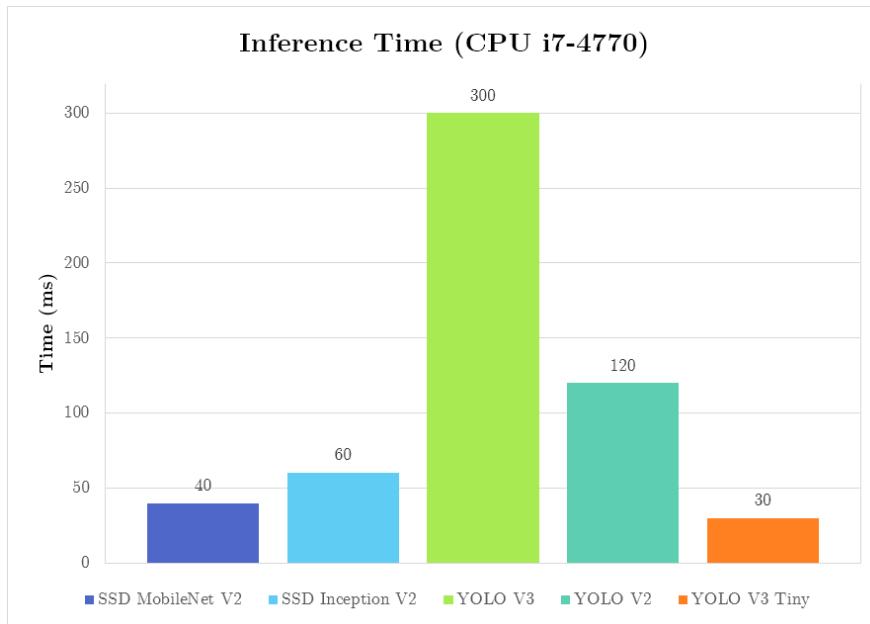


Figure 31.3 : Comparaison des temps d'inférence des différents modèles

## 32 Fiabilité

Maintenant que nous avons une idée de la vitesse des différents modèles, nous pouvons en évaluer la fiabilité et mettre les deux notions en relation. Pour ce faire, le Ground Truth des extraits pré-enregistrés a été défini et les différents modèles ont été lancées sur ces extraits. Leurs résultats (nombre de personnes entrées et sorties) est à la fin comparée au Ground Truth pour pourvoir en déduire une fiabilité en pourcentage.

### Ground Truth :

- OUT : 140 personnes (100%)
- IN : 172 personnes (100%)

## 32.1 Fiabilité de la détection d'objets

Les solutions de détection d'objets n'offrent pas la même fiabilité. Celles-ci ont déjà été étudiées dans le chapitre d'analyse technologique dans la partie "Comparaison des performances". Je vais ici simplement illustrer ces différences au moyen d'exemple :



Figure 32.1 : De gauche à droite et de haut en bas : YOLOv2, YOLOv3, YOlov3 Tiny



Figure 32.2 : De gauche à droite : SSD Inception V2, SSD MobileNet V2

Les constatations globales sont les suivantes :

- **SSD** est un détecteur d'objets agressif qui aura tendance à détecter plus d'objets, mais pourra se tromper, il est également moins robuste à maintenir une détection sur une personnes et aura tendance à "sautiller".
- **YOLO** est un détecteur d'objets plus conservateur, il oubliera certains objets, mais est très fiable sur les entités qu'il daigne détecter.

Il n'y a pas de gagnant clair, les deux offrent des avantages et des inconvénients. Il faudra voir dans un cas d'application réel lequel nous offre la meilleure fiabilité. Il est important d'avoir un système de détection qui soit le meilleur possible, car le tracking en dépend entièrement pour être fourni en BBox.

Pour les tests, la valeur "threshold" a été fixé à 30%, c'est-à-dire que si la probabilité que l'objet détecté soit un humain est inférieur à 30%, on l'ignore. En augmentant cette valeur on diminue les faux-positifs mais on ignore potentiellement des détections qui seraient corrects. Il faut trouver un juste milieu.

## 32.2 Fiabilité et comparaison des modèles

Le premier test vise à évaluer la fiabilité des modèles dans un cas d'utilisation "normal", c'est-à-dire sur les extraits longs de deux heures qui ont été préalablement enregistrées.

Comme indiqué dans le chapitre "Positionnement du boîtier", deux points de vue sont testés. Nous allons comparer les résultats obtenus sur les deux points de vue afin de nous indiquer si le positionnement de la caméra lors de sa mise en place est important ou non :

- Point de vue de face
- Point de vue à 45 degrés

Nous avons deux valeurs à tester : personnes qui sont rentrées (**IN**) et personnes qui sont sorties (**OUT**). Les deux valeurs sont séparées et auront chacune leur propre statistique.

On lance donc les tests sur les deux points de vue et pour chaque valeur de "Detection Rate" pour chacun des modèles. Pour rappel, le "**Detection Rate**" est sur l'axe des **X** et correspond à : toutes les combien de frames effectuons-nous la détection ? Un "1" indique qu'on fait la détection chaque frame et un "10" qu'on fait la détection tous les dix frames et que le reste du temps on fait le tracking.

### 32.2.1 Points de vue de face

Le premier point de vue testé est celui de face, caméra positionnée perpendiculaire à la porte en hauteur. Ce point de vue a le grand avantage de n'avoir quasiment aucune occlusion, les personnes étant filmées de dessus ne se cachent pas entre-elles.



Figure 32.3 : Exemple d'image du point de vue de face tiré de l'extrait long

### Personnes qui sont entrées (IN)

Nous déduisons la fiabilité de cette métrique en comparant les résultats des modèles avec le Ground Truth des personnes qui sont entrées (IN = 172 personnes).

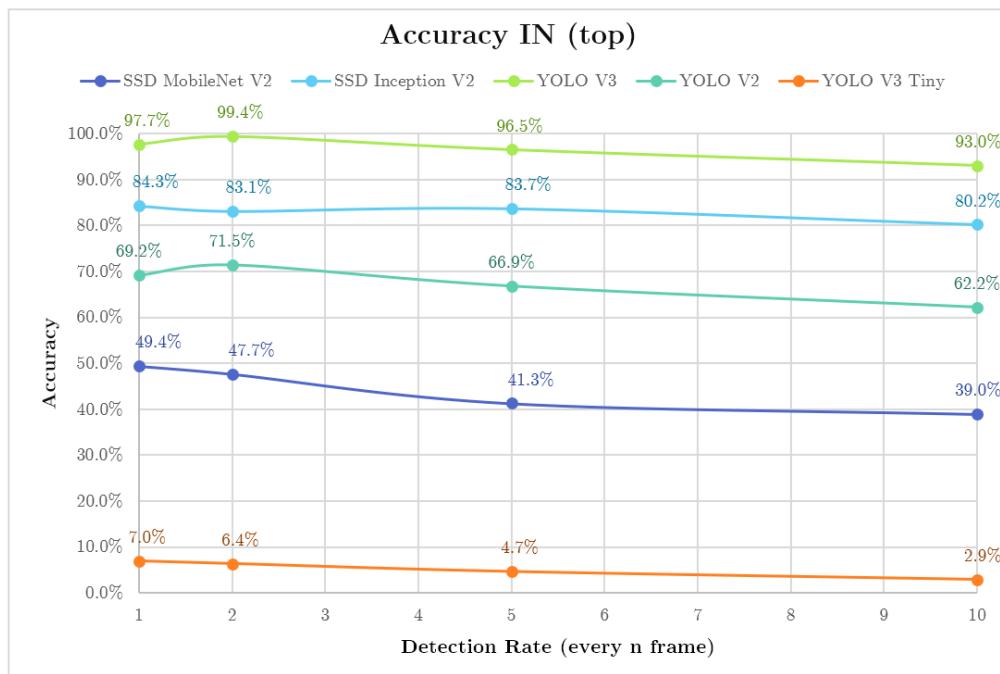


Figure 32.4 : Visualisation de la fiabilité des modèles en fonction du Detection Rate (IN)

## Personnes qui sont sorties (OUT)

On compare les résultats retournés par les modèles pour les personnes qui sont sorties et le Ground Truth (OUT = 140 personnes).

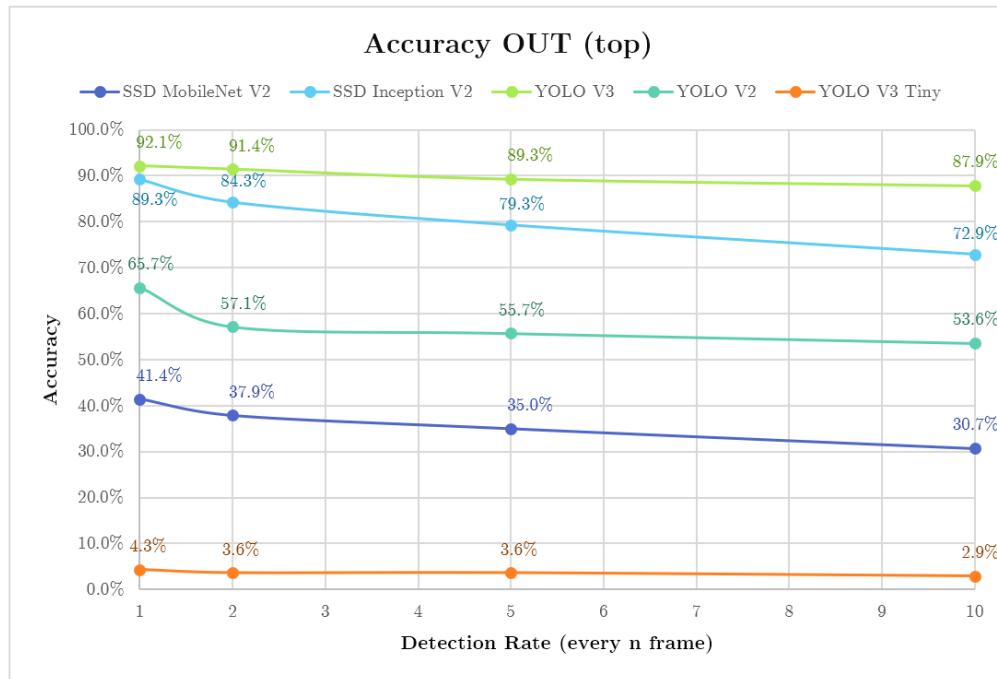


Figure 32.5 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT)

## Constatations

Nous pouvons constater que la différence de fiabilité entre les personnes qui entrent et qui sortent varie d'environ **5-10%**. Ce s'explique par le fait que les systèmes de détection parviennent mieux à reconnaître une personne de face que de dos, notamment grâce au visage.

Nous pouvons également constater que les modèles YOLOv3 Tiny, SSD MobileNet ou YOLOv2 n'offrent absolument pas une fiabilité suffisante, même dans un cas permissif. Alors que YOLOv3 et SSD Inception (avec un Detection Rate bas) offrent une fiabilité SoTA.

Si l'on met en relation ces résultats avec la vitesse des modèles, une tendance se dessine. Plus un algorithme est gourmand en ressource, plus il est fiable. SSD Inception est l'exception en proposant une fiabilité exemplaire tout en étant peu gourmand.

La dernière constatation est que, dans la plupart des cas, une diminution du Detection Rate augmente la fiabilité. En faisant plus souvent la détection, on peut très vite rattraper les erreurs faites par le tracking (un tracking sur une personne qui a été perdu par exemple). On voit cependant que pour le sens OUT, la fiabilité augmente drastiquement avec un Detection Rate de 1 (comportement attendu), mais que dans le sens IN, elle tombe pour YOLOv2 et YOLOv3. Nous n'avons pas réussi à comprendre ce comportement inattendu.

### 32.2.2 Points de vue de côté

Pour le second point de vue, la caméra est positionnée à 45 degrés face à la porte en hauteur. Il a le désavantage d'avoir plus d'occlusion que pour le point de vue de dessus. Il est intéressant de s'y intéresser afin de savoir à quel point le positionnement de la caméra est important.

En ayant déjà fait et analysé les mesures de fiabilité pour le point de vue de face, j'ai choisi d'éloigner YOLOv3 Tiny et SSD MobileNet des tests. Ils ont prouvé qu'ils n'étaient pas des modèles fiables pour notre cas d'application et qu'ils allaient uniquement me faire perdre un temps précieux lors des tests. J'évite ainsi de lancer des tests dont l'issue était quasiment certaine.

J'ai cependant choisi de garder YOLOv2 malgré sa fiabilité moyenne, il peut en effet proposer un rapport fiabilité/performances intéressant.



Figure 32.6 : Exemple d'image du point de vue de côté tiré de l'extrait long

### Personnes qui sont entrées (IN)

Rappel : Ground Truth des personnes qui sont entrées : IN = 172 personnes).

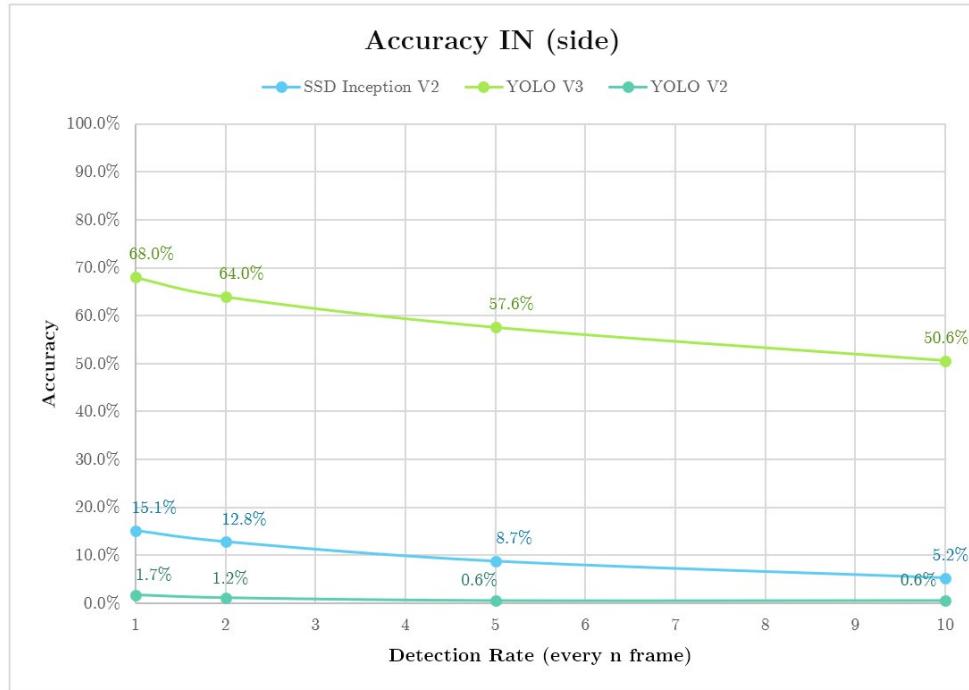


Figure 32.7 : Visualisation de la fiabilité des modèles en fonction du Detection Rate (IN)

### Personnes qui sont sorties (OUT)

Rappel : Ground Truth des personnes qui sont sorties : OUT = 140 personnes).

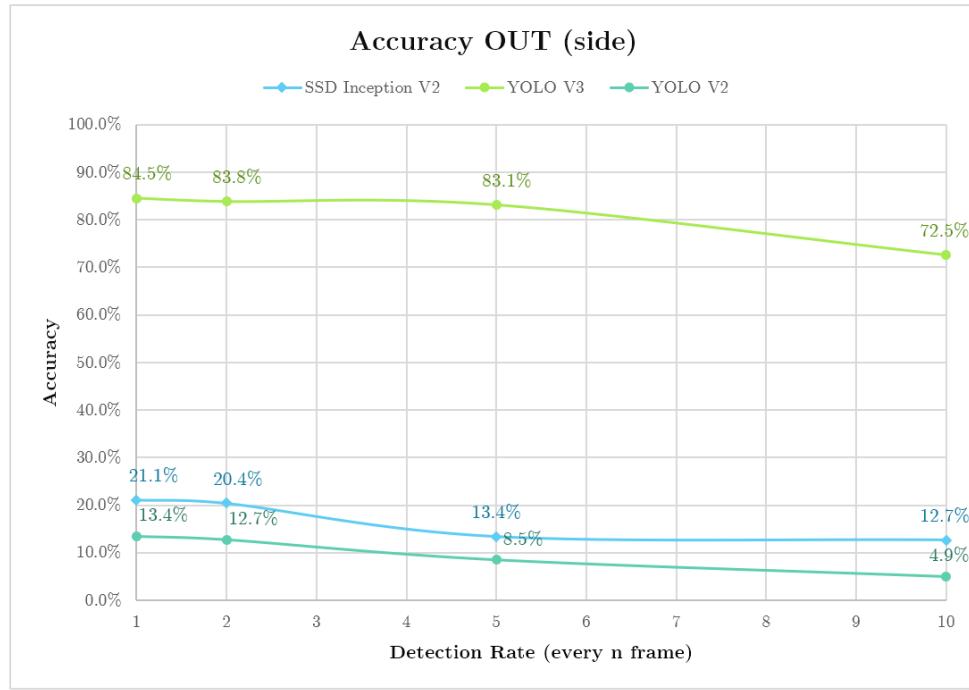


Figure 32.8 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT)

## Constatations

On peut constater que les résultats sont catastrophiques. Par exemple, YOLOv3, pour un Detection Rate de 10, a perdu **43 %** dans le sens IN et **15.5 %** dans le sens OUT. Cela s'explique, car les modèles de détection SSD et YOLO ont été entraînés pour reconnaître des personnes se tenant droite et étant à la verticale. Si on leur donne une image inclinée de plusieurs degrés, ils peineront à faire la tâche de détection.

On peut également constater que la fiabilité pour les personnes qui entrent est plus élevée que pour les personnes qui sortent, exactement l'inverse du résultat obtenu pour le point de vue de dessus. Nous ne savons pas d'où provient ce comportement, mais avant de s'y intéresser, il faut refaire le même test en redressant le cadre afin d'avoir un test valable.

### 32.2.3 Redressement du cadre du point de vue de côté

On a constaté ci-dessus que l'inclinaison de la caméra pouvait poser des problèmes pour la détection. On redresse donc le cadre au moyen de OpenCV et on relance les tests pour voir si le problème se situe vraiment là.

```

h, w = frame.shape[:2]
center = (w/2, h/2)
scale = 1.0

M = cv2.getRotationMatrix2D(center, 25, scale)
straightFrame = cv2.warpAffine(frame, M, (w,h))

```

Figure 32.9 : Code pour incliner le flux vidéo de 25 degrés antihoraire

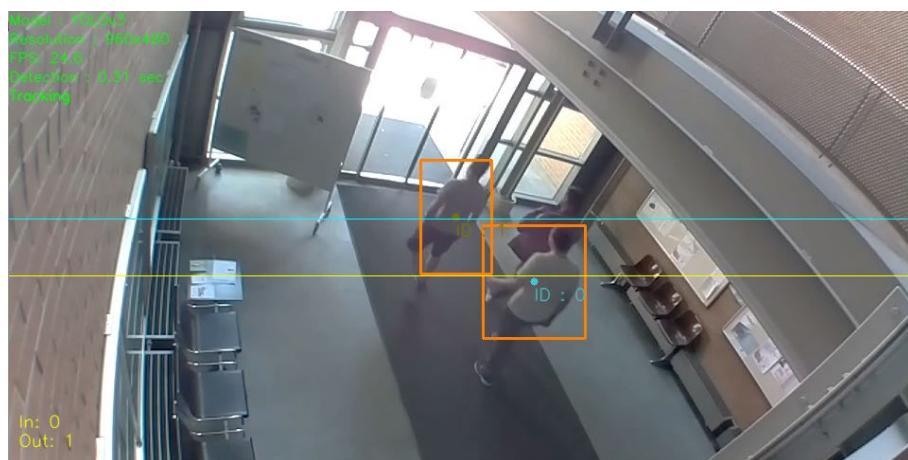


Figure 32.10 : Frame brut avant redressement

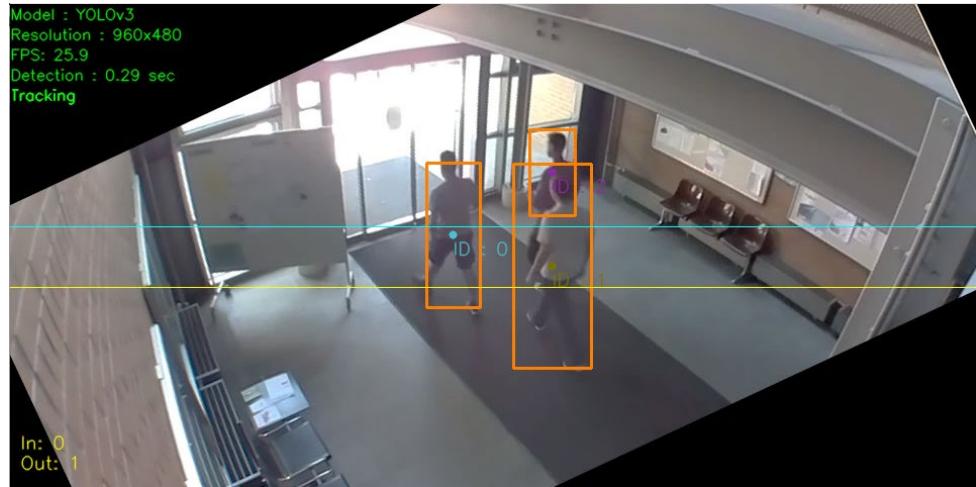


Figure 32.11 : Frame après l'avoir incliné de 25 degrés antihoraires

### Personnes qui sont entrées (IN)

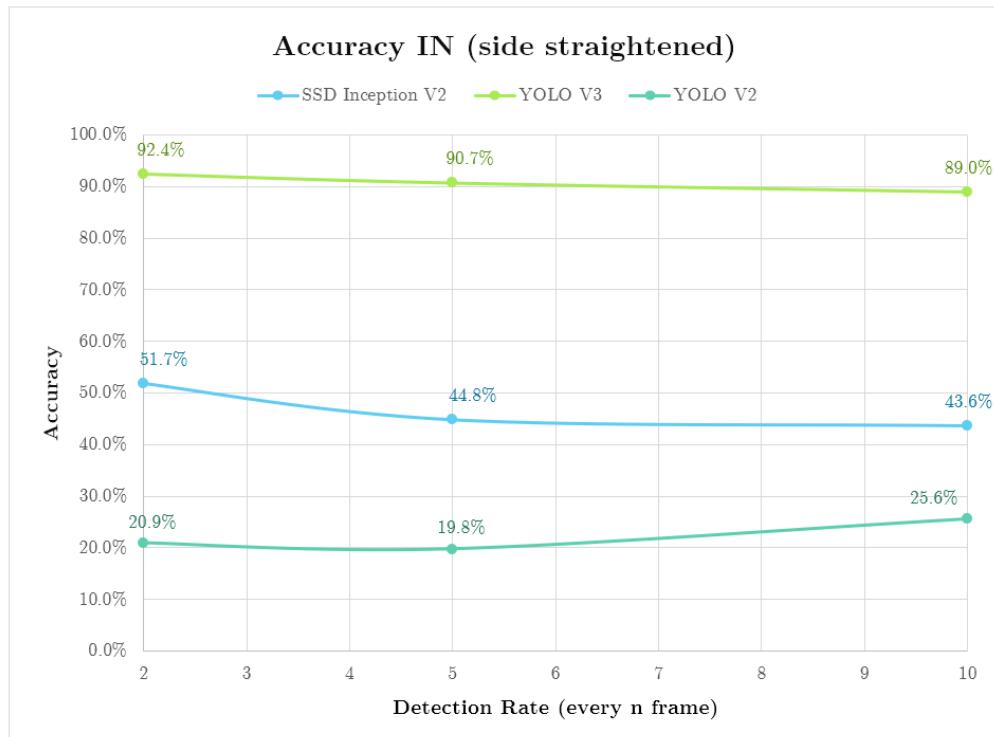


Figure 32.12 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT)

### Personnes qui sont sorties (OUT)

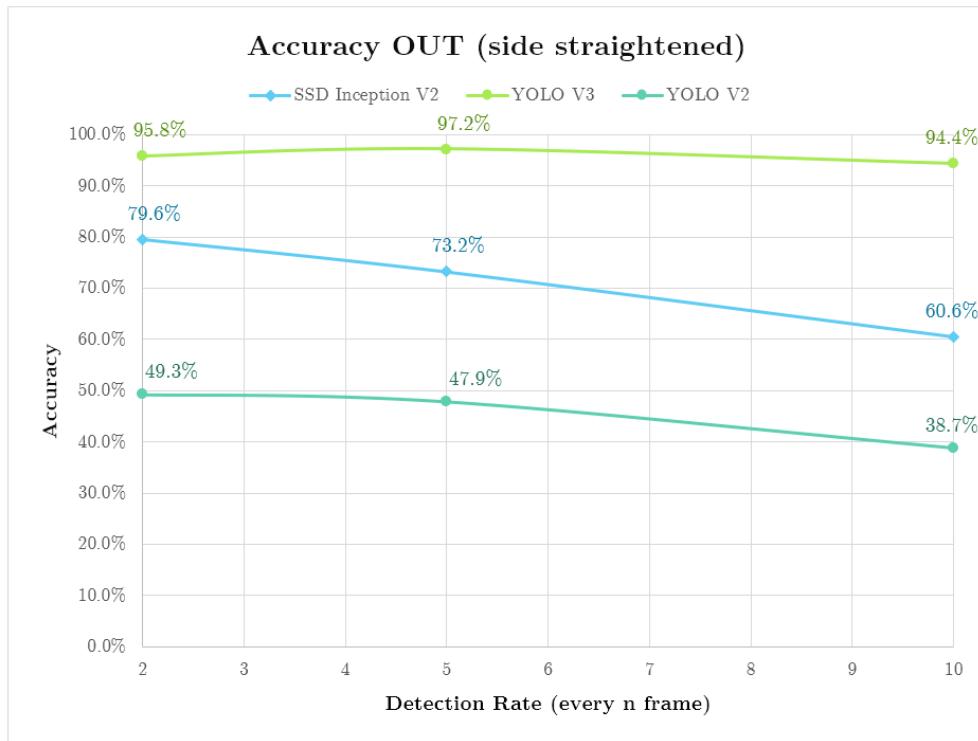


Figure 32.13 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT)

### Constatations

Et bingo, c'était bien l'inclinaison du point de vue qui causait la faible fiabilité, la fiabilité qui a drastiquement augmenté nous le prouve, car aucun autre changement n'a été fait. Dans le cas de YOLOv3, on gagne 39% dans le sens de l'entrée et 22% dans le sens de la sortie pour un Detection Rate de 10.

La fiabilité n'est cependant pas autant bien que pour le point de vue de dessus, cela s'explique, car on a plus d'occlusion et qu'on n'arrive donc pas détecter certaines personnes.

### 32.3 Influence du nombre de FPS sur la fiabilité

Il est intéressant de savoir à quel point le nombre de FPS du streaming influence la fiabilité. Cela pourrait nous éviter de streamer à 30 FPS constamment afin d'économiser la charge sur le réseau et de ménager la workstation. Dans un cas d'application en Edge Computing, la vitesse sera également plus faible et on pourra donc adapter la solution aux nombres de FPS que le dispositif est capable de donner. Par soucis de temps, les tests ont été faits uniquement sur le modèle SSD Inception V2 avec un Detection Rate de 10, 5 et 2.

### Personnes qui sont entrées (IN)

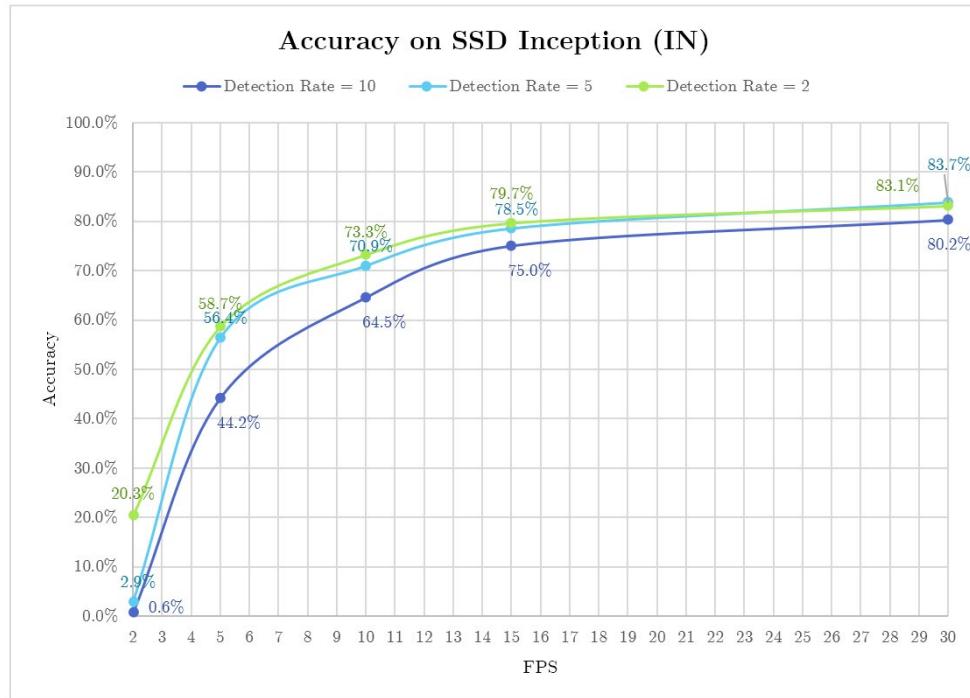


Figure 32.14 : Visualisation de la fiabilité selon les FPS (IN)

### Personnes qui sont sorties (OUT)

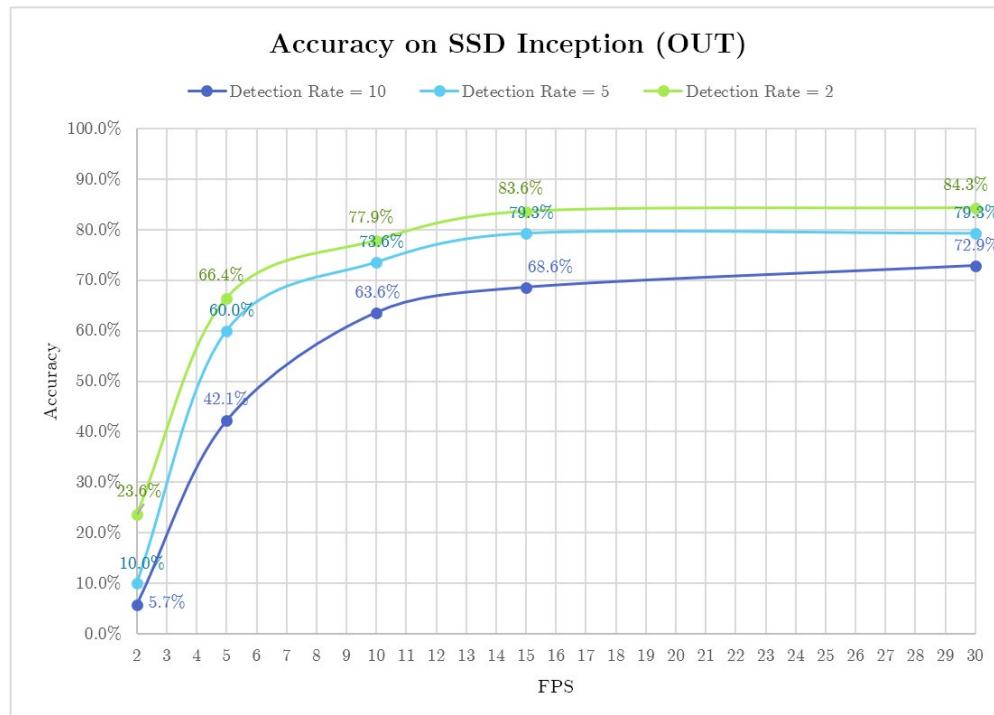


Figure 32.15 : Visualisation de la fiabilité selon les FPS (OUT)

## Constatations

Il est clair que le nombre de FPS influence la fiabilité. Si le nombre de FPS diminue, une personne dans le cadre aura le temps de parcourir une distance plus grande entre deux images et le système de tracking aura donc de la peine à le suivre. Pour pallier ce problème, on pourrait modifier l'algorithme de tracking pour qu'il implémente un système de prédiction de position futur (Filtre de Kalman).

Nous pouvons avoir un système qui fonctionne aux alentours de 15 FPS au minimum avant d'observer une dégradation de la fiabilité.

## 32.4 Récapitulatif des résultats des tests

Voir : Annexe 2 : "Résultats des tests"

## 32.5 Scénarios spécifiques

En plus des tests globaux sur l'extrait de deux heures qui représente un cas d'utilisation "typique", nous avons également effectué des tests sur des scénarios particuliers qui peuvent poser problèmes. Ceci afin de juger quel système arrive le mieux à les gérer et aussi afin de mettre le doigt sur ce qui peut nous faire gagner quelque point de fiabilité (Fine Tuning). Ces tests sont qualitatifs et non quantitatifs. Si rien n'est précisé, la distance entre les personnes du groupe est d'environ 50 cm.

Uniquement les modèles avec un Detection Rate de 5 ont été testés et uniquement les systèmes les plus fiables : YOLOv3, YOLOv2 et Inception V2. Les scénarios sont :

1. Groupe qui entre (trois groupes de six personnes)
2. Groupe qui sort (trois groupes de six personnes)
3. Groupes qui se croisent de manière désordonnée (2x2 groupes de trois personnes)
4. Personnes qui sont collées (deux personnes)
5. Personne qui sort en courant : (2x1 personnes)
6. Personne qui entre en courant : (2x1 personne)
7. Personne qui entre en poussant un objet volumineux (une personne)
8. Personne qui sort en poussant un objet volumineux (une personne)
9. Personne qui zigzague dans le cadre (2x1 personne)
10. Personne qui fait demi-tour (2x1 personne)

**OK** : Scénario validé (90-100%)

**KO** : Scénario non validé (0-20%)

**PARTIEL** : Scénario partiellement validé (30-80 %)

### 32.5.1 Point de vue de face

Scénario	YOLOv3	SSD Inception V2	YOLOv2
1.	OK	OK	PARTIEL
2.	OK	PARTIEL	PARTIEL
3.	OK	PARTIEL	PARTIEL
4.	OK	OK	OK
5.	OK	OK	OK
6.	OK	OK	OK
7.	OK	OK	KO
8.	OK	OK	KO
9.	OK	KO	KO
10.	OK	OK	OK

### 32.5.2 Point de vue de côté (redressé)

Scénario	YOLOv3	SSD Inception V2	YOLOv2
1.	PARTIEL	PARTIEL	PARTIEL
2.	OK	PARTIEL	PARTIEL
3.	PARTIEL	KO	KO
4.	PARTIEL	PARTIEL	KO
5.	OK	OK	OK
6.	OK	OK	OK
7.	OK	OK	KO
8.	OK	OK	KO
9.	OK	KO	KO
10.	OK	OK	OK

### 32.5.3 Constatations

Nous pouvons voir que les différences de fiabilité entre le point de vue de dessus et de côté sont dues à l'occlusion. Le point de vue de côté cache des personnes qui marchent en étant rapprochées.

On voit également que, de manière générale, ce sont les groupes qui posent des soucis de fiabilité. Si nous savons que dans notre cas d'application il y aura beaucoup de groupes, il faudra absolument se tourner vers YOLOv3.

On peut également juger des points forts et des points faibles de chacun des systèmes, YOLOv2 par exemple est plus sensible que SSD Inception sur certains scénarios.

---

## Synthèse

Ces tests ont été d'une grande utilité afin de savoir quel genre de performances nous pouvons espérer de notre système. Ces benchmarks permettent de définir le champ d'application du projet. Est-ce que sa fiabilité est suffisante pour permettre de le déployer ?

Le but premier du projet DeepCounter est de compter des gens à but statistique. Dans cette optique, notre fiabilité qui avoisine les 90-95% pour YOLOv3 est totalement adapté. Il faut cependant des machines puissantes derrière pour supporter le système de détection qui est lourd. Dans le cas où nous devons faire du Edge Computing, nous avons SSD Inception comme alternative. Nous avons estimé sa fiabilité dans un cas d'utilisation "normal", en connaissant cela, nous pourrions augmenter le nombre de personnes comptées artificiellement afin de mieux coller à la réalité.

Dans un cas d'utilisation nécessitant une fiabilité à toute épreuve (compter les gens restant dans le bâtiment en cas d'incendie par exemple), notre système est à exclure, il n'est pas assez fiable. Sa faiblesse étant les groupes, il ne serait non plus pas adapté à un cas d'application où il y en a beaucoup.

Il faut aussi être conscient que le dataset de test est limité, plusieurs cas spéciaux n'ont pas été mesurés par manque de temps. Comment se comporte le système la nuit par exemple ? Il faut donc prendre ces résultats pour ce qu'ils sont : un simple échantillon.



## VIII. CONCLUSIONS

---

### 33 Conclusion du projet

---

#### 33.1 Validation des objectifs

1. **Fournir un système de capture de vidéo envoyant le flux en direct sur le réseau afin qu'elles soient utilisées sur une workstation distante.**

Objectif validé, on envoie depuis le RPI un flux vidéo sur le LAN qu'on récupère via une adresse IP sur la workstation.

2. **Fournir sur une workstation un système de Deep Learning permettant de détecter en temps réel les personnes depuis le flux vidéo**

Objectif validé, on utilise le module DNN de OpenCV pour faire de la détection d'objets dans le flux vidéo. On peut utiliser alternativement YOLO ou SSD comme modèle.

3. **Il faut fournir une logique de comptage de ces personnes depuis les données retournées par la détection de personnes**

Objectif validé, on utilise les Bounding Boxes retournées par le système de détection pour faire le suivi des personnes qui se trouvent dans le cadre. Le suivi est ensuite utilisé pour définir si les personnes entrent ou sortent du bâtiment.

4. **Un système de mesure et d'évaluation des performances doit également être proposé**

Objectif validé, un dataset a été construit et des scénarios définis afin d'effectuer des tests qui soient reproductibles. Les résultats de ces tests sont sous formes de graphiques.

5. **Il faut envoyer les données fournies par le système sur BBData**

Objectif validé, le nombre de personnes entrées et sorties sont envoyées sur BBData à chaque évènement mettant à jour les compteurs.

## 6. Une étude économique doit être faite afin de juger du cadre légal de la surveillance de personnes et d'ensuite juger le potentiel commercial

Objectif validé, on connaît le cadre légal de la vidéosurveillance en Suisse, on a modélisé la proposition de valeur du projet, on connaît son offre, la demande et la concurrence.

## 33.2 Problèmes rencontrés et solutions apportées

### Données des tests perdues

A cause d'une erreur de manipulation de Git, j'ai perdu trois jours de récolte de données pour les tests. J'ai pu rattraper mon erreur en demandant un autre ordinateur afin d'effectuer plusieurs tests en parallèle. Je suis parvenu à fournir des tests complets à temps malgré cette erreur de manipulation de ma part.

### Tests très chronophages

L'extrait de test en condition réels fait plus de deux heures. On a cinq modèles de détection à tester à quatre Detection Rate différent et pour deux points de vue de caméra. Cela fait de 40 tests à lancer qui prennent, en fonction du modèle de détection utilisé, d'une heure à cinq heures à compléter. J'y suis parvenu en lançant les tests la nuit et sur plusieurs machines en parallèle. J'ai également choisi d'éloigner les modèles qui dès le départ montraient une fiabilité non suffisante.

### Doutes quant à la législation concernant le projet

Pour la partie économique, je me suis retrouvé à me poser beaucoup de questions quant à la réglementation à suivre pour notre projet qui n'est pas vraiment de la vidéosurveillance. Les textes de loi fourni par la confédération sont évasifs sur le sujet. Je me suis tourné vers des experts afin de m'apporter des réponses (voir "Réglementation").

### Volonté de faire du "temps réel"

J'avais mal compris le but du projet et m'étais concentré sur faire un seul système que je poussais au maximum afin d'avoir une démo présentable qui soit le meilleur possible. J'ai revu mes objectifs durant la conception afin de plutôt faire un éventail des solutions et de les comparer, le projet étant plus un "Proof of Concept" qu'un produit fini.

### 33.3 Perspectives futures

#### Remise à zéro des compteurs

Pour le moment, les compteurs s'incrémentent indéfiniment. Il faudrait, en fonction de l'heure (à minuit par exemple), remettre les compteurs IN et OUT à zéro afin de ne pas les overflow.

#### Point de vue du dessus

Nos points de vue de tests sont positionnés en hauteur, de face et à 45 degrés. On pourrait également tester les performances du système en positionnant la caméra directement en dessus de la porte et regardant vers le bas. Voir si la détection est capable de gérer ce cas d'application.

#### Implémentation GPU

Il serait également très intéressant de tester les performances avec un GPU et de les comparer à nos résultats. On pourrait ainsi potentiellement utiliser des méthodes de détection plus gourmandes ou diminuer le Detection Rate afin d'améliorer encore la fiabilité.

#### Positionnement manuel des limites

Actuellement dans le code, les limites IN et OUT sont positionnées horizontalement et à  $\pm 1/6$  de la hauteur du cadre. Il faudrait, par l'intermédiaire de paramètre ou d'une interface graphique, permettre d'incliner et de déplacer ces limites pour étendre le champ d'application de la situation.

#### SORT/Deep SORT tracking<sup>1</sup>

Je n'ai eu le temps d'implémenter uniquement un système de tracking utilisant la distance d'Euclide et un filtre de corrélation. Il serait intéressant d'implémenter le tracking SORT ou Deep SORT et de regarder si la fiabilité s'en retrouve accrue ou non et mesurer également l'impact sur la vitesse du système.

---

<sup>1</sup> <https://github.com/guillemlopez/python-traffic-counter-with-yolo-and-sort>

### Tests plus avancés

Notre dataset et nos scénarios de tests se veulent représenter un cas d'utilisation "normal", mais il reste un certain nombre de tests à faire pour entièrement éprouver le système. Par exemple : détecte-t-on les enfants ? Comment se comporte le système la nuit ?

## 33.4 Remerciements

Je tiens à remercier du fond du cœur les personnes qui m'ont encadré, renseigné, soutenu et aidé tout au long de ce projet. Merci pour leur précieuses remarques et informations qui m'ont permis d'améliorer mon travail et d'en être fier

- **Docteur Houda Chabbi et Docteur Jean Hennebert**, professeurs à la HEIA-FR, pour leur suivi et leur supervision en tant que responsables de travail de bachelor durant l'entier des huit semaines.
- **Docteur Emeka Mosanya et Docteur Julien Bégard** pour leur rôle d'experts. Merci pour leur suivi attentif et leur curiosité.
- **Madame Flavia Pittet et Monsieur Matthieu Jourdan** collaborateurs à la HEIA-FR pour leur aide, leur enthousiasme et leurs précieux conseil.
- **Monsieur Fabian Sipp** (Morphean SA), **Monsieur Michael Koch** (Xovis AG) et **Docteur Jean-Philippe Bacher** pour les précisions qu'ils m'ont apporté sur le sujet de la vidéosurveillance en Suisse et sur son aspect économique.
- Mes proches pour leur soutien inconditionnel durant tout mon cursus

## 33.5 Conclusion du projet

Comme indiqué ci-dessus, les objectifs principaux du cahier des charges ont été remplis et le projet est, selon moi, une réussite. J'ai créé un prototype et prouvé que le comptage de personnes au moyen de techniques de Deep Learning peut être fait de manière fiable. Des tests approfondis ont été effectuées pour le justifier ainsi qu'une étude économique afin de juger du cadre légal et du potentiel commerciale du projet.

Il est frustrant pour moi de ne pas avoir pu pousser la réalisation plus loin, implémenter d'autres algorithmes de suivi, tester d'autres systèmes de comptage, faire la détection avec l'implémentation Python de Darknet plutôt qu'avec OpenCV... Il faut cependant garder en tête que le projet est un "Proof of Concept" et que son rôle est de juger de la réalisabilité.

Le temps est compté et il faut savoir conclure. Je suis confiant que ce projet et ce rapport constituent une base solide pour quiconque souhaite en reprendre le flambeau et finaliser le long voyage qui a été entrepris. Les différentes analyses et les tests permettent de juger aisément du potentiel du projet et de diriger les décisions futures.

### 33.6 Conclusion personnelle

J'ai été attiré très vite par ce projet, car il constituait une aubaine pour moi de m'atteler au domaine du Computer Vision qui me rendait plus que curieux depuis quelque temps déjà. J'avais déjà eu l'occasion de m'intéresser au Deep Learning et aux CNN lors d'un précédent projet et ce travail était une occasion d'approfondir mes connaissances encore limitées dans l'intelligence artificielle. J'ai profité de cette porte d'entrée pour m'y mettre et ne regrette pas une seule seconde d'avoir eu le "courage" de me lancer dans ce domaine si vaste dont je ne connaissais finalement encore pas grand-chose. Cela a occasionné de nombreux questionnements et problèmes liés intimement à ma méconnaissance du sujet, mais cela m'a forcé à sortir hors de ma zone de confort et à faire un vrai effort d'apprentissage en autodidacte. Je ressors mûri de cette expérience et me suis découvert un vrai intérêt pour ce domaine en plein essor.

Je me sens un peu déçu de n'avoir utilisé que des modèles pré-entraînés et de ne pas avoir eu la possibilité ou le temps de faire du transfer learning ou de ré-entraînement. Déçu également de n'être pas parvenu à utiliser la puissance d'un GPU afin d'accélérer mon système. J'aurais peut-être souhaité de ce travail une plus grosse partie réflexion et développement.

Ces huit semaines de travail de Bachelor se sont avérées être correcte. J'ai eu un peu de mal à m'y mettre à fond au départ, fatigué par les examens de fin de semestre et l'enchaînement discontinu des activités. Heureusement, les nuages se sont vite dissipés et je me suis retrouvé un entraînement inconsidéré durant la seconde partie de ce travail.

Je ressors grandi de ce travail de bachelor. Je suis fier de vous présenter le fruit de mon travail.

## 34 Conclusion du document

---

### 34.1 Licences

Software	Version	License
OpenCV	4.0.1	3-clause BSD License
Dlib	19.17	Boost Software License
Darknet	3	WTFPL
Tensorflow	1.14	Apache License 2.0
Caffe	1.0	BSD
Numpy	1.16.1	BSD
Imutils	0.5.2	MIT License
SciPy	1.2.2	BSD
Requests	2.22.0	Apache License 2.0

### 34.2 Déclaration d'honneur

Je, soussigné, Patrick Audriaz, déclare sur l'honneur que le travail rendu est le fruit d'un travail personnel. Je certifie ne pas avoir eu recours au plagiat ou à toutes autres formes de fraudes. Toutes les sources d'information utilisées et les citations d'auteur ont été clairement mentionnées.

Fribourg, 11 juillet 2019

---

Patrick Audriaz

## IX. REFERENCES

---

1. **Florinel Radu, Jonathan Parrat, Marc Demierre, Jean Hennebert, JeanPhilippe Bacher.** *City Pulse - WP1 - Rapport cas d'utilisation*. Fribourg : s.n., Juin 2018. D1.1.
2. **Marc Demierre, Julien Esseiva, Jean Hennebert.** *City Pulse - WP1 - Rapport de spécifications matérielles et logicielles*. Fribourg : s.n., Juin 2018. D1.2.
3. **Nikhil Buduma, Nicholas Locascio.** *Fundamentals of Deep Learning*. s.l. : O'Reilly, 2017. 978-1-491-92561-4.
4. **Géron, Aurélien.** *Hands on Machine Learning with Scikit Learn and TensorFlow*. s.l. : O'Reilly, 2017. 978-1-491-96229-9.
5. **Chollet, François.** *Deep Learning With Python*. s.l. : Manning Publications Co., 2018. ISBN 9781617294433.
6. **Audriaz, Patrick.** *PS6 - Medical Machine Learning*. Haute école d'ingénierie et d'architecture de Fribourg. Fribourg : s.n., 2019.
7. **Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik.** *Rich feature hierarchies for accurate object detection and semantic segmentation*. UC Berkeley. 2014. arXiv:1311.2524.
8. **Girshick, Ross.** *Fast R-CNN*. 2015. arXiv:1504.08083v2.
9. **Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun.** *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv:1506.01497v3.
10. **Jifeng Dai, Yi Li, Kaiming He, Jian Sun.** *R-FCN: Object Detection via Region-based Fully Convolutional Networks*. 2016. arXiv:1605.06409v2.
11. **Liu, Wei, et al.** *SSD: Single Shot MultiBox Detector*. 2016. arXiv:1512.02325v5.
12. **Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár.** *Focal Loss for Dense Object Detection*. 2018. arXiv:1708.02002v2.
13. **Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi.** *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv:1506.02640.
14. **Joseph Redmon, Ali Farhadi.** *YOLO9000: Better, Faster, Stronger*. 2016. arXiv:1612.08242v1.
15. —. *YOLOv3: An Incremental Improvement*. 2018. arXiv:1804.02767v1.
16. **Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, Xindong Wu.** *Object Detection with Deep Learning: A Review*. 2018. arXiv:1807.05511.

17. Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Kevin Murphy. *Speed/accuracy trade-offs for modern convolutional object detectors.* 2017. arXiv:1611.10012v3.
18. Zhang Fukai, Li Ce, Yang Feng. *Vehicle Detection in Urban Traffic Surveillance Images Based on Convolutional Neural Networks with Feature Concatenation*. 2019. doi:10.3390/s19030594 .
19. Dadhich, Abhinav. *Practical Computer Vision*. s.l. : Packt Publishing, 2018. ISBN : 9781788297684.
20. Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, Ben Upcroft. *Simple Online and Realtime Tracking*. 2016. arXiv:1602.00763.
21. Nicolai Wojke, Alex Bewley, Dietrich Paulus. *Simple Online and Realtime Tracking with a Deep Association Metric*. 2017. arXiv:1703.07402.
22. AG, Xovis. *Data Privacy and Security*. Switzerland : s.n., 2018.
23. Villan, Alberto Fernandez. *Mastering OpenCV 4 with Python*. s.l. : Packt Publishing, 2019. ISBN: 9781789344912.
24. Millan Escriva, David, et al. *Building Computer Vision Projects with OpenCV 4 and C++*. s.l. : O'Reilly, 2019. ISBN: 9781838644673.
25. Mallick, Satya. *Deep Learning with OpenCV*. 2019.

## X. GLOSSAIRE

---

**AI** : Artifical Intelligence (Intelligence artificielle)

**API** : Application Programming Interface (Interface de Programmation)

**ASIC** : Application-Specific Integrated Circuit (Circuit intégré spécifique)

**AWS** : Amazon Web Services

**BBData** : Big Building Data (Base de données utilisée par City Pulse)

**BBox** : Bounding Box

**CCTV** : Closed-Circuit Television (Vidéosurveillance)

**CNN** : Convolutional Neural Network (Type de réseau de neurones artificiels)

**COCO** : Common Objects in Context

**CPU** : Central Processing Unit (Processeur)

**CV** : Computer Vision

**DB** : Databse (Base de données)

**DL** : Deep Learning

**DNN** : Deep Neural Networks

**FPS** : Frames per Second (Images par seconde)

**GPU** : Graphics Processing Unit (Carte Graphique)

**GST** : Gstreamer

**GT** : Ground Truth

**HEIA-FR** : Haute Ecole d'Ingénierie et d'Architecture Fribourg

**HTTP** : Hypertext Transfer Protocol

**iCoSys** : Institut des systèmes complexes (Institut de la HEIA-FR)

**ID** : Identifieur unique

**IoT** : Internet of Things (Internet des objets)

**IoU** : Intersection over Union

**JPEG** : Joint Photographic Experts Group (Format de compression d'image)

**JSON** : JavaScript Object Notation (Format de données textuelles)

**LPD** : Loi fédérale sur la protection des données

**mAP** : Mean Average Precision

**MJPEG** : Motion JPEG

**ML** : Machine Learning

**MQTT** : Message Queuing Telemetry Transport (Protocole pour l'IoT)

**NC** : Netcat (Utilitaire pour ouvrir des connexions TCP ou UDP)

**NN** : Neural Network

**OpenCV** : Open Source Computer Vision Library

**OS** : Operating System (Système d'exploitation)

**PoE** : Power over Ethernet

**R-CNN** : Resion Based CNN

**REST** : Representational State Transfer

**RGPD** : Règlement général de l'UE sur la protection des données

**RPI** : Raspberry Pi (Nano-ordinateur à processeur ARM)

**RTSP** : Real Time Streaming Protocol

**SGBD** : Système de gestion de bases de données

**SORT** : Simple Online and Realtime Tracking

**SOTA** : State of the Art

**SQL** : Structured Query Language

**SSD** : Single Shot Object Detection

**SSH** : Secure Shell (Terminal à Distance)

**SSL** : Transport Security Layer

**TCP** : Transfer Control Protocol

**TF** : TensorFlow (Librairie open source pour le Deep Learning)

**TL** : Transfer Learning

**TPF** : Transports Publics Fribourgeois

**UDP** : User Datagram Protocol

**VOC** : Visual Object Classes

**YOLO** : You Only Look Once (Détection d'objets en temps réel)

## XI. TABLE DES FIGURES

---

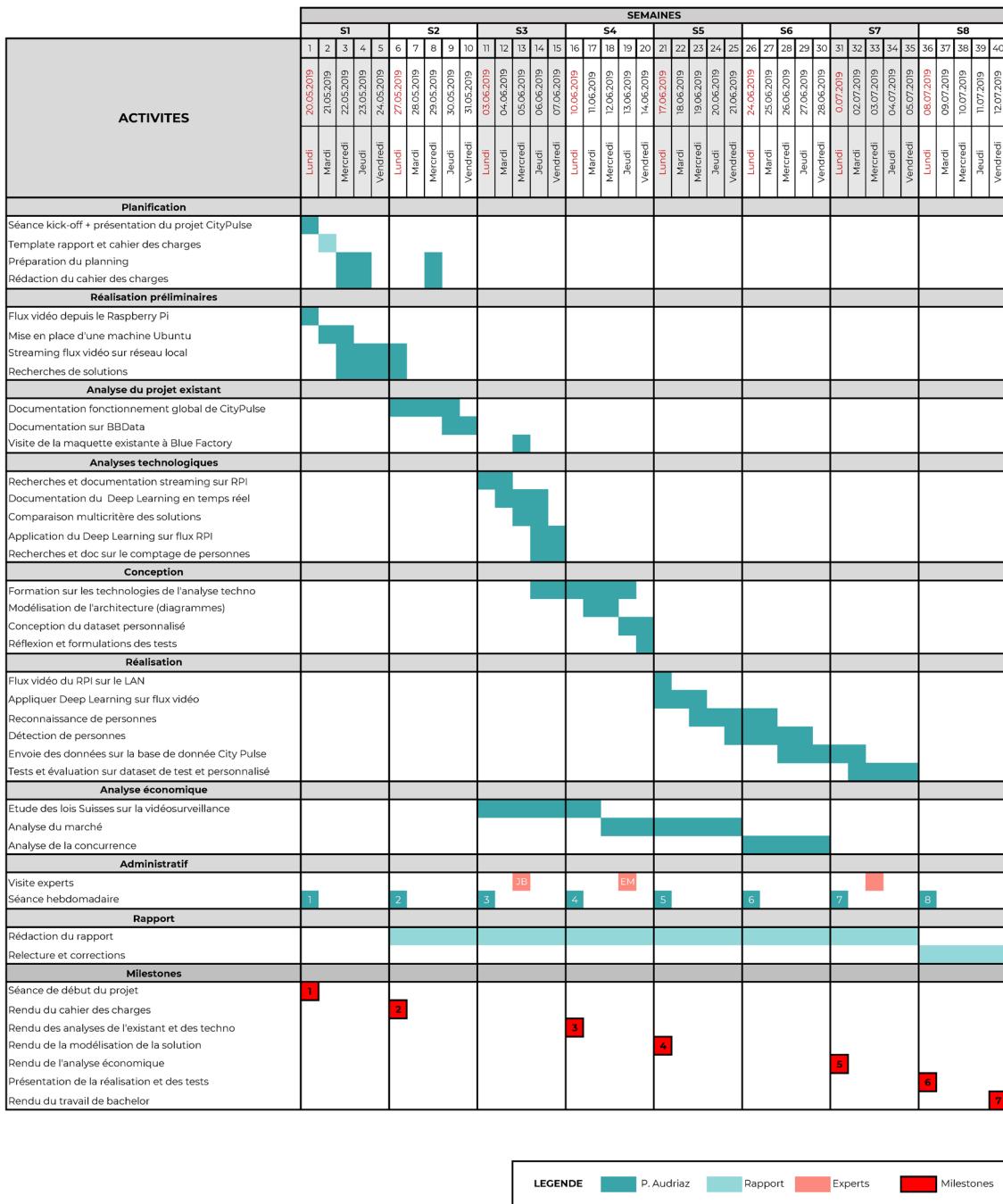
Figure 8.1 : Storyboard du projet City Pulse.....	17
Figure 8.2 : Maquette du prototype actuel. Le taux d'occupation des bâtiments d'habitation est représenté en les illuminant en rouge .....	19
Figure 8.3 : Maquette du prototype actuel. Le taux de densité d'emplois par bâtiment est représenté en les illuminant en bleu. On peut également apercevoir des voitures modélisées sous forme de rectangles sur les axes routiers .....	19
Figure 8.4 : Schéma de collecte de données (Crédit : Flavia Pittet).....	20
Figure 8.5 : Schéma de flux de données (Crédit : Flavia Pittet) .....	21
Figure 9.1 : Logo de BBData.....	22
Figure 9.2 : JSON retourné lorsque l'on effectue une requête GET sur un objet BBData	23
Figure 9.3 : Parcours d'une mesure dans le "pipeline" BBData .....	24
Figure 9.4 : Parcours d'un utilisateur souhaitant utiliser BBData .....	25
Figure 9.5 : Stack technologique de BBData.....	26
Figure 10.1 : Coût de la correction de bugs en fonction de l'avancée dans le cycle de vie	28
Figure 11.1 : Schéma de l'architecture du traitement distant.....	29
Figure 11.2 : Schéma de l'architecture du traitement intégré.....	30
Figure 12.1 : Commandes permettant de faire un stream avec Netcat.....	34
Figure 12.2 : Interface web de UV4L .....	35
Figure 12.3 : Comparaison de l'utilisation de bande passante de MJPEG et H264 (à 30 fps) dans différents scénarios .....	37
Figure 13.1 : Place du CV dans le domaine de l'intelligence artificielle .....	39
Figure 13.2 : Schéma d'apprentissage supervisé, phases d'entraînement et de prédiction.	40
Figure 13.3 : Visualisation des différences entre ML et DL .....	41
Figure 13.4 : Schématisation du fonctionnement d'un modèle CNN .....	42
Figure 13.5 : DNN pour faire de la classification .....	42
Figure 13.6 : Comparaison des différentes architectures CNN .....	44
Figure 13.7 : Différence entre "classification" et "déttection".....	45
Figure 13.8 : Schéma de fonctionnement de R-CNN.....	46
Figure 13.9 : Image originale et prédictions pour chaque case de la grille.....	48
Figure 13.10 : Différents type de Feature Pyramid Network.....	49
Figure 13.11 : Schéma du fonctionnement de YOLO .....	50
Figure 13.12 : Schéma résumant l'évolution des modèles de détection d'objets (16) .....	51
Figure 13.13 : Résumé des performances des différentes architectures (hors YOLO) (17)	52

Figure 13.14 : Comparaison des performances des différents modèles sur GPU Titan X	52
Figure 13.15 : Étude comparant les différentes méthodes (taille de l'image en entrée, fiabilité et FPS) (18).....	53
Figure 13.16 : Tracking online et offline.....	54
Figure 13.17 : Illustration de la distance Euclidienne. Frame précédente (points violets) et frame actuelle (points jaunes).....	55
Figure 13.18 : Exemples de "score" IoU pour différentes bounding boxes.....	56
Figure 13.19 : Visualisation du fonctionnement du système de similarités d'apparence de Deep SORT.....	57
Figure 13.20 : Tracking et identification individuelle des personnes avec Deep Sort .....	57
Figure 13.21 : Comparaison des performances .....	58
Figure 13.22 : Exemple de mise en pratique de comptage unidirectionnel utilisant une frontière” .....	59
Figure 13.23 : Visualisation des différentes catégories que COCO propose .....	60
Figure 15.1 : Niveaux de privacité proposés par XOVIS (22).....	67
Figure 15.2 : Infographie résumant les points importants du RGPD .....	68
Figure 16.1 : Canvas de proposition de valeur du projet DeepCounter.....	70
Figure 16.2 : Interface web du service proposé par Morphéan.....	73
Figure 16.3 : Comparaison des coûts en fonction du nombre de capteurs .....	76
Figure 16.4 : Prédiction de la taille du marché global des caméras intelligentes.....	77
Figure 17.1 : Pièces du boîtier à imprimer en 3D .....	79
Figure 17.2 : RPI dans son boîtier avec le module PoE et le module caméra sur pivot....	80
Figure 17.3 : Emplacement et point de vue des caméras dans l'entrée.....	81
Figure 18.1 : Schéma du flux des données du système effectuant la détection et le tracking sur chaque frame.....	82
Figure 18.2 : Schéma du flux des données du système effectuant la détection et le tracking chaque n frame .....	83
Figure 21.1 : Exemple de modèles mis à disposition pour Tensorflow sur OpenCV.....	87
Figure 23.1 : Schéma du fonctionnement de l'algorithme de comptage.....	89
Figure 23.2 : Schéma du fonctionnement de l'algorithme de comptage.....	90
Figure 25.1 : Création d'un objet VideoCapture.....	93
Figure 25.2 : Boucle sur les images du flux vidéo .....	94
Figure 26.1 : Définition du modèle à utiliser et charge les poids et la configuration.....	96
Figure 26.2 : Définition du modèle à utiliser et charge les poids et la configuration.....	96
Figure 26.3 : Schéma montrant les étapes pour faire une prédiction dans OpenCV .....	97
Figure 26.4 : Création du blob et envoi dans le réseau afin d'avoir une prédiction .....	97

Figure 26.5 : Fonction permettant de filtrer les prédictions et de les afficher .....	98
Figure 26.6 : Object Detection affichant que les humains avec YOLOv3.....	99
Figure 26.7 : Object Detection affichant que les humains avec SSD Inception V2 .....	99
Figure 27.1 : Objet permettant de suivre une personne.....	100
Figure 27.2 : On envoie les données de la détection au tracking Dlib .....	101
Figure 27.3 : Tracking en utilisant la détection de YOLOv3.....	102
Figure 27.4 : Tracking en utilisant la détection de SSD Inception V2 .....	102
Figure 28.1 : Algorithme de comptage .....	103
Figure 28.2 : Deux personnes qui sortent du bâtiment, on peut voir le compteur "OUT" en bas à gauche qui a été incrémenté au moment où ils ont traversé la limite.....	104
Figure 29.1 : Détection chaque n frame.....	104
Figure 30.1 : Objet BBData pour un capteur "IN", son unité est un entier .....	105
Figure 30.2 : Envoie des données sur BBData pour un compteur "IN" .....	106
Figure 31.1 : Comparaison de la vitesse de OpenCV et Darknet sur <b>YOLOv3</b> .....	108
Figure 31.2 : FPS des différents modèles en fonction du Detection Rate .....	109
Figure 31.3 : Comparaison des temps d'inférence des différents modèles .....	110
Figure 32.1 : De gauche à droite et de haut en bas : YOLOv2, YOLOv3, YOlov3 Tiny	111
Figure 32.2 : De gauche à droite : SSD Inception V2, SSD MobileNet V2.....	111
Figure 32.3 : Exemple d'image du point de vue de face tiré de l'extrait long .....	113
Figure 32.4 : Visualisation de la fiabilité des modèles en fonction du Detection Rate (IN) .....	113
Figure 32.5 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT) ...	114
Figure 32.6 : Exemple d'image du point de vue de côté tiré de l'extrait long .....	115
Figure 32.7 : Visualisation de la fiabilité des modèles en fonction du Detection Rate (IN) .....	116
Figure 32.8 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT) ...	116
Figure 32.9 : Code pour incliner le flux vidéo de 25 degrés antihoraire.....	117
Figure 32.10 : Frame brut avant redressement .....	117
Figure 32.11 : Frame après l'avoir incliné de 25 degrés antihoraires .....	118
Figure 32.12 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT) .	118
Figure 32.13 : Visualisation de la fiabilité des modèles selon le Detection Rate (OUT) .	119
Figure 32.14 : Visualisation de la fiabilité selon les FPS (IN) .....	120
Figure 32.15 : Visualisation de la fiabilité selon les FPS (OUT) .....	120

## XII. ANNEXES

### 1 Planning



## 2 Résultats des tests

---

### Ground Truth :

- **OUT** : 140 personnes (100%)
- **IN** : 172 personnes (100%)

### 2.1 Fiabilité point de vue de face

SSD MobileNet V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	40	110	43	30.7%	67	39.0%
	5	40	75	49	35.0%	71	41.3%
	2	40	36	53	37.9%	82	47.7%
	1	40	18	58	41.4%	85	49.4%
SSD Inception V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	60	90	102	72.9%	138	80.2%
	5	60	55	111	79.3%	144	83.7%
	2	60	26	118	84.3%	143	83.1%
	1	60	13	125	89.3%	145	84.3%
YOLO V3	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	300	27	123	87.9%	160	93.0%
	5	300	14	125	89.3%	166	96.5%
	2	300	6	128	91.4%	171	99.4%
	1	300	3	129	92.1%	168	97.7%
YOLO V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	120	55	75	53.6%	107	62.2%
	5	120	32	78	55.7%	115	66.9%
	2	120	14	80	57.1%	123	71.5%
	1	120	7	92	65.7%	119	69.2%
YOLO V3 Tiny	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	30	120	4	2.9%	5	2.9%
	5	30	85	5	3.6%	8	4.7%
	2	30	40	5	3.6%	11	6.4%
	1	30	20	6	4.3%	12	7.0%

### 2.2 Fiabilité point de vue de côté

SSD Inception V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	60	90	18	12.7%	9	5.2%
	5	60	55	19	13.4%	15	8.7%
	2	60	26	29	20.4%	22	12.8%
	1	60	13	30	21.1%	26	15.1%
YOLO V3	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	300	27	103	72.5%	87	50.6%
	5	300	14	118	83.1%	99	57.6%
	2	300	6	119	83.8%	110	64.0%
	1	300	3	120	84.5%	117	68.0%
YOLO V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	120	55	7	4.9%	1	0.6%
	5	120	32	12	8.5%	1	0.6%
	2	120	14	18	12.7%	2	1.2%
	1	120	7	19	13.4%	3	1.7%

## 2.3 Fiabilité point de vue de côté (redressé)

SSD Inception V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	60	0	86	60.6%	75	43.6%
	5	60	0	104	73.2%	77	44.8%
	2	60	0	113	79.6%	89	51.7%
YOLO V3	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	300	0	134	94.4%	153	89.0%
	5	300	0	138	97.2%	156	90.7%
	2	300	0	136	95.8%	159	92.4%
YOLO V2	Skip Frame	INF (ms)	FPS	OUT	OUT (%)	IN	IN (%)
	10	120	0	55	38.7%	44	25.6%
	5	120	0	68	47.9%	34	19.8%
	2	120	0	70	49.3%	36	20.9%

## 2.4 Fiabilité en fonction des FPS

SSD Inception V2	GT OUT	GT IN	FPS	INF (ms)	OUT	OUT (%)	IN	IN (%)
TOP	140	172	30	60	102	72.9%	138	80.2%
Detection Rate = 10	140	172	15	60	96	68.6%	129	75.0%
	140	172	10	60	89	63.6%	111	64.5%
	140	172	5	60	59	42.1%	76	44.2%
	140	172	2	60	8	5.7%	1	0.6%
SSD Inception V2	GT OUT	GT IN	FPS	INF (ms)	OUT	OUT (%)	IN	IN (%)
TOP	140	172	30	60	111	79.3%	144	83.7%
Detection Rate = 5	140	172	15	60	111	79.3%	135	78.5%
	140	172	10	60	103	73.6%	122	70.9%
	140	172	5	60	84	60.0%	97	56.4%
	140	172	2	60	14	10.0%	5	2.9%
SSD Inception V2	GT OUT	GT IN	FPS	INF (ms)	OUT	OUT (%)	IN	IN (%)
TOP	140	172	30	60	118	84.3%	143	83.1%
Detection Rate = 2	140	172	15	60	117	83.6%	137	79.7%
	140	172	10	60	109	77.9%	126	73.3%
	140	172	5	60	93	66.4%	101	58.7%
	140	172	2	60	33	23.6%	35	20.3%