

ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC



DANS LE CADRE DU COURS GPA793 - PROJET DE FIN D'ÉTUDES

Rapport technique final

Système de surveillance amovible

PRÉSENTÉ À
T. Wong

PAR

R. LUPIEN
K. CLERCY
M. LE VERGOS

MONTRÉAL, LE 12 AOÛT 2021

Table des matières

1 Définition de la problématique	3
1.1 Énoncé de la problématique	3
1.1.1 Aspects législatifs et réglementation	3
1.1.2 Client	3
1.1.3 Économique	3
1.1.4 Santé et sécurité	3
1.1.5 Contexte culturel et social	4
1.1.6 Développement durable	4
1.2 Objectifs visés et accord de collaboration de l'équipe	4
1.2.1 Objectifs généraux	4
1.3 Gestion de projet	4
1.3.1 Cahier des charges	4
1.3.2 Spécifications	5
1.3.3 Analyse de faisabilité économique et financière	5
1.3.4 Identification et évaluation des risques	6
1.4 Revue de la documentation	6
1.4.1 Modèles et jeux de données	6
1.4.2 Configuration du modèle	8
1.4.3 Traitement d'image	9
1.4.4 Envoi de notifications à l'utilisateur	9
1.4.5 Choix des moteurs	9
1.4.6 Branchements	9
1.4.7 Remise à zéro	9
1.4.8 Librairie AccelStepper	10
1.4.9 Protocole de communication I2C	10
1.5 Ressources	10
1.5.1 Ressources humaines	10
1.5.2 Ressources matérielles	10
1.5.3 Ressources logicielles	10
1.5.4 Méthodologie d'expérimentation	10
2 Processus de conception	11
2.1 Traitement d'image et détection d'objets	11
2.1.1 Personnalisation du modèle YOLOv4-tiny	11
2.1.2 Intégration du système sur le Raspberry Pi	11
2.2 Support de caméra amovible	12
2.2.1 Mise en place du protocole I2C	12
2.2.2 Architecture du système de contrôle	12
2.2.3 Calcul des positions	14
2.2.4 Déplacement du servomoteur	14
2.2.5 Déplacement de la caméra avec un joystick	14
2.2.6 Test de la plateforme	15
3 Analyse des résultats	16
3.0.1 Plateforme physique	16
3.0.2 Métriques de performance du modèle	16
3.0.3 Nombre de paramètres du modèle	19
3.0.4 Rapidité du traitement d'image	20
4 Conclusion et recommandations	21
Références	22

Table des figures

1	Système de caméra amovible	5
2	Comparaison de modèles basée sur COCO	7
3	Arborescence des fichiers darknet	8
4	Montage physique	12
5	Structure de données	12
6	Diagramme de classe	13
7	Calcul de la position	14
8	Courbe de Précision versus Recall	17
9	Courbe de Précision versus Recall avec aire	18
10	Résultats de validation	19
11	Réseau YOLOv4	19
12	Images d'entraînement vs. images réelles	21

Liste des tableaux

1	Contraintes	5
2	Spécifications matérielles	5
3	Évaluation des coûts	6
4	Tableau des résultats de test	15
5	Données de validation	16
6	Précision du modèle	18

Liste des acronymes

AP	Average Precision
COCO	Common Objects in COntext
FN	Faux Négatif
FP	Faux Positif
FPS	Frame Per Second
GPA	Génie de la Production Automatisée
GPU	Graphics Processing Unit
IDE	Integrated Development Environment
IMAP	Internet Message Access Protocol
IoU	Intersection of Union
mAP	mean Average Precision
OID	Open Image Dataset
POP	Post Office Protocol
SD	Secure Digital
SMS	Short Message System
SMTP	Simple Mail Transfer Protocol
SPI	Serial Peripheral Interface
VM	Virtual Machine
VN	Vrai Négatif
VP	Vrai Positif

1 Définition de la problématique

1.1 Énoncé de la problématique

Dans le cadre du cours de GPA793 « Projet de fin d'études », notre équipe a travaillé à la réalisation d'un système de surveillance pour enfant avec support de caméra amovible. Puisque la sécurité est une préoccupation principale pour tous les parents, nous avons décidé de joindre nos connaissances afin de les aider, dans la mesure du possible.

Notre objectif avec ce projet était de réaliser un prototype sans toutefois concentrer une majeure partie de notre temps à la conception, c'est pourquoi nous nous sommes tournés vers un projet d'intégration. Étant tous les trois des étudiants en génie de la production automatisée (GPA), ce projet nous a permis de mettre en pratique des notions faisant appel à nos compétences mécaniques, électriques et logiciel. Outre les aspects techniques, ce projet a également fait appel à notre sens de l'éthique, notre sens des responsabilités ainsi que notre engagement social, toutes des valeurs essentielles à la profession d'ingénieur.

1.1.1 Aspects législatifs et réglementation

Pour ce projet, notre système doit filmer à l'intérieur de la résidence des clients. Pour cette raison, plusieurs lois concernent notre technologie, soit la charte canadienne des droits et libertés, la loi sur l'accès à l'information et la loi sur la protection des renseignements personnels sont toutes des lois canadiennes qui nous concernent [1]. Au Québec, c'est la loi sur la protection des renseignements personnels dans le secteur privé qui s'applique [25]. Cependant les seules images qui sont susceptibles de poser problèmes sont celles que le système envoie à l'utilisateur par courriel lorsqu'un élément dangereux est détecté afin d'indiquer sa position, car autrement les images récoltées ne sont pas enregistrées, elles sont uniquement utilisées par notre système de détection. Donc pour la sécurité des utilisateurs, ces images seront encryptées.

1.1.2 Client

Notre produit s'adresse au grand public. Nos principaux concurrents seront les moniteurs pour bébés numériques. Toutefois, les moniteurs sont par définition destinés aux bébés, alors que notre système peut être utilisé pour les jeunes enfants comme les plus grands qui sont capable de se déplacer. Évidemment, notre technologie offre la possibilité de détecter les objets dangereux, ce que les moniteurs pour bébé ne font pas, mais ces derniers offrent aussi plusieurs fonctionnalités que nous n'avons pas. Toutefois, plusieurs aspects de notre projets pourraient continuer à être développés comme le système de notifications. D'ailleurs, si une application pour téléphone était développée, elle pourrait offrir davantage de fonctionnalités.

1.1.3 Économique

Notre système est encore à l'étape du prototypage et ne peut être commercialisé dans l'état actuel. L'objectif de ce travail demeure une preuve de concept. Toutefois, dans la mesure où le produit serait commercialisé, plusieurs modifications devraient être apportées au montage, car les fils sont apparents et la plateforme se transporte avec délicatesse. De plus, une caméra de meilleure résolution et un système avec une meilleure capacité de calcul sont des améliorations qui pourraient permettre à notre système de meilleures performances. De ce fait, il est difficile pour nous d'estimer le montant réel associé à un tel produit puisque les changements à apporter représentent un investissement supplémentaire. Toutefois, ce risque est minime, car nous savons que le produit est fonctionnel grâce à la preuve de concept.

1.1.4 Santé et sécurité

La santé et la sécurité sont des enjeux principaux de notre travail. En effet, l'outil que nous développons vise à offrir un environnement plus sécuritaire pour les enfants en permettant à leur parents de prévenir le danger. Par exemple, certains objets comme une fourchette sont banals pour un adulte, mais représentent un danger pour l'enfant et il arrive que par distraction ces objets soient laissés sans surveillance, ce genre de situations pourront être évités grâce à notre système. Évidemment, il existe un

risque que notre technologie commette aussi des erreurs, c'est pourquoi notre système sera plus sensible que possible. De ce fait, notre système enverra peut-être de fausses alertes, mais ce scénario demeure préférable.

1.1.5 Contexte culturel et social

Les enjeux liés au contexte culturel sont peu applicables dans le cadre de notre projet. En effet, tel que nous l'avons expliqué dans la section client, le produit s'adresse au grand public. Plus précisément, notre produit peut être intéressant pour les parents monoparentales qui ont constamment besoin de partager leur attention entre les tâches du quotidien et leur enfant. Toutefois, il est certain que ce gadget est plus accessible pour les familles mieux nanties.

1.1.6 Développement durable

Le concept du projet n'a pas été développé dans un objectif de développement durable. Toutefois, un des principes directeurs du développement durable, soit faire plus avec moins a été respecté. Le choix de matériaux c'est d'abord fait dans une optique technologique, mais par conscience écologique et économique nous avons réussi à réutiliser plusieurs éléments. Notamment, les Raspberry Pi, Arduino, fils et composants électriques achetés pour le cours de GPA788 ont été réutilisés. De plus, des moteurs d'imprimantes 3D ainsi qu'un moteur de ventilateur brisé ont été récupérés.

1.2 Objectifs visés et accord de collaboration de l'équipe

1.2.1 Objectifs généraux

Le premier objectif de ce projet concerne le traitement d'image. Plus précisément, le système doit être en mesure d'effectuer un traitement d'image pour isoler l'élément d'intérêt et en faire la reconnaissance. Les sources de danger potentiel sont multiples, mais le problème a été restreint à l'identification de couteaux seulement pour permettre de se concentrer sur d'autres objectifs. Afin que chaque membre de l'équipe puisse avancer individuellement, nous avons séparé les tâches et l'objectif de traitement d'image.

Le second objectif concerne le support de caméra mobile. Ce dernier doit effectuer des mouvements verticaux ainsi qu'horizontaux. Les mouvements se font en mesure des déplacements de l'enfant, ce qui requiert également de la détection. L'objectif est d'utiliser les données renvoyées par la détection pour suivre les déplacements d'une personne et la garder au centre de l'écran. Lorsqu'il n'y a personne, la caméra sera à sa position initiale. Il est important de préciser que le système ne gère pas la présence de plusieurs enfants. Nous sommes conscients qu'il s'agit d'une amélioration potentielle, car cela peut être problématique. Toutefois nos objectifs sont simplifiés considérant les délais. En raison de la distance qui nous sépare, car le montage mécanique et électrique peut difficilement être séparé.

Finalement, le dernier objectif concerne l'interface utilisateur. Celle-ci permet d'envoyer des notifications par message texte lorsqu'un danger est détecté dans la pièce où se trouve l'enfant.

1.3 Gestion de projet

1.3.1 Cahier des charges

Fonction globale : Permettre de suivre les déplacements d'un enfant et identifier les objets dangereux à proximité de ce dernier.



FIGURE 1 – Système de caméra amovible

Milieu	Description
Physique	Le système doit pouvoir de suivre un enfant dans toute une pièce de la maison.
Technique	Le système doit détecter des objets dangereux qui se trouvent à proximité de l'enfant et alerter les parents.
Industriel	Le système doit être simplement reproduit une fois le concept fonctionnel.
Économique	Le coût de fabrication doit être minimal. Au plus, autour de 200\$ (budget alloué aux PFE)
Humain	Le système doit être sécuritaire pour les enfants
Environnemental	Le système doit permettre de récupérer les cartes Arduino et Raspberry Pi du cours de GPA788 ainsi que des moteurs usagés.

TABLE 1 – Contraintes

1.3.2 Spécifications

Voici les spécifications physiques finales du système détaillées dans la Table 3. À noter que dans la configuration finale, il doit y avoir 3 sources de courant (12v pour le Raspberry Pi, 1-12 V pour l'alimentation des moteurs, et 5v pour le Arduino, alimenté sur USB pendant nos tests. D'ailleurs, c'est une amélioration que nous pourrions faire pour le futur.

Déplacement latéral	60 cm
Angle de rotation camera	Horizontal ± 30 degrés
Électrique	
Alimentation Arduino : 5V USB	
Alimentation RaspPi : 12V	
Alimentation Moteur : 12V	
Courant des moteurs : 0,4 Ampères	
Physique	
Déplacement de gauche à droite en 10 secondes.	
Angle d'observation de la caméra	130 degrés

TABLE 2 – Spécifications matérielles

1.3.3 Analyse de faisabilité économique et financière

Pour la partie programmation du projet, nous prévoyons utiliser des librairies open source. Donc dans notre analyse économique, nous incluons majoritairement les dépenses reliées à la plateforme physique ainsi que les coûts de serveur pour l'entraînement. La table 3 résume tous les coûts des matériels qu'il faut acheter pour mener à bien ce projet.

Tableau des prix				
Matériel	Coût (\$)	Qté	Total	Source
Arduino	17,00	1	17,00	Elegoo UNO R3
Nema17	23,00	1	23,00	Nema 17 Stepper Motor, Quimat Stepper Motor Bipolar
DriverMoteur	4,00	1	4,0	Longrunner A4988 Stepstick Stepper Motor Driver
RaspberryPi 4	134,00	1	134,00	CanaKit Raspberry Pi 4 Starter Kit
Caméra	18,59	1	18,59	LABISTS Raspberry Pi 4 Camera Module 1080P 5MP
Rail+ fixations	50,00		50,00	Rail linéaire en aluminium anodisé +Aluminum Profile 2020 Series with Slot 6mm Connection Set
Servomoteur	5,00	1		SG90 9G Micro Servo Motor
Entraînement	22,00	n.a.	22,00	Simulateur de coûts Google cloud
TOTAL			273,59\$	

TABLE 3 – Évaluation des coûts

1.3.4 Identification et évaluation des risques

1. Travailler en temps réel : La détection, la segmentation et l'envoi des instructions de contrôle pour les moteurs doivent se faire en temps réel. Il se pourrait que le temps de calcul soit trop élevé ou encore que la qualité de l'image soit affectée, dans ce cas la détection en temps réel devra être ajustée avec des temps de pause tout dépendants la problématique rencontrée.
2. Traitement d'image avec OpenCV : Les membres de l'équipe ont suivi un cours de réseaux de neurones et intelligence artificielle, mais n'ont jamais utilisé OpenCV dans le cadre de leurs projets, leurs connaissances à ce niveau sont donc limitées.
3. Contrôle des moteurs : Les membres de l'équipe ont déjà utilisé la plateforme Arduino lors du cours d'objets connectés, mais n'ont jamais contrôlé de moteurs.
4. Mise en relation des 2 systèmes de détection : L'équipe a déjà réalisé des projets de détection, mais n'a jamais utilisé plusieurs modèles en même temps. La solution pour mettre en relation le système de détection d'enfant avec le système de détection d'objets dangereux est une notion qui est encore abstraite.
5. Utilisation de serveurs pour alerter les parents : L'équipe a déjà utilisé le module REQUEST pour communiquer par courriel avec un utilisateur, mais n'a jamais utilisé des options de communication SMS, ce qui semble plus adapté dans le cadre de ce projet.

1.4 Revue de la documentation

1.4.1 Modèles et jeux de données

Afin de faire nos premières détections à l'aide de OpenCV, nous avons eu recours à COCO (Common Objects in COntext), un jeu de données d'environ 120 000 images permettant d'entraîner et de tester pour la détection d'objets provenant de 80 classes. En effet, plusieurs modèles sont disponibles avec des poids pré-entraînés sur ce jeu de données. Notamment, parmi les 80 classes disponibles COCO permet de détecter la classe personne, couteau, ciseau et fourchette qui sont tous des éléments pertinents dans le cadre de notre projet. COCO est également une référence populaire pour évaluer la performance des modèles de détection d'objets. D'ailleurs, c'est ainsi que nous avons identifié le modèle utilisé dans notre projet. Tel que nous pouvons l'observer, la figure 2 représente les performances de plusieurs modèles pour le même jeu de données. On peut constater que le modèle choisi, YOLOv4, permet d'obtenir une

bonne précision moyenne (AP pour average precision) sans compromettre la rapidité d'exécution (FPS pour frame per second). Il est important de mentionner qu'il existe une version YOLOv5, cependant, la documentation est beaucoup moins développée.

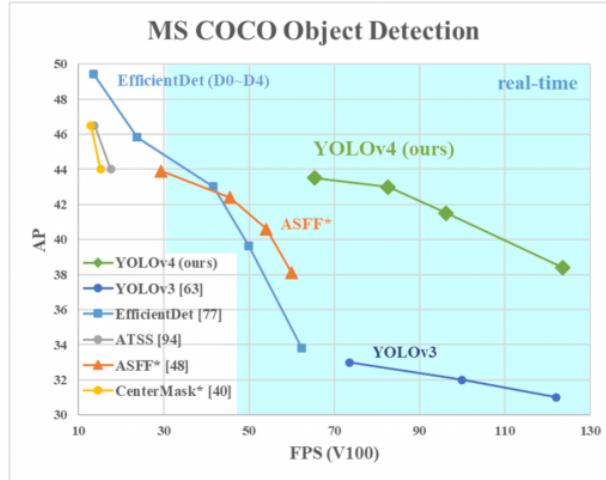


FIGURE 2 – Comparaison de modèles basée sur COCO

En poursuivant nos recherches, nous avons découvert le modèle YOLOv4-tiny, une version compressée du YOLOv4 qui est conçue pour s'entraîner sur des machines avec moins de puissance de calcul. Bien que l'entraînement ne soit pas effectué directement sur le Raspberry Pi, l'ensemble de nos résultats de recherche suggéraient une utilisation du modèle compressé pour un projet avec Raspberry Pi.

Une fois les premières détections effectuées, nous avons eu recours au OIDv4 ToolKit afin de créer notre propre jeu de données. En effet, COCO est un bon point de départ, mais il permet de détecter beaucoup plus de classes que nécessaire, alors qu'il pourrait mieux performer dans les classes d'intérêt. Plus précisément, Open Image Dataset (OID) est un jeu de données comportant 600 classes et plus de 1 700 000 images. Les images disponibles sont annotées et prêtes à être utilisées, ce qui permet une grande économie de temps. L'outil que nous avons utilisé, développé en partie par TheAIGuy [13], permet non seulement de télécharger les images, mais également de convertir les étiquettes dans le bon format pour la détection d'objets avec YOLOv4. Il suffit de sélectionner les classes désirées et le nombre d'images maximum à télécharger, puis le logiciel en téléchargera la quantité désirée ou moins s'il n'y a pas suffisamment d'images disponibles. La découverte de cet outil nous a permis de perfectionner notre modèle, sans toutefois y mettre trop de temps.

Une des principales contraintes des modèles en général est le sous/surentraînement. On peut le constater grâce à la courbe des pertes que l'on obtient lors de l'entraînement des modèles. En effet, à la fin de l'entraînement la courbe continue de descendre, le modèle va être sous-entraîné. D'autre part, si la courbe stagne horizontalement, il va être surentraîné.

1.4.2 Configuration du modèle

Pour la réalisation de ce projet, nous avons choisi d'utiliser Darknet. Plus précisément, il s'agit de l'architecture open source contenant le réseau de neurones. Ce dernier peut être utilisé en dupliquant le répertoire GitHub d'AlexeyAB [15], le créateur de Darknet. Une fois le téléchargement complété, l'arborescence ressemble à ceci :

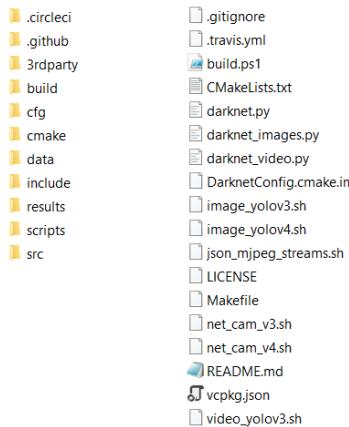


FIGURE 3 – Arborescence des fichiers darknet

La majorité des fichiers intéressants pour la personnalisation du modèle se trouvent dans le répertoire data. En effet, ce dossier contient des fichiers appelés « train.txt » et « test.txt » dans lesquels est indiqué le chemin d'accès vers chaque photo nécessaire à l'entraînement et à la validation. D'ailleurs, il existe plusieurs outils permettant de générer ces deux fichiers, car ils peuvent contenir des milliers de chemins d'accès et doivent pratiquement toujours être modifiés. Le fichier « coco.names » peut être dupliqué, modifié et renommé. Par exemple, dans le cadre de notre projet nous l'avons appelé « obj.names » et il doit lui aussi se trouver dans le répertoire data. Ce fichier contient le nom de toutes les classes que nous désirons détecter. Initialement, il contenait 80 classes, mais notre version n'en contient que deux. Ensuite, « obj.data » initialement « coco.data » se trouve encore dans le même dossier, il contient le nombre de classes ainsi que les chemins d'accès vers « train.txt » et « test.txt » mentionnés précédemment. De plus, il contient le chemin d'accès vers le dossier « backup » où seront enregistrés les poids à toutes les 1000 itérations. En effet, ce dossier permet de reprendre l'entraînement si jamais le programme est interrompu. En revanche, pour faire ce backup, on doit absolument être placé sur Google Drive et non sur la machine virtuelle (VM), car lorsque l'exécution est interrompue l'environnement se ferme et l'ensemble des fichiers se trouvant sur la VM sont supprimés. Le répertoire data contient un dernier fichier important, celui de configuration du modèle dans lequel plusieurs modifications doivent être apportées. Finalement, un dernier fichier est nécessaire à l'entraînement, soit celui contenant les poids du modèle. Contrairement aux autres, il se trouve directement dans le répertoire darknet. Ce fichier peut être téléchargé lorsque des poids pré-entraînés sont utilisés ou obtenu à la suite de l'entraînement du modèle.

Les modifications à apporter dans le fichier de configuration concerne les paramètres suivants : « class », « width », « height », « max_batches », « filters » et « steps ». Pour commencer, le nombre de classes doit être modifié pour chacune des couches YOLO, le modèle tiny YOLO en contient deux. Ensuite, le paramètre « filters » doit correspondre à 3 (nombre de classes +5) et doit être modifié pour les deux couches convolutionnelles qui se trouvent juste avant les couches YOLO. Le reste des paramètres à modifier se trouvent au début du fichier dans la section entraînement. D'abord, le paramètre « max_batches » doit correspondre au nombre de classes multiplié par 2000, il ne doit pas être inférieur à 6000. Donc pour un modèle à seulement une, deux ou trois classes le paramètre correspond à 6000. Ensuite, les deux paramètres associés à « steps » doivent correspondre à 80% et 90% du « max_batches ». Pour terminer, la largeur et la hauteur sont généralement mis à 416, une valeur standard. En réalité, n'importe quel multiple de 32 est valide pour ces deux paramètres. Toutefois, plus la valeur augmente, meilleurs sont les résultats, mais plus l'entraînement est long. C'est pourquoi, 416 est souvent utilisé, car il s'agit du meilleur compromis temps/résultats.

1.4.3 Traitement d'image

Nous avons fait le choix d'utiliser la librairie OpenCV, car il s'agit d'une bibliothèque graphique spécialisée dans le traitement d'image en temps réel. Notamment, OpenCV permet d'isoler des objets dans une image pour ensuite les analyser avec le système de détection. De plus, la librairie permet de lire et d'afficher les vidéos avec le résultat de détection en temps réel, ce qui est indispensable pour notre projet. D'ailleurs, la librairie est bien adaptée pour le Raspberry Pi.

En cours de route, nous avons également fait l'intégration de la librairie Tensorflow Lite. Cette version plus légère, a été développée pour les applications mobiles. Toutefois, même si cette version est allégée, elle regroupe beaucoup de fonctionnalités reliées à l'apprentissage machine qui ne nous sont pas utiles. En effet, l'utilisation d'une telle librairie pourrait sembler inutile puisque OpenCV permet déjà d'effectuer les tâches nécessaires en matière de traitement d'image. Nous expliquerons dans la section « problèmes rencontrés » les raisons qui ont motivés l'intégration de Tensorflow Lite et par la suite, l'abandon de cette idée.

1.4.4 Envoi de notifications à l'utilisateur

Pour l'envoi de notification, l'application client de messagerie en ligne de Gmail nous permet d'envoyer des courriels avec pièces jointes. Il envoie les données vers un serveur de messagerie SMTP (Simple Mail Transfer Protocol). Ce serveur récupère les courriels et les mets au bon emplacement pour que le destinataire puisse y accéder par POP (Post Office Protocol) ou IMAP (Internet Message Access Protocol) [22]. Pour pouvoir envoyer ce courriel avec notre système, il faut définir le MTA (Mail Transfer Agent) à Gmail. Par défaut, ce dernier est sécurisé afin de ne pas effectuer de communication avec un client inconnu. Il faut donc désactiver cette sécurité pour pouvoir communiquer avec notre système [29].

L'envoi de notifications par SMS (Short Message System) se fait également par courriel, car dans le programme chaque numéro de téléphone est suivi de @NomOpérateurTéléphonique.com que l'on appelle passerelle SMS (SMS gateway en anglais) [16]. Cette forme varie quelque peu selon l'opérateur téléphonique, cette information est disponible en ligne.

1.4.5 Choix des moteurs

Le choix des moteurs ne s'est pas basé sur des recherches, nous avons plutôt réutilisé des moteurs que nous possédions déjà, soit un moteur pas-à-pas Nema17. Ces moteurs sont très communs sur les imprimantes 3D. Toutefois après quelques recherches nous avons constaté qu'il s'agit d'un moteur qui ne nécessite pas beaucoup de puissance pour opérer. De plus, l'avantage d'un moteur pas-à-pas est que nous connaissons sa position en fonction du nombre d'impulsions qui lui sont envoyées. Pour faciliter l'utilisation de ces signaux, nous avons utilisé la librairie AccelStopper.

1.4.6 Branchements

Pour contrôler le moteur pas à pas et lui fournir la puissance nécessaire, nous avons utilisé un jockey A4889, voir figure 4. Ce jockey de 12v contrôle la distribution de puissance au moteur Nema. Son courant maximum est de 2 ampères, mais après vérification, lors de ses déplacements le moteur ne consomme que 0,30 ampères au maximum. Le Arduino est alimenté directement via USB, le moteur 9 g ne nécessite pas de jockey. Il est directement alimenté par la sortie 5V de l'Arduino.

1.4.7 Remise à zéro

Pour aider au contrôle du robot et éviter de faire forcer les moteurs, nous avons établi une procédure de retour au point de référence. Au démarrage de la plateforme, la caméra se déplace vers la gauche jusqu'à ce qu'elle active un bouton poussoir, voir figure 4 étiquette 2. Par la suite, sa position est enregistrée comme zéro. La fin du rail se trouve à 6500, soit 22 cm. La librairie AccelStepper permet de garder en mémoire la position du moteur.

1.4.8 Librairie AccelStepper

Cette librairie permet un meilleur contrôle des accélérations, des déplacements et de l'enregistrement de la position, car elle permet de connaître la position du moteur en tout temps.

1.4.9 Protocole de communication I2C

Inspirés par le cours de GPA788, nous communiquons entre le Arduino et le Raspberry Pi à l'aide du protocole I2C. L'objectif est de recevoir la position de l'encadrement et faire les calculs dans le Arduino pour évaluer le déplacement nécessaire à envoyer au moteur. Nous aurions aussi pu utiliser le protocole SPI (Serial Peripheral Interface), mais nous avons choisi d'y aller avec I2C, car il s'agit de la méthode avec laquelle nous sommes le plus familiers.

1.5 Ressources

1.5.1 Ressources humaines

Tout au long de notre parcours, nous avons rencontré des personnes expérimentées que nous avions identifiées comme ressources pour ce projet :

- T. Wong, Professeur superviseur (C++, Python, RaspberryPi, Arduino, etc.)

1.5.2 Ressources matérielles

- Raspberry Pi
- Arduino
- Carte micro SD
- Caméra
- Rail 8mm
- Moteur Nema17
- Rail profilé en aluminium
- Courroies

1.5.3 Ressources logicielles

- Visual Studio Code
- Raspberry Pi OS
- OpenCV
- Google compute engine / Microsoft azure
- Arduino IDE

1.5.4 Méthodologie d'expérimentation

Nous avons choisi de décortiquer le projet en plusieurs sous-problèmes afin de faciliter le travail d'équipe et par le fait même maximiser nos chances d'atteindre les objectifs fixés. Les tests concernant le support amovible seront faits séparément des tests de détection et la mise en commun de ces deux éléments se fera vers la fin du projet. En attendant la mise en commun, le support amovible simulera la réception de données de position. Pour ce qui est de la vision, les tests de détection de l'enfant et les tests de détection d'objets se feront séparément, la mise en commun sera faite plus loin dans le projet à l'aide d'un contrôleur. De plus, s'il est possible de tester le traitement d'image avant de l'intégrer au modèle, cette option sera priorisée. De cette manière, nous serons assurés de son bon fonctionnement, ce qui réduit les sources d'erreur.

2 Processus de conception

2.1 Traitement d'image et détection d'objets

2.1.1 Personnalisation du modèle YOLOv4-tiny

Dans le but de maîtriser OpenCV, nous avons commencé par utiliser les poids pré-entraînés avec COCO et le modèle de base qui détecte 80 classes. Pour ce faire, nous avons rassemblé plusieurs objets, compris dans les classes détectées par YOLOv4 ainsi que des intrus. Les photos ont été prises dans différents scénarios, soit des scénarios idéals et d'autres plus complexes afin d'évaluer la qualité des résultats. À l'aide de Jupyter Notebook, nous avons exécuté un court programme permettant de tester la détection d'objets sur nos photos. Après avoir obtenu nos premiers résultats et développé une meilleure compréhension du sujet, nous avons choisi de modifier notre manière de travailler. Effectivement, pour d'obtenir un modèle personnalisé il fallait éventuellement réentraîner, ce qui requiert des GPU (Graphics Processing Unit). Nous avons donc eu recourt à la machine virtuelle de Google. Il suffit de la synchroniser avec Google Drive pour avoir accès à tous nos documents. De ce fait, l'arborescence des fichiers nécessaires à l'entraînement n'a pas besoin d'être recommandée à chaque utilisation. L'exécution des commandes se fait à partir d'un fichier Google Collab dans lequel il est possible d'activer l'utilisation des GPU.

Une fois une bonne méthode de travail établie, nous avons procédé à l'installation de différents outils. Premièrement, l'installation du OIDv4 ToolKit nous a permis de télécharger une grande quantité d'images prêtes à être utilisées pour les classes personne et couteau. Ensuite, nous avons procédé à la modification des fichiers « coco.names », « coco.data » ainsi que le fichier de configuration afin d'obtenir un modèle qui ne détecte que les deux classes d'intérêt. Les modifications à effectuer dans ces documents sont décrites dans la section revue de documentation, configuration du modèle. Une fois ces tâches réalisées, nous avons débuté l'entraînement. Plus précisément, nous avons exécuté un entraînement pour ne détecter que les couteaux, un deuxième pour ne détecter que les personnes ainsi qu'un troisième pour détecter les personnes et les couteaux. En effet, les matrices de confusion peuvent être complexes à analyser pour un problème à plus d'une classe. Toutefois, nous ne savions pas non plus comment intégrer deux modèles, nous avons donc préféré entraîner plusieurs fois pour obtenir les poids associés aux deux scénarios envisageables. Finalement, nous avons utilisé les poids obtenus avec l'entraînement et nous avons exécuté de nouvelles commandes pour tester la détection d'objets à partir d'un flux vidéo.

2.1.2 Intégration du système sur le Raspberry Pi

Nous avons commencé par faire fonctionner notre modèle sur un ordinateur pour s'assurer de son bon fonctionnement. Ensuite, nous avons procédé à l'intégration du système sur le Raspberry Pi 4. Pour ce faire, nous avons installé la librairie de OpenCV à l'aide du package d'installation pip3. Nous avons poursuivi avec l'installation des librairies Numpy et Matplotlib. Ces librairies nous permettent d'identifier l'enfant et les objets dangereux en plaçant des rectangles autour des éléments détectés. Initialement nous voulions utiliser TensorFlow Lite afin de pour notre système mais la conversion entre TensorFlow, que nous avons utilisé pour l'entraînement et TensorFlow lite sur le Raspberry Pi il y avait des erreurs de segmentation. Cette option avait été envisagé car elle nous permettait de nous baser sur la documentation de TheAIGuy [23]. Nous avons donc écrit le programme de détection en python directement sur le Pi grâce à visual studio code. Nous nous sommes basés sur un article de GeeksforGeeks [6] n'utilisant pas TensorFlow.

Pour procéder à l'installation, nous utilisons le Raspberry Pi directement avec un écran afin de faciliter les communications. Nous avons abandonné l'idée d'utiliser la communication ssh, car la latence amplifie la lenteur du système et il est parfois compliqué pour le système de retrouver la communication une fois perdue.

2.2 Support de caméra amovible

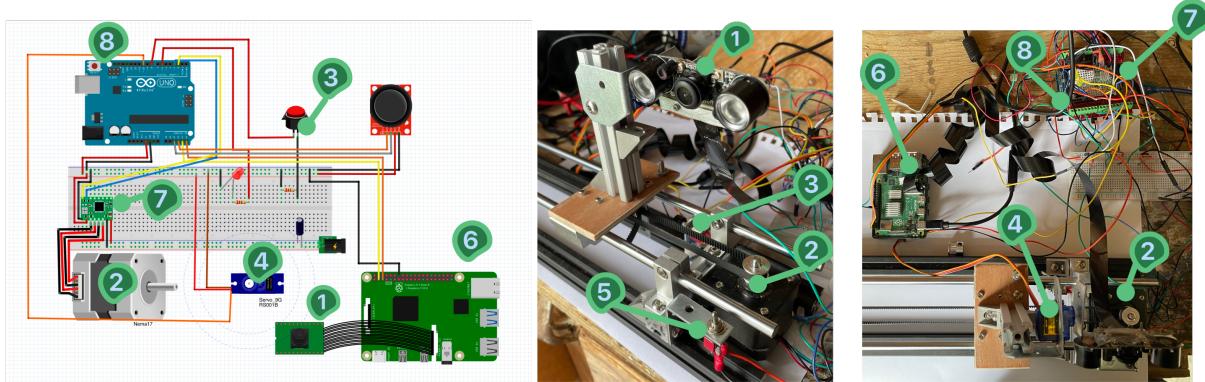


FIGURE 4 – Montage physique

Pour le matériel final, nous avons utilisé le matériel que nous avions choisi dans le rapport préliminaire (table 4). La figure 4 nous détail les branchements qui ont été effectués. La liste exhaustive du matériel que nous avons utilisé se trouve en annexe. Pour le reste de la section, nous traiterons des spécifications du système, le fonctionnement au moment de la réception des données de positionnement, l'architecture des classes utilisées, le détail de l'algorithme de positionnement, une revue des résultats et une analyse de ces derniers.

2.2.1 Mise en place du protocole I2C

Pour le transfert de données tel qu'imaginé, nous avons utilisé le BUS I2C [31]. La vitesse de transfert est suffisante puisque la détection ne se faisait qu'à un ratio de 1 détection la minute. Pour permettre le transfert, étant donné que les valeurs pouvaient varier de 0 à 800, les 255 bits offerts par un codage sur 8 bits n'était pas suffisant. Pour régler le problème, les données en provenance du Raspberry Pi sont envoyées dans un tableau de 9 bits.

CMD	XmaxMSB	XmaxLSB	XminMSB	XminLSB	YmaxMSB	YmaxLSB	YminMSB	YminLSB
-----	---------	---------	---------	---------	---------	---------	---------	---------

FIGURE 5 – Structure de données

La première envoi la commande au Arduino. Les commandes implémentées sont :

1. Envoie de Coordonnées
2. Démarrage ou arrêt manuel du système

2.2.2 Architecture du système de contrôle

Le graphique UML suivant indique le fonctionnement du système. Pour faciliter le contrôle, nous avons utilisé deux bibliothèques pour aider le déplacement des moteurs pas-à-pas et des servomoteurs (AccelStepper et servo). Par contre, pour organiser les données reçues à travers le I2C et les analyser, nous avons mis en place deux autres classes. La première est COORDONNÉE et la seconde est CORDO. Créer une instance de Coordonnée permet d'avoir un endroit où stocker les lectures de données effectuées sur le BUS. Cette classe permet également de calculer le centre et le centre. La deuxième classe, CORDO, est le cœur du programme. Une instance de cette classe permet de coordonner les mouvements des moteurs. Pour créer un coordonnateur, il faut lui fournir les pointeurs des moteurs utilisés et un pointeur de COORDONNÉE. Avec les informations du coordonnateur et certaines informations contenues sur le moteur pas-à-pas (ex. : position actuelle), il peut donner au contrôleur du moteur la prochaine position à aller.

Notamment, le CORDO est responsable de s'assurer que le moteur se déplace dans les limites physiques du système. Pour ce faire, au démarrage, une séquence de remise à zéro est lancée. Le robot tourne

contre la montre jusqu'à ce qu'il appuie sur le bouton (3). Cette position est la position 0. La position finale du moteur, avant de rentrer en contact avec la limite du rail, est de -6500. Cette valeur est entrée dans le programme et dans ces déplacements, le CORDO s'assure de ne jamais dépasser cette valeur.

Pour éviter trop de mouvement de la caméra à cause des variations de détection, une zone morte a été mise en place. Si le coordonnateur calcule que son déplacement requis est sous 100 pixels, le déplacement ne se fait pas. Cela évite de voir la caméra bouger même si le sujet ne bouge pas.

Un problème avec l'architecture actuelle est que le PI et le arduino ne sont pas cadencés. Peu importe le rythme de détection, le arduino reçoit les nouvelles positions de détection. Ces données sont perdues si le arduino est en mouvement et qu'il n'est pas disposé à recevoir une nouvelle position. Il faudrait mettre en place une communication Arduino-PI qui donnerais l'état du Arduino pour savoir si c'est un bon moment pour envoyer des nouvelles données.

On retrouve aussi dans cette classe quelques fonctions utilitaires :

1. DeplacementPourcentage()

Cette fonction permet à l'utilisateur de se déplacer à un certain pourcentage sur le rail, peu importe sa dimension (50 % sera toujours au centre du rail).

2. ControleJoystick : Cette fonction permet de faire la lecture des positions du joystick et modifie la position du rail en fonction de du positionnement du joystick.

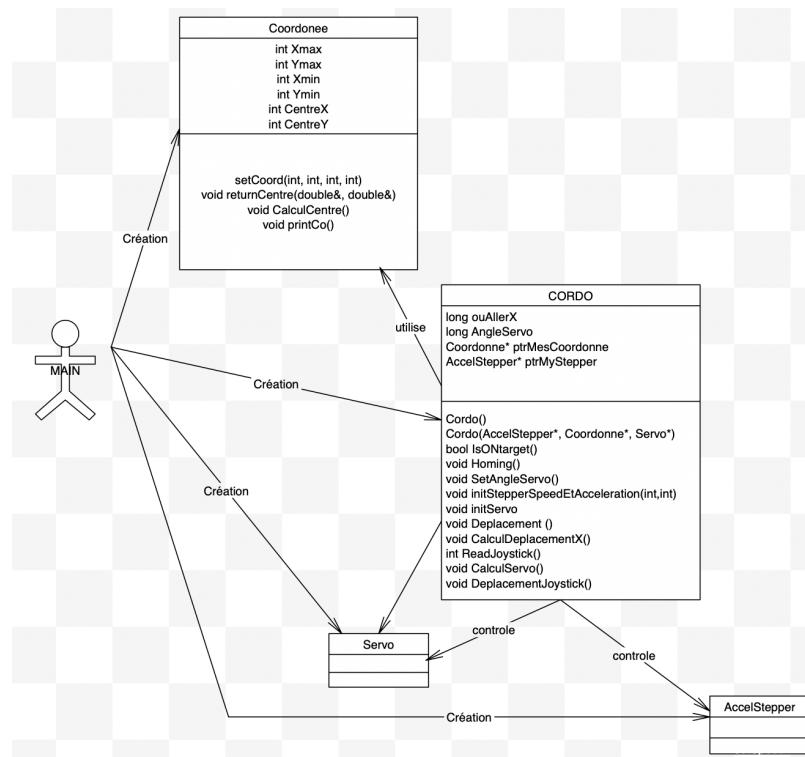


FIGURE 6 – Diagramme de classe

2.2.3 Calcul des positions

Le calcul de prochaine position sur le rail et l'angle de la caméra est fait de manière linéaire. Lorsque le système détecte un centre inférieur à 300 (400 - zoneMorte).

$$\text{FacteurConversion} = \frac{\text{TailleRail}}{\text{DimensionX}} = \frac{-6400}{800} = -8 \quad (1)$$

$$(CentreCamera - CentreX) * \text{FacteurConversion} = \text{Deplacement}$$

$$400 - 150 * -8 = -1600 \quad (2)$$

$$\text{Deplacement} = -1600 \quad (3)$$

(4)

La nouvelle position du moteur envoyé par le COORDO sera :

$$\text{myStepper} \rightarrow \text{moveTo}(\text{position}) \quad (5)$$

$$\text{position} = -3200 - (-1600) = 1600$$

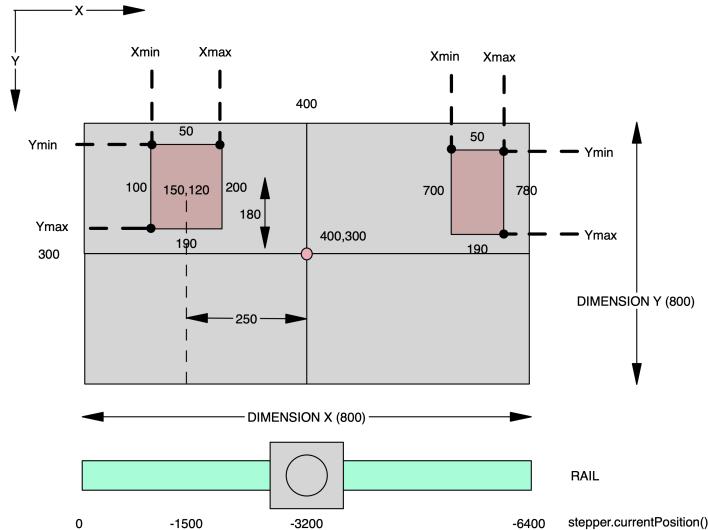


FIGURE 7 – Calcul de la position

Tel qu'expliqué dans la section d'architecture, la zone morte se situe entre 300 et 500, pour limiter le mouvement.

2.2.4 Déplacement du servomoteur

Le Servomoteur se déplace grâce au contrôle donné par la librairie SERVO[19] de Arduino. Sa position est limitée avec la fonction MAP et est paramétrée en fonction de l'angle inférieur maximum et minimum ainsi que la hauteur(Dimension). Les valeurs sont modifiables dans le fichier de paramètres VALEURS.H.

2.2.5 Déplacement de la caméra avec un joystick

Comme discuté au début du projet, nous avons continué avec la librairie AccelStepper. Par contre, alors que nous l'avions choisi pour avoir plus de contrôle sur la vitesse et l'accélération, nous n'avons pas utilisé ces paramètres. Cependant, il reste à la disposition d'utilisateur futur. De plus, nous avons rajouté une fonctionnalité supplémentaire qui permet de prendre le contrôle de la caméra avec les lectures de valeurs du joystick [11]. L'utilisateur doit sélectionner les ports sur lesquels est connecté le joystick. Par la suite, dans le programme python, lors de l'exécution, la touche M sert à activer le mode Manuel et la touche « O » à l'éteindre.

2.2.6 Test de la plateforme

Voici les quelques tests effectuer sur la plateforme pour évaluer le fonctionnement général. Par la suite, nous discuterons des résultats.

Test	Observations	Résultats
1	Sujet complètement à gauche	La caméra se déplace en x sec
2	Sujet complètement à droite	La caméra se déplace en x sec
3	Le sujet bouge de 30 cm (G ou D)	La caméra va dans la bonne direction
4	Le sujet se lève	Modification de l'angle de camera
5	Aucun déplacement du sujet	La caméra reste sur sa position

TABLE 4 – Tableau des résultats de test

3 Analyse des résultats

3.0.1 Plateforme physique

Physiquement, les résultats sont concluants. La caméra se déplace sur le rail et son SERVO sans rentrer en contact avec la plateforme. Avec le système actuel, le déplacement doit être complété avant de faire un nouveau mouvement. Dans le cas d'un mouvement rapide de gauche à droite, la caméra se déplace en coup saccadé. Une solution à ce problème serait de mettre à jour la position même pendant le déplacement de la caméra. Il faudrait traiter le déplacement à l'aide d'interruption plutôt que de vérifier continuellement le statut du moteur. Cette modification permettrait de rendre le mouvement moins saccadé. Puisque la caméra a un grand angle, l'utilité du déplacement sur le rail est discutable. La translation ne rajoute que 60 cm de jeux au système ce qui n'a pas une grande valeur. Par contre, le système est facilement paramétrable pour un rail qui serait plus grand. On pourrait utiliser le même système et augmenter la dimension du rail. Cela dit, le déplacement transversal pourrait être remplacé par une rotation pour permettre d'observer une plus grande zone. Le rythme de détection avec un FPS fonctionne bien avec la plateforme puisque cela laisse un peu de temps à la caméra de se déplacer.

3.0.2 Métriques de performance du modèle

Dans la détection d'objets, il existe deux critères à considérer :

1. L'encadrement également appelé bounding box de l'objet dans l'image
2. La classe de l'objet

Donc, il faut correctement encadrer les objets à détecter et classer correctement ces objets. L'encadrement des objets est mesuré par le IoU (Intersection of union) qui représente le degré de recouvrement entre le bounding box des objets et le bounding box produit par l'algorithme de détection. Habituellement, un recouvrement $\text{IoU} > \alpha$ avec $\alpha = 0.5$ est considéré comme acceptable.

Voici les règles pour déterminer le score des détections pour une image :

- Vrai positif (VP) – $\text{IoU} > \alpha$ ET classification correcte ;
- Faux positif (FP) – $\text{IoU} \leq \alpha$;
- Faux négatif (FN) – Un objet dans l'image n'est pas détecté
- Vrai négatif (VN) – N'est pas utilisé dans la détection des objets car il peut exister un très grand nombre de BB qui ne recouvrent pas d'objet dans une image.

Ainsi, nous pouvons avoir plusieurs scores associés à une même image puisque l'algorithme de détection peut produire plus d'un bounding box, si plusieurs objets sont dans l'image, ou aucun si aucun objet n'est dans l'image.

	Personne	Couteau
VP	339	21
FP	512	17
VN	289	8

TABLE 5 – Données de validation

Précision, rappel et F1-score

À l'aide de ces trois informations (VP, FP, FN), nous pouvons calculer la précision, le rappel et le F1-score. La précision (P) peut être interprétée comme la proportion des détections qui s'avère être correcte en réalité. La métrique rappel (R), quant à elle, mesure la proportion des objets détectés. Le F1-score est simplement la moyenne harmonique entre P et R. Pour les données obtenues du projet PFE, nous avons :

$$P = \frac{VP}{VP+FP} \quad R = \frac{VP}{VP+FN} \quad F1 = \frac{2PR}{P+R}$$

Personne :

$$P = \frac{339}{339+512} = 0.4 \\ R = \frac{339}{339+289} = 0.54 \\ F1 = 0.46$$

Couteau :

$$P = \frac{21}{21+17} = 0.55 \\ R = \frac{21}{21+8} = 0.72 \\ F1 = 0.62$$

Constatation : seule la proportion des couteaux détectés est bonne, le reste des mesures sont faibles.

Average precision (AP) et mean average precision (mAP)

La métrique AP est calculée à partir de la courbe Précision versus Recall. Elle donne une valeur scalaire $AP \in [0, 1]$ qui résume la performance moyenne de l'algorithme de détection pour une classe d'objets. La définition exacte de la métrique AP est

$$\int_0^1 p(r) dr \quad (6)$$

où $p(r)$ est la précision associée à la valeur de rappel r . Autrement dit, AP est l'aire sous la courbe de PR. On remarque que la courbe Précision versus Recall est une fonction du seuil ϵ et sa forme peut varier selon la valeur de ce seuil. Une fois toutes les images traitées par l'algorithme de détection, on peut tracer une courbe Précision versus Recall par classe d'objets. En voici un exemple :

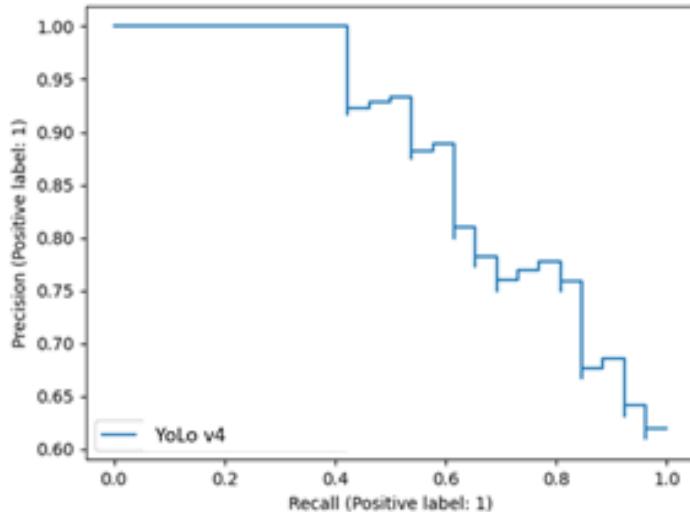


FIGURE 8 – Courbe de Précision versus Recall

Clairement, cette courbe n'est pas facilement intégrable. En pratique, la valeur de AP est calculée par une approximation rectangulaire ou intégration numérique.

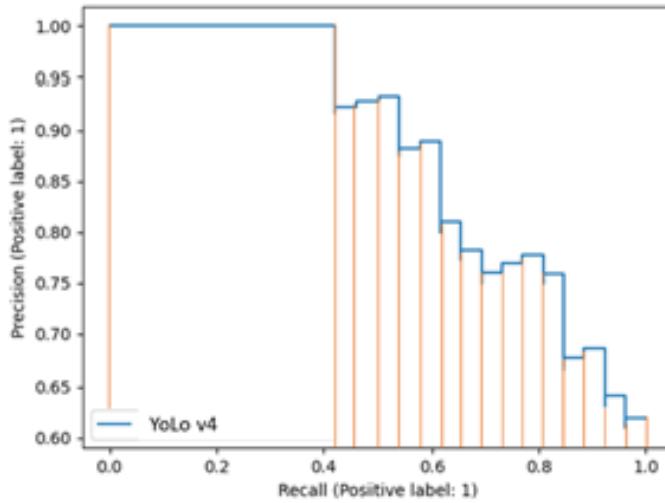


FIGURE 9 – Courbe de Précision versus Recall avec aire

Ainsi, la valeur AP est la somme des aires rectangulaires ainsi formées. Pour les données obtenues du projet PFE, nous avons :

	Personne	Couteau
AP	0.3590	0.6658
mAP	$(0.3590 + 0.6658)/2 = 0.5124$	

TABLE 6 – Précision du modèle

D'après ce résultat, la détection des couteaux est nettement supérieure à la détection des personnes. Enfin, la métrique mAP (mean average precision) est simplement la moyenne arithmétique des AP,

$$mAP_{IoU \alpha} = \sum_{k=1}^K AP_{IoU \alpha}^{(k)} \quad (7)$$

où K est le nombre de classes à détecter.

La figure 10 a été générée grâce aux données de validation. Tel que nous pouvons l'observer la mAP est de 51,2%, ce qui signifie que la performance globale du modèle n'est pas très bonne. Toutefois, il s'agit de la performance attendue par l'équipe. En effet, lors de nos recherches sur les différents modèles, nous avons analysé plusieurs graphiques comme celui de la figure 2, présentée dans la section revue de documentation. Notamment, un de ces graphiques indiquait que pour des images de la même taille que les nôtres, la précision du modèle YOLOv4 sur 1 image se situe entre 45% et 50%, tout dépendant le nombre de FPS [15]. Considérant que les performances du tiny YOLOv4 sont supposés être inférieures à celles du YOLOv4, nous pouvons conclure que les résultats obtenus sont plutôt satisfaisants considérant les limitations connues du modèle.

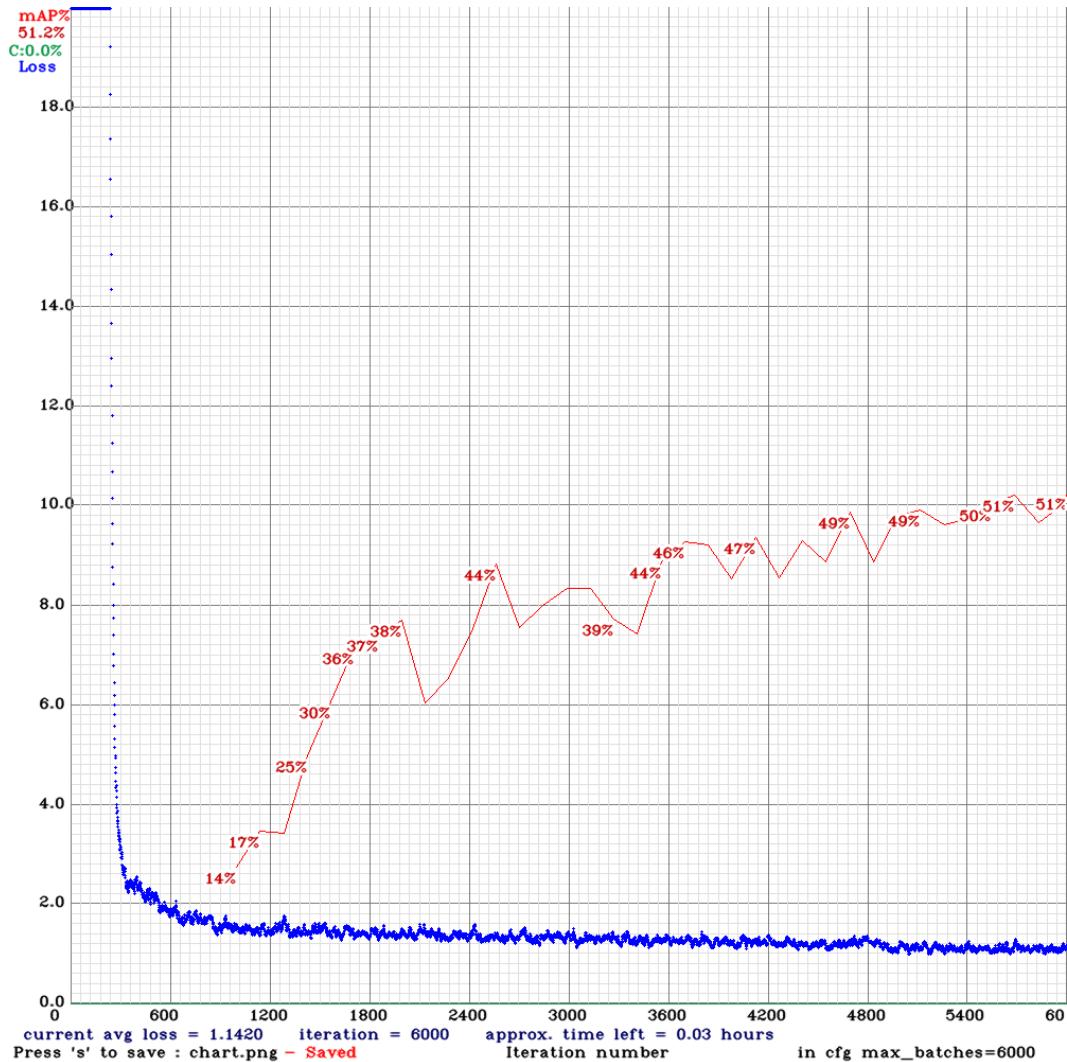


FIGURE 10 – Résultats de validation

3.0.3 Nombre de paramètres du modèle

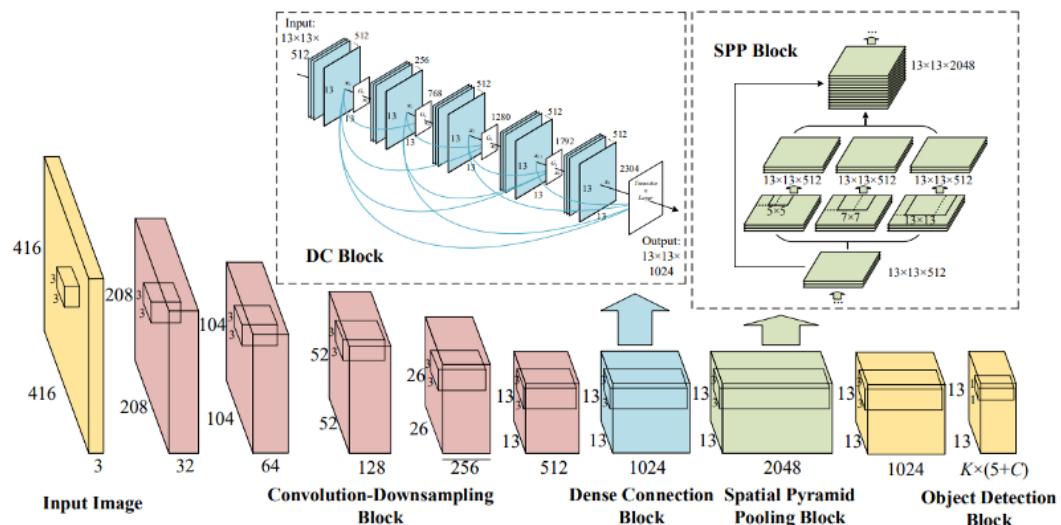


FIGURE 11 – Réseau YOLOv4

Le nombre de paramètres entraînables d'un modèle est une métrique importante, car elle indique sa capacité ainsi que sa complexité. Tel que nous l'avons mentionné précédemment, YOLOv4 était notre premier choix de modèle, puis nous avons changé pour YOLOv4-tiny, car il s'agit d'un modèle plus adapté pour la capacité de calcul du RaspberryPi. En effet, la taille de YOLOv4-tiny est beaucoup plus petite que celle du YOLOv4, car le nombre de couches convolutionnelles est réduit, section « Convolution-Downsampling Block » de l'image 11. De plus, YOLOv4-tiny possède une couche YOLO de moins que le YOLOv4. Ces différences expliquent pourquoi le modèle YOLOv4 compte plus de 60 000 000 de paramètres entraînables, alors qu'après calcul nous avons totalisé 2 505 024 paramètres pour le YOLOv4-tiny. D'ailleurs, le temps d'entraînement pour le modèle YOLOv4 s'élevait à plus de 24h, alors que celui pour le tiny YOLOv4 était autour de 2h seulement.

3.0.4 Rapidité du traitement d'image

La détection est très gourmande en calcul pour le processeur ce qui provoque des surchauffes du Raspberry Pi. Nous avons donc ajouté une pause de 100 ms après chaque détection afin de ralentir notre système et donc de réduire les calculs. Un dissipateur de chaleur a également été installé sur le processeur. Une fois que tout notre système fut bien fonctionnel, nous avons fait rouler notre programme pour mesurer sa vitesse. Nous avons fait la mesure sur 100 images et nous avons obtenu une vitesse moyenne de 1,214 FPS. Cette vitesse est largement suffisante pour notre projet, car un enfant en bas âge ne se déplace pas très rapidement et notre système peut aisément le suivre.

4 Conclusion et recommandations

Lors de la réalisation de ce projet, plusieurs améliorations potentielles ont été notées. Malheureusement, le temps ne nous a pas permis d'apporter les correctifs qui seront mentionnés dans cette section. En effet, la mise en commun des trois objectifs a été priorisée par rapport à l'amélioration individuelle de chacun. Pour commencer, la rapidité d'exécution a été une source de compromis tout au long du projet. En effet, il existe des outils bien plus performants que le Raspberry Pi et le Arduino, mais ces derniers ont été choisis, car il s'agit d'un moyen peu coûteux et efficace pour les preuves de concept. Notamment, cet aspect a influencé le choix du modèle et a eu un grand impact sur la détection en temps réel. Avec plus de temps, une optimisation du code aurait pu permettre une meilleure gestion des ressources et certains outils auraient pu être intégrés pour améliorer cette faiblesse.

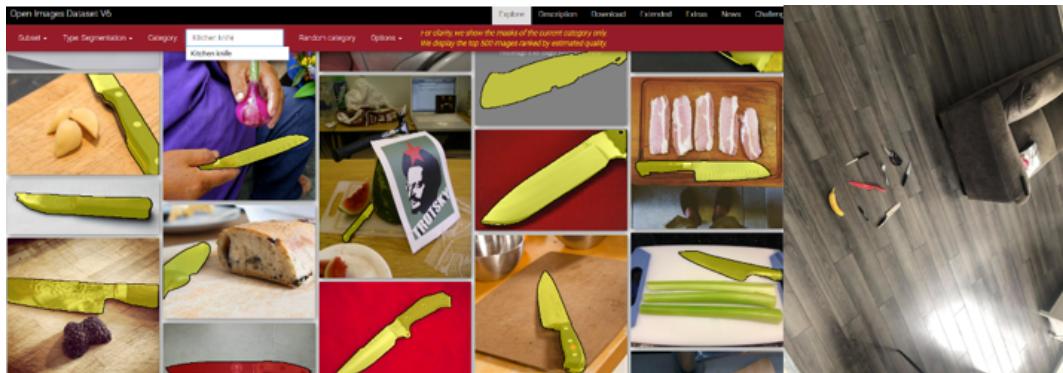


FIGURE 12 – Images d'entraînement vs. images réelles

Ensuite, trois améliorations auraient pu être apportées au niveau de la détection. Premièrement, la précision des résultats aurait pu être améliorée en utilisant notre propre jeu de données. Cependant, cette amélioration demande beaucoup de temps et d'efforts, car pour être bénéfiques une grande quantité d'images doivent être étiquetées. De plus, il est important que le jeu de données ait une grande variété d'images pour bien généraliser sans surentraîner le modèle, ce qui peut être délicat. Ensuite, tel que nous l'avons mentionné dans les objectifs, la détection de personne ne gère pas le cas où plusieurs enfants se trouvent dans la pièce. Puisque ce scénario est fort probable dans la réalité, il s'agirait d'une amélioration intéressante. Finalement, la détection de plusieurs classes d'objets dangereux serait également une amélioration intéressante. Cette dernière demande beaucoup de travail, car pour chaque classe ajoutée un jeu de données doit exister ou être créé.

De plus, plusieurs modifications pourraient être apportées à l'outil de notification. Dans un monde idéal, une application aurait été développée et plusieurs fonctionnalités auraient été disponibles. Par exemple, l'accès au flux vidéo en direct, la personnalisation des objets détectés, etc.

Finalement, la plateforme physique contient elle aussi quelques éléments qui pourrions être amélioré. Dans le code du Arduino, mettre en place une architecture qui n'attend pas la fin d'un mouvement pour en commencer un autre pourrait permettre de faire évoluer la position voulue en fonction du déplacement actuel. Cela permettrait selon nous de réduire les déplacements saccadés. La librairie AccelStepper offre aussi plusieurs outils pour gérer les accélérations et les vitesses. Un meilleur contrôle de l'accélération pourrait permettre de rendre les déplacements plus doux. La plateforme a apporté son lot de défi au niveau des branchements. Maintenant que le système est fiable, créer un circuit imprimé dédié à notre plateforme pourrait permettre de faciliter sa manipulation. Au niveau du code du CORDO, il serait intéressant d'inclure les SETTER pour permettre des applications futures de modifier les paramètres sans rentrer dans le fichier de configuration.

Tous les programmes de notre système sont disponible sur notre Github [24] ainsi que le modèle et toutes informations nécessaire à la reproduction de notre projet. De plus, l'accès est public donc cela permet de contribuer à aider quiconque voudrait réaliser un projet similaire.

Références

- [1] Commissariat à la protection de la vie privée du CANADA. *Lignes directrices sur la surveillance vidéo au moyen d'appareils non dissimulés dans le secteur privé*. fra. Last Modified : 2008-03-06. Mar. 2008. URL : https://www.priv.gc.ca/fr/sujets-lies-a-la-protection-de-la-vie-privee/surveillance/videosurveillance-par-les-entreprises/gl_vs_080306/ (visité le 15/07/2021).
- [2] Bernard BAYLE. "Robotique mobile". In : (jan. 2016).
- [3] *A4988 Stepper Motor Driver with Arduino Tutorial (4 Examples)*. en-US. Fév. 2019. URL : <https://www.makerguides.com/a4988-stepper-motor-driver-arduino-tutorial/> (visité le 01/07/2021).
- [4] Darshan ADAKANE. *Object Detection with OpenCV-Python using YOLOv3*. en. Nov. 2019. URL : <https://medium.com/analytics-vidhya/object-detection-with-opencv-python-using-yolov3-481f02c6aa35> (visité le 21/05/2021).
- [5] BRAINY-BITS. *Homing stepper motors using the AccelStepper library !* en. Oct. 2020. URL : <https://www.brainy-bits.com/post/homing-stepper-motors-using-the-accelstepper-library> (visité le 05/07/2021).
- [6] *Detect an object with OpenCV-Python*. en-us. Section : Python. Jan. 2020. URL : <https://www.geeksforgeeks.org/detect-an-object-with-opencv-python/> (visité le 02/08/2021).
- [7] *How To Calculate the mean Average Precision (mAP) - an overview*. en-GB. Déc. 2020. URL : <https://hungsblog.de/en/technology/how-to-calculate-mean-average-precision-map/> (visité le 23/07/2021).
- [8] Murat KARAKAYA. *LSTM : Understanding the Number of Parameters*. en. Déc. 2020. URL : <https://medium.com/deep-learning-with-keras/lstm-understanding-the-number-of-parameters-c4e087575756> (visité le 30/07/2021).
- [9] Jacob SOLAWETZ, Samrat Sahoo JUL 01 et 2020 6 Min READ. *Train YOLOv4-tiny on Custom Data - Lightning Fast Object Detection*. Juil. 2020. URL : <https://blog.roboflow.com/train-yolov4-tiny-on-custom-data-lighting-fast-detection/> (visité le 30/07/2021).
- [10] Jacob SOLAWETZ et al. *Scaled-YOLOv4 is Now the Best Model for Object Detection*. Déc. 2020. URL : <https://blog.roboflow.com/scaled-yolov4-tops-efficientdet/> (visité le 23/06/2021).
- [11] XUKYO. *Utilisation d'un Joystick avec Arduino • AranaCorp*. fr-FR. Avr. 2020. URL : <https://www.aranacorp.com/fr/utilisation-dun-joystick-avec-arduino/> (visité le 05/07/2021).
- [12] ANM-P4F. *ANM-P4F/ObjectTrackingCamera*. original-date : 2020-02-16T09:34:58Z. Mar. 2021. URL : <https://github.com/ANM-P4F/ObjectTrackingCamera> (visité le 15/07/2021).
- [13] The AI GUY. *theAIGuysCode/OIDv4_ToolKit*. original-date : 2020-01-04T00:57:50Z. Juin 2021. URL : https://github.com/theAIGuysCode/OIDv4_ToolKit (visité le 24/06/2021).
- [14] The AI GUY. *theAIGuysCode/yolov4-custom-functions*. original-date : 2020-08-02T13:25:04Z. Juin 2021. URL : <https://github.com/theAIGuysCode/yolov4-custom-functions> (visité le 30/06/2021).
- [15] Joseph REDMON. *pjreddie/darknet*. original-date : 2014-04-11T07:59:16Z. Juin 2021. URL : <https://github.com/pjreddie/darknet> (visité le 23/06/2021).
- [16] *SMS gateway*. en. Page Version ID : 1036055459. Juil. 2021. URL : https://en.wikipedia.org/w/index.php?title=SMS_gateway&oldid=1036055459 (visité le 30/07/2021).
- [17] TECHZIZOU. *TRAIN A CUSTOM YOLOv4-tiny OBJECT DETECTOR (Using Google Colab)*. en. Juin 2021. URL : <https://medium.com/analytics-vidhya/train-a-custom-yolov4-tiny-object-detector-using-google-colab-b58be08c9593> (visité le 18/06/2021).
- [18] *21.png (800×400)*. URL : https://pylessons.com/static/images/TensorFlow-2.x-YOLOv3/10_YOLOv4_TF2_introduction/21.png (visité le 30/07/2021).
- [19] *Arduino - Servo Librairie / Référence du Langage Arduino en Français*. fr. URL : <https://arduinogetstarted.com/fr/reference/library/arduino-servo-library> (visité le 06/08/2021).
- [20] *Caméras de vidéo surveillance agricole*. fr-FR. URL : <http://www.agritechnologies.fr/nos-produits/vid%C3%A9o-surveillance-agricole/> (visité le 02/08/2021).

- [21] *darshanadakane/yolov3_objectdetection.* en. URL : https://github.com/darshanadakane/yolov3_objectdetection (visité le 21/05/2021).
- [22] *Email Security – Cyber Security Enthusiast.* URL : <https://my-cybersecurity.blog/category/email-security/> (visité le 30/07/2021).
- [23] *GitHub - theAIGuysCode/tensorflow-yolov4-tflite : YOLOv4, YOLOv4-tiny, YOLOv3, YOLOv3-tiny Implemented in Tensorflow 2.0, Android. Convert YOLO v4 .weights tensorflow, tensorrt and tflite.* URL : <https://github.com/theAIGuysCode/tensorflow-yolov4-tflite> (visité le 01/07/2021).
- [24] *KevinClercy/Yolov4_Tiny-sur-Raspberry-Pi-4 : Système de detection d'objet sur Raspberry Pi 4.* en. URL : https://github.com/KevinClercy/Yolov4_Tiny-sur-Raspberry-Pi-4 (visité le 09/08/2021).
- [25] *Loi sur la protection des renseignements personnels dans le secteur privé.* fr-CA. Jurisdiction : Québec, Canada. URL : <http://legisquebec.gouv.qc.ca/fr>ShowDoc/cs/P-39.1> (visité le 15/07/2021).
- [26] MIKE McCUALEY. *AccelStepper : AccelStepper library for Arduino.* URL : <http://www.airspayce.com/mikem/arduino/AccelStepper/> (visité le 05/07/2021).
- [27] *Open Images Dataset V6.* URL : <https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F058qzx> (visité le 21/05/2021).
- [28] *OpenCV : Drawing Functions.* URL : https://docs.opencv.org/master/d6/d6e/group__imgproc__draw.html#ga07d2f74cadcf8e305e810ce8eed13bc9 (visité le 21/05/2021).
- [29] “Python sur le Pi (II)”. fr. In : (), p. 20.
- [30] Q-ENGINEERING. *Install OpenCV 4.5 on Raspberry Pi 4 - Q-engineering.* en-GB. URL : <https://qengineering.eu/install-opencv-4.5-on-raspberry-pi-4.html> (visité le 27/05/2021).
- [31] TONYWONG. “Configuration et test du port I2C sur le Pi”. fr. In : (), p. 14.