Rapport de stage de fin d'année

1ère année de Master

Université Paris Nanterre

à GFI Informatique

Étude de caméras et d'algorithmes de tracking pour la Computer Vision sur OpenCV

Étudiant :
Ariel NATAF

Maître de stage :
Jean-Paul MULLER

avril-juillet 2018

Table des matières

Introduction

Le stage a eu lieu au sein de l'entreprise GFI dans le service de la "Forge" au **157 Boulevard Macdonald à Paris**. Il s'est déroulé **du 9 avril au 31 juillet 2018**.

Un projet de Computer Vision a été mis en place au sein de la forge pour répondre à la demande d'un client (une commune) qui souhaitait être capable d'**analyser la circulation à l'aide de caméra de surveillances**.

Au sein de la forge, le besoin de tester des caméras et de comparer des algorithmes s'est exprimé. J'ai consacré les premiers mois de mon stage à chercher à apporter des réponses à ces problèmes.

Le rapport présentera d'abord l'entreprise et le service dans lequel j'ai travaillé, puis va tourner autour des problèmes rencontrés et des solutions présentées avant de conclure avec un bilan.

1 GFI

GFI est une ESN et le stage a eu lieu dans les locaux de la **Forge**.

1.1 L'entreprise

Fondée en 1995, GFI est une Entreprise de Service Numérique (*ESN*) dont le siège se trouve à Saint-Ouen en France. Elle compte aujourd'hui 16 000 collaborateurs dans une vingtaine de pays (Suisse, Belgique, Espagne, Brésil...). Son chiffre d'affaires s'élève à 1 132 M€ en 2017 et 74,5 % de son activité se situe en France.

> *Notre stratégie : Être reconnu comme le 1er acteur régional des services et solutions à valeur ajoutée.* Vincent Rouaix - Président-Directeur Général

Pour parvenir à cela GFI a trois leviers de croissances :

- La **proximité**, avec des implantations régionales qui ont vocation à être souple et réactives pour être à proximité des clients.
- L'**industrialisation**, avec à la clef des gains de productivité, une réduction du *time-to-market* et des coûts optimisés.

- L'**innovation** pour apporter des solutions innovantes.

Afin de concrétiser le levier de l'innovation, GFI a notamment lancé un *Fablab* dans les locaux du **Cargo** dans les 19e arrondissement de Paris.

1.2 La Forge

En octobre 2016, GFI a lancé son Fablab dans les locaux de l'incubateur *Le Cargo* à Paris. Ce Fablab vient renforcer la phase amont du processus d'**idea generation**. Au travers d'ateliers thématiques, il a vocation de prototyper les cas d'usage des clients. Pour prolonger cette initiative, GFI a lancé *La Forge*.

La Forge a pour vocation d'accompagner les entreprises dans l'industrialisation du prototype. Ses experts conseillent les entreprises en matière de financement, de sourcing matériel ou encore de maintenance de l'innovation. En amont, ils ont aussi un rôle de veille technologique et organisent des journées d'acculturation aux « *buzzwords* » (machine learning, chatbot...) : ce que ces nouvelles techniques permettent ou ne permettent pas, en donnant les clefs de compréhensions aux clients pour qu'ils se projettent...

1.3 Organisation

La Forge gère plusieurs **missions en parallèle**, elles peuvent concerner la *Computer Vision*, mais aussi des *chatbots*, des questions de *blockchain*. L'équipe au complet compte près d'une vingtaine de personne qui se répartissent ces différentes missions.

L'équipe est **composée** d'employés de GFI à plein temps, d'alternants et de stagiaires.

Missions pour contribuer aux travaux de Computer Vision

Les missions que j'ai reçu à l'occasion de mon stage ont eu pour objectif d'aider à identifier différents éléments afin d'aider à choisir parmi ces éléments ceux qui sont les plus adaptés au besoin dans le futur. Ainsi dans le cadre de la computer Vision ceci s'est concrétisé comme :

1. Le test de **connexion** de caméra,
2. Une réflexion sur comment tester leur **connexion**.
3. L'utilisation d'une librairie de *Computer Vision*, **OpenCV et à en comparer différentes fonctions**.

2 Connexion des caméras IP

L'objectif de cette mission a été d'étudier différentes caméras IP misent à ma disposition, quels sont leurs fonctionnalités, et établir un rapport qui servira à guider l'achat de nouvelles caméras.

Pour cette mission, la réflexion tournera d'abord autour de l'**état de l'art**, puis sur les **difficultés rencontrées** et enfin sur des **améliorations possibles** à l'étude de connexion des caméras IP.

2.1 État de l'art

L'état de l'art se décompose ici en l'étude des **caméras IP** et des **solutions existantes** pour la connexion de ces caméras

2.1.1 Les caméras IP

Une *caméra IP*, est une **caméra numérique** généralement utilisée pour la **surveillance**. Contrairement à des caméras dites CCTV (*Closed-circuit television*), la caméra IP peut envoyer et recevoir des données via un **réseau informatique et internet**. Bien que la plupart des caméras pouvant faire cela sont des webcams, le terme *caméra IP* est généralement utilisée dans le cadre de la surveillance qui peut être directement accédé par une connexion réseau.

Une caméra IP peut être câblée avec du *RJ45* vers un *routeur* ou « box ADSL », ce qui lui permet à la fois d'être alimentée et les images visionnées sur le réseau, ou alors par Wi-Fi (une alimentation en courant électrique devient alors nécessaire). **Contrairement aux Webcams**, la compatibilité avec les logiciels de visioconférence n'est pas toujours garantie.

2.1.2 Solutions existantes

La connexion aux caméras IP n'est pas un monde vierge de toute fondation, on y trouve des **interfaces**, des **flux *RSTP***, et une **norme *Onvif***

2.1.2.1 Les interfaces constructeurs

Les caméras IP sont en règle générale accompagnées dans leur partie logicielle d'une interface utilisateur. Ces interfaces permettent de configurer les caméras et éventuellement d'accéder au flux vidéo. L'approche de conception de ces interfaces varie grandement en fonction du public visé et de l'utilisation préconisée.

Voici quelques exemples d'interface rencontrée au cours du stage :

- **D-Link**, une interface complète, localisée en français, conçue pour être facilement configurable par des professionnels et des amateurs, bien qu'elle nécessite la connaissance de son adresse IP pour y accéder.
- **XIAOMI, Xiao Fang (Fang Hacks)**, XIAOMI n'étant pas encore implanté en France quand il a fallu essayer ce modèle (avant le 22 mai 2018 [2]), son service de Cloud n'était *a fortiori* pas non plus accessible sur le territoire français. Il a fallu installer nouveau système d'exploitation sur la caméra, le *« Fang Hacks »* [17]. Cette interface est très rudimentaire et ne contient qu'une interface destinée à gérer la connexion. Elle s'adresse surtout à un public de technophiles ayant envie d'utiliser cette caméra premier prix.
- **VR CAM**, marque d'origine chinoise, cible un très grand public et permet de connecter et d'utiliser leurs caméras via leur application mobile [5]. Une utilisation via une autre médium n'est pas prévue par le constructeur. L'interface de l'application a visiblement été localisée en utilisant un traducteur automatique à partir du chinois.

Figure *1* – *D-Link*

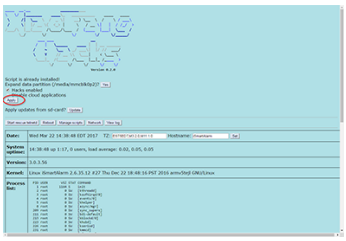

Figure *2* – *Fang Hacks*

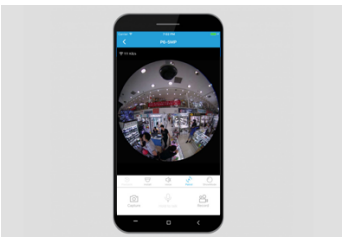

Figure *3* – *VRCAM*

2.1.2.2 Le RTSP

RTSP ou *Real Time Streaming Protocol* (protocole de streaming temps-réel) est un protocole de communication de niveau applicatif(niveau 7 du modèle *OSI*)

destiné aux systèmes de streaming média. Il permet de **contrôler un serveur de média** à distance, offrant des fonctionnalités typiques d'un lecteur vidéo telles que « lecture » et « pause », et permettant un accès en fonction de la position temporelle. [19]

La **transmission des données** de streaming elle-même n'est pas la tâche du protocole *RTSP*. La plupart des servers *RTSP* utilise le *RTP* (*Real-time Transport Protocol*) en conjonction avec *RTCP* (*with Real-time Control Protocol*) pour la distribution du stream média. [18]

Le *RTSP* permet le **multicast** pour la diffusion simultanée à plusieurs clients. Une variété de formats vidéos et audio sont supportés tel que *.264*, *MPEG4*, *MJPEG*, *AAC*, *WMV* etc.

Son utilisation peut se faire avec le logiciel de lecture vidéo *VLC*, ou par un programme qui utilise la librairie **OpenCV** par exemple.

2.1.2.3 La norme Onvif

Le protocole *ONVIF* [11] est l'acronyme du **O**pen **N**etwork **V**ideo **I**nterface **F**orum. Le principe remonte au début des années 2010, où les grandes marques du domaine telles que *Sony* ou *Bosch* ont proposé de mettre en place un forum industriel qui vise à développer une **norme mondiale d'inter-opérabilité** des produits de sécurité IP.

ONVIF travaille aussi avec des groupes de standardisation tel qu'*IEC* (*International Electrotechnical Commission*), *CENELEC* et *ISO* pour leur faire adopter les spécifications ONVIF. Ces spécifications sont des services web, utilisent des standards ouverts dont *XML*, *SOAP* et *WSDL* pour définir la communication entre deux appareils électroniques dans un réseau IP.

Ceci permet aux produits de vidéo-surveillance IP **de pouvoir échanger des informations** entre eux, même s'ils proviennent **de différent fabricants**. Cependant il appartient à ces fabricants d'implémenter ce standard dans leurs produits. Toutes les caméras Onvif n'auront pas la même liste exhaustive de contrôles, ni la même qualité d'implémentation.

La connexion via le protocole Onvif peut se faire en passant par le logiciel open source *ONVIF Device Manager* [12]. Pour la connexion, il suffit de renseigner l'adresse **[IP] :[Port Onvif]** de la caméra concernée en utilisant l'interface du logiciel.

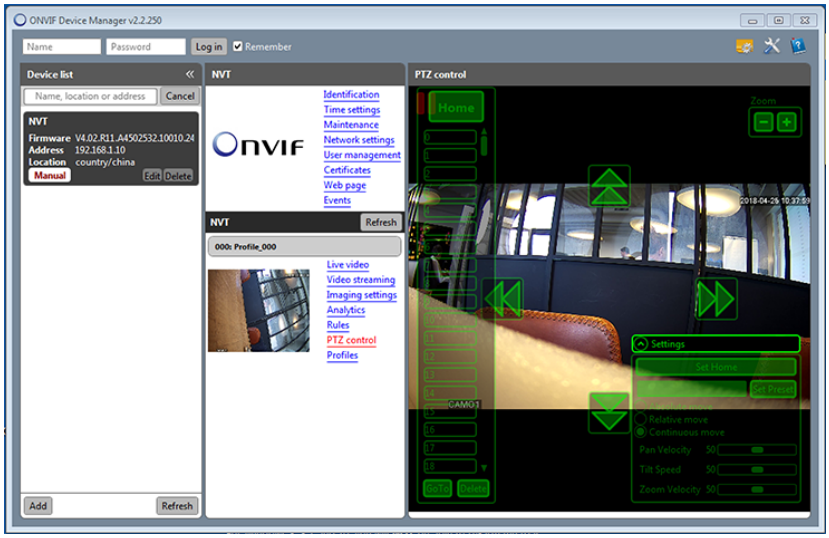

Figure *4 – Interface d'ONVIF Device Manager*

Durant cette mission, des difficultés ont été rencontrées.

2.2 Difficultés rencontrées

Les caméras qui m'ont été demandées d'étudier ne venaient pas toujours avec toutes les clefs. Cela s'illustre par :

1. la documentation parfois lacunaire,
2. des soucis pour les connecter au réseau,
3. une compatibilité logicielle,
4. des formats de flux vidéo.

2.2.1 La documentation

Lors de ma mission, certaines caméras à connecter n'avaient pas forcément une documentation très fournie.

Par exemple une caméra (*Escam Q1039*) avait comme documentation officielle une vidéo sur youtube en russe.

Figure *5 – vidéo explicative en russe publiée sur la chaîne Youtube d'EScam [4]*

Le manque de documentation peut s'avérer problématique. En effet, chaque fabricant dans l'installation de la partie logiciel de la caméra attribut notamment des adresses pour accéder aux flux vidéos pour **RTSP** ou **Onvif** par exemple.

Toutefois, il reste possible de parfois retrouver certaines de ces informations. Par exemple le site *ipsyConnect* met à disposition une liste de caméra avec des adresses pour se connecter en *RTSP*. [9]

Models	Type	Protocol	Path	Audio	Talk Model	Cookies	Flags
0WL QD100	FFMPEG	rtsp://	/user=admin_password=tlJwpbo6_channel=1_stream=1.sdp	Possible	None		
2mp	FFMPEG	rtsp://	/user=admin_password=tlJwpbo6_channel=1_stream=0.sdp	Possible	None		
300	FFMPEG	rtsp://	/user=admin_password=tlJwpbo6_channel=1_stream=0.sdp	Possible	None		
300	FFMPEG	rtsp://	/11	Possible	None		
500	FFMPEG	rtsp://	/user=admin_password=d2gNs1nj_channel=1_stream=0.sdp	Possible	None		
500	FFMPEG	rtsp://	/user=admin_password=tlJwpbo6_channel=1_stream=0.sdp	Possible	None		

Figure *6 – extrait de la liste de caméra EScam sur ISpyConnect*

2.2.2 La connexion au réseau

Pour configurer une caméra IP une connexion au routeur internet est nécessaire.

L'installation des locaux de la « *Forge* » est faite de telle façon à ce qu'internet soit accessible par wi-fi uniquement. L'accès au routeur n'est pas possible. La

connexion a donc dû se faire sans passer par le réseau internet. Pour établir une connexion, la connexion s'est établie en pair à pair avec deux nœuds en branchant directement un câble *RJ45* à l'ordinateur et à la caméra IP.

Cette configuration réseaux particulière a permis d'explorer les fonctionnalités des caméras, mais sans avoir accès un internet ce qui a pu poser problème lors de la recherche d'informations au cours de l'utilisation.

2.2.3 La compatibilité logicielle

Durant le stage j'utilise mon ordinateur personnel. Un Macbook Pro avec **MacOS**. Cependant certains logiciels ne sont pas compatible avec ce système d'exploitation. Un appareil a même son interface web qui utilise **ActiveX**. ActiveX n'est compatible qu'avec **Internet Explorer** qui n'est pas compatible avec MacOs.

Pour contourner ce problème j'ai dû installer une **Machine Virtuelle Windows** pour lancer Internet Explorer et y utiliser Active X.

Figure *7 – Environnement de travail ad hoc*

2.2.4 Le format de flux vidéo

Chaque fabricant ayant le droit de faire comme bon lui semble, certaines caméras peuvent avoir des formats de flux vidéos qui ne sont pas exploitables. En effet, il est arrivé qu'une caméra ne retourne pas un flux vidéo, mais une image au format *.jpeg* en se connectant avec l'adresse **RTSP**. Cette méthode a réputation d'avoir des images de meilleures qualité au prix d'une vidéo plus saccadée. Avec une vidéo mois fluide, l'étude du déplacement d'un objet est plus compliquée.

Il a été décidé d'écarter les caméras qui renvoient un flux d'image qui n'est pas au format vidéo.

Des **améliorations sont possibles**.

2.3 Améliorations possibles

L'étude des caméras c'est fait à partir d'un nombre réduit de caméras (quatre). Pousser l'étude consisterai à **augmenter le nombre de caméras**. Ainsi de nouveaux problèmes vont se dévoiler et alors de nouveaux critères vont émerger pour **affiner le guide pour l'achat de caméras**.

Un autre aspect serait de pouvoir **connecter les caméras à un router et y accéder à distance**. En raison des limitations techniques cet aspect à été mis entre parenthèse, mais il reste néanmoins un aspect fondamental des caméras IP.

3 Établissement d'un protocole de test de caméras

La deuxième partie de ma mission d'étude de caméra est d'effectuer un travail de veille. Il sert à établir un protocole de test de ces caméras. Ces tests ont pour vocation d'estimer la qualité des enregistrements vidéos et d'en reconnaître les plus adaptés au besoin de la *Computer Vision*.

La présentation de cette mission commence par un **état de l'art**, suivie d'une de ma **contribution** qui sera ensuite **évaluer** et des **perspectives** possibles.

3.1 État de l'art

Les tests de qualité des caméras est essentiellement basée sur l'évaluation de la netteté de l'image. Les sociétés qui effectuent les tests de caméras ont leurs propres méthodologies, mais ont tendance à s'inspirer de la **norme ISO-12233** [8]. Cette norme présente notamment un panneau qui sert à évaluer la netteté de l'image. Cette norme est mise à jour environ tout les quatre ans. La dernière version en date est celle de 2017.

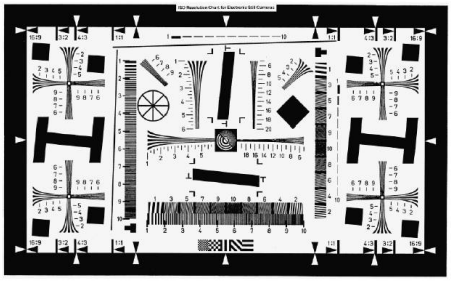

Figure *8 – ISO12233-2000*

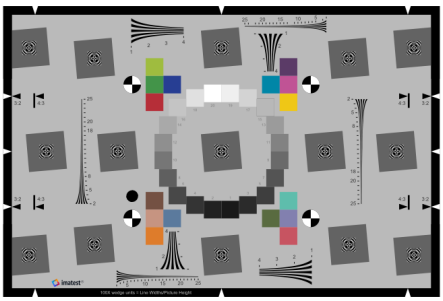

Figure *9 – ISO12233-2017*

La netteté est analysée à partir de zébrures de plus en plus resserrées, plus l'image sera nette, plus chacune de ses rayures seront distinctes. L'analyse se fait avec des algorithmes dit de **MTF** (pour **M**odulation **T**ransfer **F**unction) [20]. Plusieurs programmes payants permettent ces analyses (*IMATEST*, *QuickMTF*), il est également possible d'utiliser un script python avec des librairies gratuites. Les résultats obtenus vont toutefois varier en fonction de l'implémentation de l'algorithme.

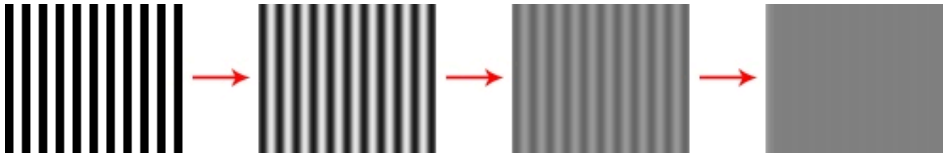

Figure *10 – Effet de perte de netteté sur les zébrures*

Les panneaux de la norme **ISO-12233** disposent des rayures de façon à prendre en compte les courbures de la lentille de l'appareil. L'image ayant tendance à être plus nette au centre que sur les bords.

Pour avoir une meilleure luminosité de la façon la plus consistante possible, les images sont prises en intérieur, avec des lumières neutres et en utilisant des panneaux servant filtrer la lumière pour la rendre plus diffuse et éviter les reflets.

La capacité d'une caméra à produire une image nette varie en fonction de plusieurs facteurs comme la **luminosité**, la **distance focale** ou le **mouvement**.

3.2 Contributions

La finalité de cette mission a été de **présenter un document** qui compile les conclusions du **travail de veille** pour les caméras. Ce document a pris la forme d'une présentation *Microsoft Power Point*.

Une solution naïve pour les tests de caméra dans un environnement **contrôlable et prédicable** est de filmer un écran qui diffuse une vidéo. Cette solution permet en théorie de contrôler simplement toutes les variables comme la luminosité, les éléments de l'image, le mouvement. Une solution idéale en théorie, en pratique elle présente plusieurs **inconvénients**.

Filmer un écran qui diffuse une vidéo ne revient pas au même que de filmer directement la vidéo. L'écran est plat, projette de la lumière et en reflète. Ceci a pour effet de dégrader fortement la qualité de l'image. En conséquence l'image que filme la caméra n'est plus de qualité suffisante pour tester la qualité de l'image en situation réelle.

Figure *11 – Écarter explicitement cette solution et pourquoi*

La solution retenue utilise donc un support **analogique** pour une meilleure fidélité d'image.

L'installation voulue serait une installation qui correspond à l'état de l'art (lumière neutre, panneau diffuseur de lumière, trépied pour la caméra) et sans un logiciel particulier pour évaluer la netteté. La luminosité ayant un effet important sur la qualité d'image, l'utilisation d'un **photomètre**, appareil servant à mesurer les niveaux de luminosité, est recommandé pour s'assurer des bonnes conditions de tests.

Les tests principaux se font en utilisant un **panneau ISO-12233** compatible.

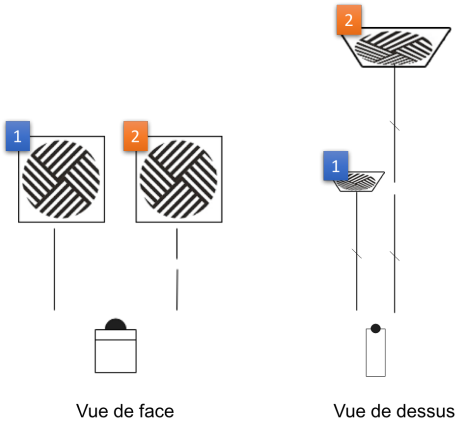

Figure *12 – schéma pour l'installation des tests de focal*

Les vidéos devant tenir compte de la **profondeur de champ**, il est intéressant de chercher à s'assurer que les caméras retenues puissent offrir une bonne qualité d'image à différent niveau de profondeur. L'idée pour ce test est de prendre deux panneaux, de tailles différentes placés à une distance calculée telle que les deux panneaux apparaissent de même taille. On analyse la netteté de l'image avec les deux panneaux. **Si la caméra est capable de bien gérer bien les profondeurs d'image, les deux panneaux seront nets**. Sinon, un panneau sera plus flou que l'autre.

Pour l'analyse de la **qualité d'image lors du mouvement**, un autocollant posé sur un démonstrateur capable de le déplacer selon un trajet défini. On analyse la netteté de l'image lors du déplacement sur plusieurs directions.

Figure *13 – exemple de grille de couleur*

Un autre critère qui n'est pas lié à la netteté de l'image est la **fidélité des couleurs**. Certains algorithmes de Computer Vision passent par une sélection de couleur. Pour tester la fidélité, on capture l'image d'un panneau avec une grille de couleurs avec des valeurs cibles connues.

3.3 Évaluation et perspectives

La démarche de protocole de test ne **s'est jamais concrétisée**. Les moyens à mettre en place n'étaient pas justifiés par le besoin estimé de réaliser ces tests. Sans réaliser ces tests il est **difficile d'estimer la pertinence** de certain d'entre eux.

Par exemple tester **à différent niveau de luminosité** lors de chaque test. Avec trois niveaux de luminosité (jour, pluie et nuit), le nombre de tests se multiplie donc par trois. Le temps de réaliser ces tests augmente d'autant plus qu'il faut compter le temps de changer le niveau et s'assurer que la luminosité est correcte en la mesurant à chaque fois. On peut se demander si la différence que provoque les écarts de luminosité diffère significativement entre un test de focale, de mouvement ou juste pour un panneau simple.

Le démonstrateur qui déplace la zébrure pour le test **en mouvement** n'est pas très rapide. J'avais réfléchi à un système avec différents cylindres superposés avec des vitesses de rotations différentes. Le souci c'est qu'il aurait fallu une imprimante 3D pour produire les pièces (cylindres, rouages...), un logiciel de modélisation 3D pour créer les fichiers à imprimer, se former en électronique et en mécanique, acheter du matériel électronique (*arduino*, *breadboard*...). Dans la mesure où avec le démonstrateur les tests n'étaient pas une priorité, mettre en place ce test aurait demandé un effort bien trop important pour cette mission.

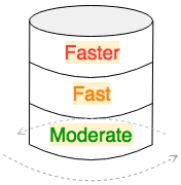

Figure *14 – schéma rudimentaire du projet de test de netteté en mouvement*

Ce qui a été accompli a servi à constituer un document avec des observations et des recommandations qui seront peut-être réutilisés, quand le besoin de tester des caméras se présentera pour l'équipe.

La poursuite du développement de ces protocoles de tests ne semble pas être une priorité pour le moment.

Tout mon stage n'a pas été dédié uniquement à étudier des appareils caméras. J'ai également eu la charge d'établir un benchmark d'algorithmes de suivi fournis avec OpenCV.

4 Benchmark d'algorithmes avec Open CV

Durant mon stage, j'ai eu l'opportunité d'utiliser l'API **OpenCV** pour Python. Cette partie présente d'abord OpenCV, puis la mission qui m'a été confiée de réfléchir à des tests pour comparer des algorithmes de Computer Vision.

4.1 Présentation d'Opencv

OpenCV [13] (Open Source Computer Vision Library) est une librairie open source de *Computer Vision* et de *Machine Learning*. Avec une licence BSD, *OpenCV* est simple à utiliser et à modifier pour les business.

La librairie comporte plus de 2500 algorithmes, allant des algorithmes classiques de *Machine Learning* et de *Computer Vision* aux plus modernes. Ces algorithmes permettent notamment de détecter et de reconnaître des objets, tracer les mouvements des objets, ou encore du traitement d'image.

OpenCV a des interfaces *C++, Python, Java* ainsi que *MATLAB* et il supporte *Windows, Linux, Android* et *Mac OS*. *OpenCV* continue d'étendre ses compatibilités. Par exemple, une interface pour être utilisé par les processeurs graphiques Nvidia utilisant *CUDA* est actuellement en cours de développement. La version d'*OpenCV* utilisée durant ce stage est la version *Python* 3.4.1.

4.2 Objectif : tests d'algorithmes de tracking

Cette mission consiste à établir des tests pour comparer différents algorithmes. Ceci passe par la détermination de critères à évaluer et de la mise en place d'une solution pour les évaluer.

4.2.1 État de l'art

4.2.1.1 Le tracking d'objet

Le tracking d'objet lui même est la tâche de suivre un ou plusieurs objets dans une scène, de la première apparition à la sortie. Un objet peut être n'importe quel objet qui peut être détecter dans une séquence d'image [7]. Il peut s'agir d'une voiture qui traverse une intersection. En général, dans un environnement dynamique l'objet et le fond sont susceptibles de changer. La voiture par exemple peut tourner et ainsi changer d'angle de vue et de taille avec la perspective. Le fond avec l'évolution de la luminosité au fil de la journée ou un mouvement de caméra.

En principe la résolution de ces problèmes est complexe. Il est nécessaire de mettre en place une série de contraintes pour qu'il puisse devenir résoluble. Parmi les contraintes qui peuvent aider avec les problématiques de la Computer Vision on compte :

- Une caméra fixe
- Un nombre d'objet connu
- La taille des objets connue
- Une obstruction des objets limitée
- Un déplacement des objets fluide
- Pas de changement brutal du fond ou des objets

Une fois ces contraintes établies pour générer les vidéos qui serviront de base de travail, des *API* avec leurs lots de solutions existent pour répondre aux besoins en *Computer Vision*. On retiendra principalement celles d'**Opencv**, tout en s'intéresserant aussi à d'**autres solutions généralistes**.

Dans la computer Vision, plusieurs types d'algorithmes sont utilisés. Des algorithmes de **détection**, de **reconnaissance** et **tracking** [6].

4.2.1.2 Les solutions fournies par OpenCV

OpenCV est une librairie qui fournit des API de détection, de reconnaissance et de tracking.

4.2.1.2.1 API de détection

OpenCV permet d'utiliser des algorithmes pour détecter un objet sur une image. Un algorithme de détection permet de détecter automatiquement un

objet. Cette reconnaissance peut se faire en se focalisant sur les objets en mouvement. Toutefois, cette méthode peut être confrontée à des limitations quand il s'agit d'observer un objet immobile ou obstrué par un objet au premier plan.

Sur OpenCV on retrouve une variété d'algorithmes via son API, dont **HAAR Cascade** [15] qui repose sur du Machine Learning.

Cette méthode peut être utilisée pour suivre un objet image après image, mais risque de le perdre quand il se retrouve à l'arrêt.

4.2.1.2.2 API de reconnaissance

Une fois un objet détecté, il peut être utile de le reconnaître. Dans ce cas, il est nécessaire d'associer les points d'intérêts de l'objet (*features*) à ceux d'une galerie d'objet que l'algorithme a été entraîné à reconnaître.

Les objets que l'on a besoin de reconnaître sont très liés au contexte, les algorithmes de détection sont souvent spécialisés. Par exemple, depuis Open CV 2.4, une API de reconnaissance de visage est inclue [3].

4.2.1.2.3 API de tracking

Les algorithmes de tracking permettent de suivre un objet. l'algorithme va comparer l'évolution entre plusieurs images successives dans une vidéo. Ces algorithmes permettent de suivre un objet, qu'il se déplace ou non. À la différence d'un algorithme de détection, ils ne permettent pas de repérer automatiquement un objet si on ne le lui pas indiqué initialement. Certains algorithmes sont plus fluides que d'autres, gèrent mieux le changement d'angle de la prise de vue ou de tailles d'un objet, tiennent compte de la perte d'un objet qu'il est supposé suivre.

OpenCV intègre dans sa version 3.4.1 plusieurs fonctions de tracking prête à l'emploi. La plus ancienne **BOOSTING** a une dizaine d'années alors que les plus récentes comme **CSRT** ont été ajoutées dans la version la plus récente de l'API. [14]

4.2.1.3 Autres solutions généralistes existantes

De nombreuses solutions pour le tracking d'objets existent actuellement. Certaines de ces solutions sont apportées par des API de Computer Vision prêtes-

à-l'emploi provenant de grands groupes comme *Google*, *Microsoft* et *IBM* [1]. Cependant ces outils universels ne sont pas toujours utile. Une solution plus spécialisée à un problème donné peut être plus efficace pour sa résolution.

Figure *15* – *Google Tensorflow*

Figure *16* – *MS Cognitive Service*

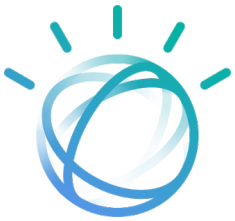

Figure *17* – *Watson d'IBM*

Aujourd'hui les algorithmes dans des compétitions de reconnaissance d'objet ne sont toujours pas capable d'avoir des résultats parfaits dans le tracking d'objets multiples [10].

4.2.2 Contribution sur OpenCV

4.2.2.1 Pertinence et difficulté

Ma mission concerne l'analyse des algorithmes de fonctions de **suivie** (ou **tracking**) d'OpenCV.

En effet, les algorithmes concernés serviront à analyser le flux de la circulation routière et piétonne filmé par des caméras de sécurité. Les éléments à tester sont :

- La **reconnaissance** des objets en premier plan.
- L'**identification** des objets en premier plan.
- Le **parcours** des objets
- La **vitesse d'exécution** en fonction du nombre d'objets et de la résolution de l'image.

Les solutions retenues pour chacun de ces points doit tenir en compte des particularités de ce contexte. Parmi ces particularités on peut retenir que l'image est fixe, la qualité risque de varié en fonction de la météo, les zones piétonnes et routières ne varient pas, les objets en mouvement sont des véhicules de tout types et des piétons.

4.2.2.2 Résolution

La résolution des difficultés passe par la mise en place matérielle et logicielle du test, et par l'étude des critères à analyser.

4.2.2.2.1 La mise en place

Ces tests pour avoir des résultats comparables devront être réalisés sur du matériel avec une configuration connue afin d'éviter les écarts de performance liés au matériel alors que l'on souhaite évaluer les performances du logiciel.

Afin d'avoir des tests reproductibles et que les données attendues soient prévisibles, le flux vidéo qu'analyseront ces algorithmes de *Computer Vision* sera constitué de vidéos enregistrés et correspondant à différentes situations (de circulation ou météorologiques par exemple) pour couvrir au maximum l'efficacité des algorithmes.

En vue d'établir un benchmark qui répond au mieux aux besoins en l'espèce, deux angles ont été retenus, la précision de l'identification et la vitesse d'exécution.

4.2.2.2.2 La précision

L'algorithme a pour mission d'identifier et de cataloguer les objets filmés et leurs déplacements dans le cadre.

Ces résultats doivent être catalogués dans un log. On attend du log de contenir les objets en mouvement identifiés, les objets reconnus et leurs parcours au travers de différentes zones. Ce log sera analysé et comparé au résultat attendu.

Dans le cadre de la **reconnaissance** d'un objet en mouvement on peut rencontrer plusieurs cas. Un vrai positif, un objet à détecter a été détecté. Un faux négatif, un objet à détecter n'a pas été détecté. Un faux positif, un objet a été détecté alors qu'il n'y avait pas d'objet à détecter. L'analyse du log doit en tenir compte afin de ne pas favoriser un algorithme trop sensible qui détectera des objets là où rien n'est à analyser.

Pour l'**identification** d'un objet, soit un objet est correctement identifié soit il ne l'est pas. La situation est la même pour le **traçage de parcours des objets**. Ce traçage se fait en enregistrant l'entrée et la sortie dans des zones définies pour la caméra (trottoir, passage piété, voie à sens unique...).

L'analyse du log se conclut par l'établissement d'un score qui augmente en fonction des résultats identifiés comme vrai positif et diminue en fonction des autres.

4.2.2.2.3 La vitesse d'exécution

Les algorithmes n'ont pas nécessairement la même vitesse d'exécution et plus elle est réduite, plus l'algorithme sera apte à analyser un flux continu.

Le test tiendra simplement en compte la durée d'exécution et établira un score en fonction de la durée de la vidéo comme référence.

4.2.2.2.4 Implémentation

Sélections d'algorithmes de tracking :

Pour tester différents algorithmes de tracking rapidement, une fenêtre de sélection de l'algorithme souhaité a été créée. Cette interface est faite en utilisant le module **tkinter** de *Python*. Dans la liste des algorithmes se trouve pour l'instant les algorithmes inclus dans **OpenCV**. Ceci évite de modifier manuellement le code à chaque fois que je désire tester un algorithme différent.

On va s'intéresser à la partie qui consiste à la sélection d'un algorithme en passant par une interface graphique créée avec tkinker.

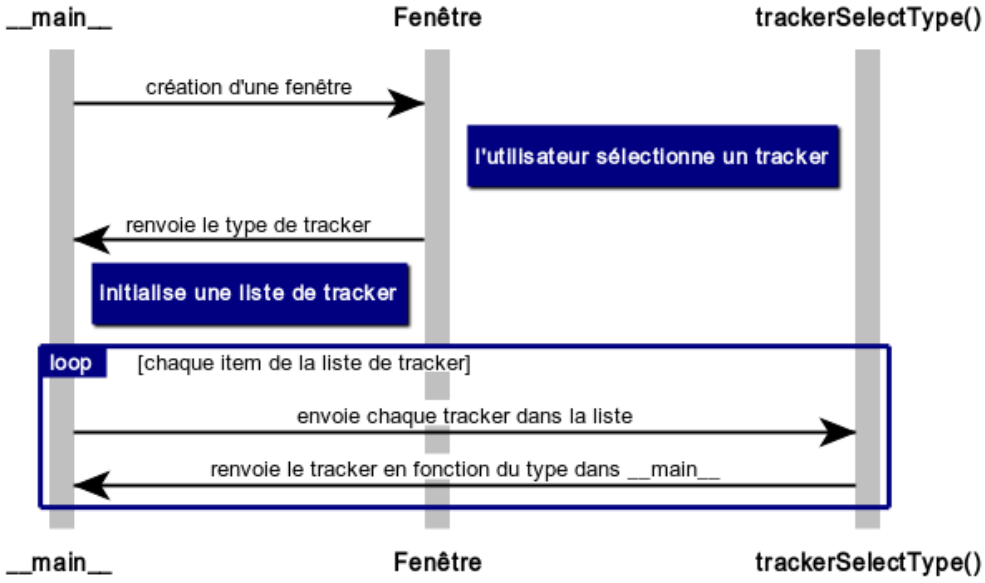

Figure *18 – Sélection et initialisation de l'algorithme de tracking*

La première étape est de créer une liste avec le nom de chaque algorithme.

Mise en place d'openCV sur Python et de la liste des algorithmes

```
import numpy as np
import cv2 as cv
import tkinter as tk
from tkinter.messagebox import showinfo

# Les différents types de trackers inclus dans l'API
    tracker_types = ['BOOSTING', 'MIL', 'KCF', 'TLD',
    ↪  'MEDIANFLOW', 'GOTURN', 'MOSSE', 'CSRT']
```

Cette liste va servir à créer une fenêtre avec une liste avec un item à sélectionner

Fenêtre de sélection de l'algorithme

```
fenetre = tk.Tk()
liste = tk.Listbox(fenetre)
for track in tracker_types:
    liste.insert(tracker_types.index(track), track)
liste.pack()
bouton = tk.Button(fenetre,
          text = "Select",
```

```
          command = fenetre.quit)
bouton.pack()
fenetre.mainloop()

# Algorithme choisi
tracker_type = tracker_types[liste.curselection()[0]]
```

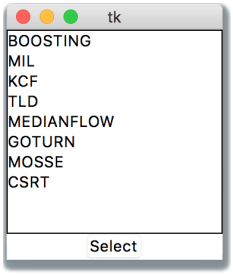

Figure *19 – boite de dialogue pour choisir le tracker*

On initialise une liste de trackers. Cette liste aura la taille du nombre de *Bounding Box* créées. L'algorithme boosting est sélectionné par défaut afin de créer une liste de trackers, il va être écrasé par un autre tracker.

```
_____ Initialisation s'adaptant au nombre de bounding box _____
    bbox = [(290, 205, 40, 90),
            (605, 230, 40, 90),
            (700, 250, 40, 90)]

for i in range (0,len(bbox)):
        if i == 0:
            trackerselect = [cv.TrackerBoosting_create()]
        else:
            trackerselect.append(cv.TrackerBoosting_create())
    for i in range(0,len(bbox)):
        trackerselect[i] =
        ↪ trackerSelectType(trackerselect[i])
```

L'item sélectionné va servir à choisir la fonction pour initialiser un nouveau tracker qui va être inclus dans la liste des trackers. Il écrase le tracker par défaut. Il est nécessaire de repasser par cette fonction à chaque initialisation, autrement la liste de trackers pointera toujours vers le même tracker.

```
_________ Attribution de l'algorithme à des trackers _________
def trackerSelectType(trackerselect):
```

```
    (major_ver, minor_ver, subminor_ver) =
    ↪ (cv.__version__).split('.')
    if int(minor_ver) < 3:
            trackerselect = cv.Tracker_create(tracker_type)
    else:
        if tracker_type == 'BOOSTING':
            trackerselect = cv.TrackerBoosting_create()
        if tracker_type == 'MIL':
...
    return trackerselect
```

La vidéo et les régions d'intérêts des objets à suivre

Comme on souhaite tester le tracking, les algorithmes sont utilisés dans des situations similaires. Pour cela le programme assure que l'on aura à chaque fois :

- La même vidéo.
- Les mêmes zones d'intérêt sélectionnées.

De légères différences dans la zone sélectionnée pour encadrer l'objet à suivre peuvent changer sensiblement le comportement de l'algorithme. Idéalement cette zone doit contenir au maximum l'objet avec un minimum d'arrière plan. Ceci peut être compliqué quand on analyse des formes non rectangulaires comme un cycliste sur son vélo.

Plusieurs vidéos ont été sélectionnées pour ce test. Une première, *Shibuya* a été choisie car elle permet de voir la différence de comportement sur des objets de différentes tailles (voiture, bus, et arrière d'une voiture) et des obstacles (vue obstruée par le feu tricolore, la foule de piéton).

Les régions d'intérêts sont entrées dans un **tableau de tuples** (collection ordonnée d'attributs relatifs à un même objet) qui contiennent les coordonnées de chaque *bounding box* (sélection rectangulaire de l'objet à suivre). L'utilisation de la bounding box se fait sous forme de boucle dont l'index s'adapte à la taille de se tableau de tuples. Ainsi en ajoutant ou retirant simplement un tuple dans ce tableau, le programme va pouvoir les suivre sans autre modification.

On peut déjà s'apercevoir en observant la vidéo que le comportement de ces *Bounding Box* varie en fonction des algorithmes.

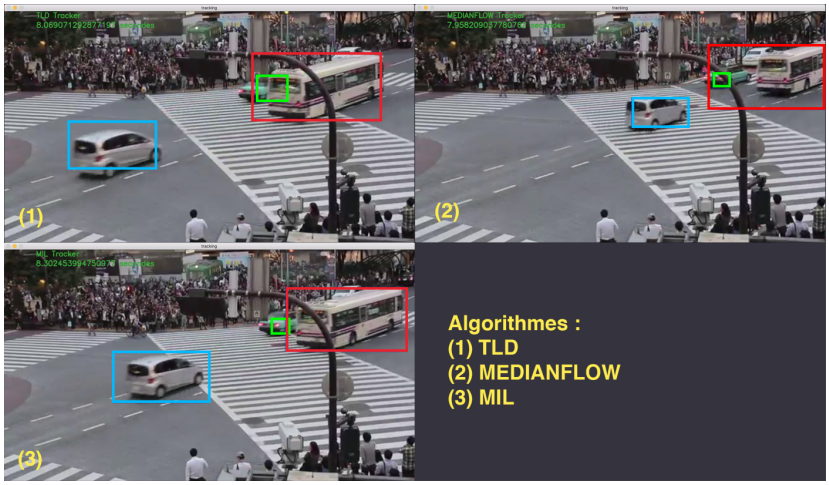

Figure *20 – Différence de résultats entre différents algorithmes à environ 8 secondes d'exécution.*

Ces *Bounding Box* vont avoir des temps d'exécutions plus ou moins long et vont parfois échouer à suivre chaque voiture sur la totalité de leur trajet. Avec ces informations, on peut commencer à l'élaborer un score pour chaque algorithme. L'incapacité à suivre des objets se déplaçant en ligne droite est éliminatoire et est indexé sur le temps d'exécution.

Algorithme	Temps (secondes)	Perte	Score $(3 - Pertes) * (100/3) * (Temps * Durée)$
Boosting	51.208574295043945	2	3,254663286
MIL	54.95484709739685	2	3,032792838
KCF	35.16063404083252	2	4,740149636
TLD	110.28673601150513	3	0
MEDIANFLOW	**24.363162994384766**	**0**	**20,5227868**

Table *1 – Exemple de score pour différents algorithmes sur Shibuya*

On peut alors observer sur cinq algorithmes un écart très important dans les temps d'exécution et la capacité à suivre un objet.

La vidéo *Shibuya* a comme inconvénient un encodage qui conduit à une exécution très lente des algorithmes. La vidéo de **4,6 secondes** peut avoir un temps d'exécution de l'algorithme TLD proche de **2 minutes**. Si cela peut être pénible lorsque l'on souhaite effectuer plusieurs tests successifs, ceci permet toutefois de mettre en exergue les différences de performances.

Lors de la suite de l'élaboration des tests, d'autres vidéos ont été privilégiées.

Zones d'entrées et de sorties

Un autre vidéo présente dans une intersection prise en coopération avec la ville cliente a servi pour les tests. Avec cette vidéo les temps d'exécution des algorithmes était bien plus proche de la durée de la vidéo.

Pour utiliser ces zones il a fallu choisir le critère pour déterminer quand un objet est ou non dans une zone. Pour cela les coordonnées Bounding Box sont exploitées. Plutôt que de considérer un objet comme dans une zone quand elle se superpose à la *Bounding Box*, l'objet est considéré dans la zone quand son centre est dans la zone.

Une fois ce critère choisi il a été possible d'afficher le parcours qu'effectue la *Bounding Box* pour simplifier l'observation humaine. Dans ce même soucis d'observation, un message est affiché quand un objet est dans une zone.

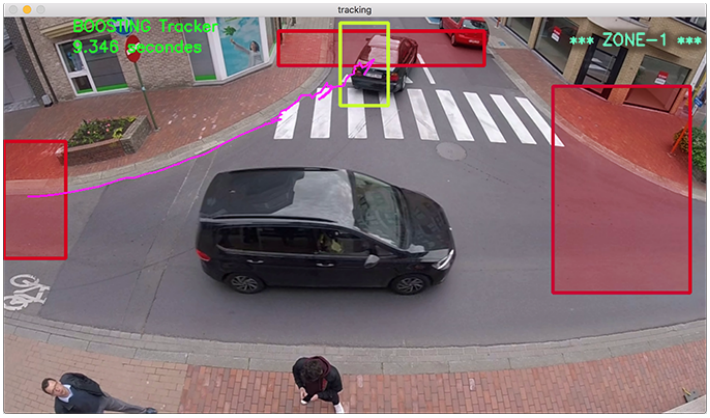

Figure *21 – Objet entrant dans une zone*

Il arrive que l'objet n'arrive pas à entièrement entrer dans la zone mais se superpose, choisir l'ensemble de la surface de la **Bounding Box** est plus favorable à produire des **faux positifs**.

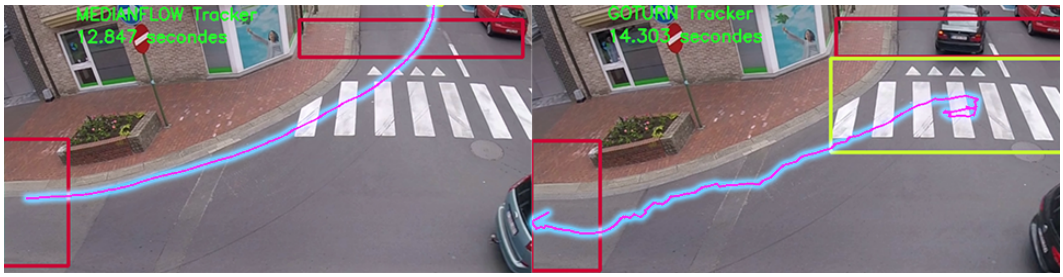

Figure *22 – Différence de trajet entre deux algorithmes en magenta surligné*

Dans le cas de figure de l'algorithme **Goturn**, l'objet était partiellement superposé dans la zone centrale. Choisir un paramètre plus restreint comme le centre a permis d'éviter un **faux positif**.

Création d'un log

Pour la création du log j'ai utilisé la librairie Python **logging** [16]. Cette librairie permet de simplement créer un fichier de log en fonction de l'activité du programme. Ce fichier a vocation à garder une trace de la dite activité et ainsi être utilisé pour comparer des résultats. Il enregistre quel algorithme est utilisé lors de la session, la position toute les trente frames, et quand l'objet entre et sort d'une zone.

Élaboration d'un benchmark

Pour affiner le benchmark par rapport au tableau de score précédent, des critères ont été ajoutés. Ces critères sont surtout qualitatifs et sont alors difficilement quantifiables. Avec la constatation des différences de performances entre les algorithmes, j'ai passé la vitesse d'exécution comme une variable qualitative. Au vu de la variabilité d'un critère comme les pertes en fonction de la vidéo, j'ai préféré en faire une variable qualitative en fonction de la capacité à ne pas perdre et sous le nom de « *Précision* ».

Nom	Vitesse	Adapte taille	repère quand perdu	Fluidité	Précision
Boosting	lent –	non	non	–	–
MIL	len -	non	non	-	-
KCF	lent -	non	oui	+	-
TLD	lent —	oui	non	—	—
Median Flow	lent -	oui	oui	+	+
Goturn	lent –	oui	non	-	–
MOSSE	lent -	non	oui	+	+
CSRT	lent –	oui	non	+	-

Table *2 – benchmark de différents algorithmes de tracking*

Un score n'est plus déduit, mais les différences restent facilement visibles.

4.3 Résultats et améliorations possibles

En l'état, le programme connaît encore plusieurs lacunes.

Il est fait pour lancer les algorithmes fournis par OpenCV. Cece le rend **limité par ces algorithmes implémentés de base**. Les algorithmes développés à part ne seront pas compatibles et demanderont des refontes importantes du programme pour y être utilisable.

Configurer le programme peut s'avérer **laborieux**. Sélectionner les zones

d'intérêt se fait en rentrant des **coordonnées** d'abscisse, d'ordonnée, de longueur et de la largeur à la main. Pour entrer un **temps** à laquelle une voiture est attendu dans une zone, il faut choisir une frame. Le temps en utilisant une horloge ne fonctionne pas car les algorithmes ont des temps d'exécutions différents. Trouver ces informations demande de procéder par tâtonnements et peut demander un nombre important d'essais.

Également le programme n'est doté que d'une **faible automatisation**. Il n'est pas possible avec de programme d'automatiser avec certitude quand un objet entre ou sort d'une zone, car aucun de ces algorithmes de tracking n'est parfaitement fiable.

Parmi les objectifs posés lors de la réflexion autour des tests d'algorithmes de tracking se trouvaient les notions de **faux et vrais positifs**. Lors de la conception du programme je me suis retrouvé confronté à des problèmes pour reconnaître un faux positif. Par exemple, certains algorithmes ne sont pas capables de comprendre quand l'objet est perdu. Certains algorithmes peuvent voir leur zone d'analyse faire des « sauts » quand ils perdent l'objet et se retrouver dans la zone attendue. De même algorithme peut se mettre à suivre un mauvais objet quand il en croise un autre.

Effectuer un **benchmark** des algorithmes n'est pas parfaitement rigoureux. Les différences de vitesses changent en fonction de l'encodage de la vidéo. La capacité d'un algorithme à suivre un objet dépend grandement de la zone d'intérêt sélectionnée au départ. Il est possible qu'une zone d'intérêt favorise plus un algorithme qu'un autre.

L'observation humaine reste essentielle dans ces tests.

5 Bilan des acquis

Ces trois premiers mois de stages ont été l'occasion de travailler dans un environnement de travail qui laisse une grande place à l'**autonomie**.

Ce stage a d'abord été l'occasion d'appliquer mes connaissances acquises en MIAGE, tel que des **notions de réseaux** à la connexion de caméra IP, ou à m'abstraire pour résoudre des problèmes et concevoir des solutions.

Pour les besoins du stage j'ai appris le langage de programmation **Python**. J'ai notamment pu explorer comment fonctionne les rudiments de la syntaxe Python et l'utilisation de librairies comme *tkinter* pour créer une interface graphique sommaire ou *logger* pour des logs.

Ceci a été fait dans le cadre de mission liée à la **Computer Vision**, domaine que j'ai pu découvrir à cette occasion, et mis en application avec **OpenCV**.

Références

[1] Harish Choudhary. *Comparing the Top Computer Vision APIs for OCR*. Sept. 2017. url : https://medium.com/playment/comparing-the-top-computer-vision-apis-for-ocr-32def95f41ac.

[2] Rémy Dessarts. *Les smartphones chinois du géant Xiaomi arrivent en France*. Mai 2018. url : https://www.lejdd.fr/economie/les-smartphones-chinois-du-geant-xiaomi-arrivent-en-france-3639701.

[3] OpenCV 2.4.13.6 documentation. *Face Recognition with OpenCV*. url : https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html.

[4] EScam. *youtube : ESCAM Q1039*. Mai 2017. url : https://youtu.be/uSwwngBvzVg.

[5] *Google Play : VR Camera,VR CAM*. url : https://play.google.com/store/apps/details?id=com.generalcomp.vrcam&hl=fr.

[6] Evelyn Grevelink. *A Closer Look at Object Detection, Recognition and Tracking*. Déc. 2017. url : https://software.intel.com/en-us/articles/a-closer-look-at-object-detection-recognition-and-tracking.

[7] Suresh Merugu PhD de l'Indian Institute of Technology Roorkee. *What are the state of the art methods for tracking many multi objects with static camera videos ?* url : https://www.researchgate.net/post/What_are_the_state_of_the_art_methods_for_tracking_many_multi_objects_with_static_camera_videos.

[8] ISO. *Photographie – Imagerie des prises de vues électroniques – Résolution et réponses en fréquence spatiale*. url : https://www.iso.org/fr/standard/71696.html.

[9] ISpyConnect. *Escam camera list*. url : https://www.ispyconnect.com/man.aspx?n=eScam.

[10] A. Milan et al. "MOT16 : A Benchmark for Multi-Object Tracking". In : *arXiv :1603.00831 [cs]* (mar. 2016). arXiv : 1603.00831. url : http://arxiv.org/abs/1603.00831.

[11] *Onvif*. url : https://www.onvif.org/.

[12] *Onvif Device Manager*. url : https://sourceforge.net/projects/onvifdm/.

[13] OpenCV. *ABOUT*. url : https://opencv.org/about.html.

[14] OpenCV. *documentation openCV sur les fonctions de trackings*. url : https://docs.opencv.org/3.4.1/d9/df8/group__tracking.html.

[15] OpenCV. *Face Detection using Haar Cascades*. url : https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html.

[16] «Sam». *Écrire des logs en Python*. Mar. 2013. url : http://sametmax.com/ecrire-des-logs-en-python/.

[17] samtap. *Fang Hacks*. Sept. 2017. url : https://github.com/samtap/fang-hacks.

[18] iWave Systems. *RTSP Media Streaming Server on Windows Embedded Compact 7(WEC7*. url : http://www.iwavesystems.com/rtsp-media-streaming-server-on-windows-embedded-compact-7-wec7.

[19] Wikipedia. *Real Time Streaming Protocol*. url : https://fr.wikipedia.org/wiki/Real_Time_Streaming_Protocol.

[20] Don Williams. *Benchmarking of the ISO 12233 Slanted-edge Spatial Frequency Response Plug-in*. url : https://pdfs.semanticscholar.org/42c9/1d77d8d2ab9e415557fef88aae57bc6d3ebe.pdf.