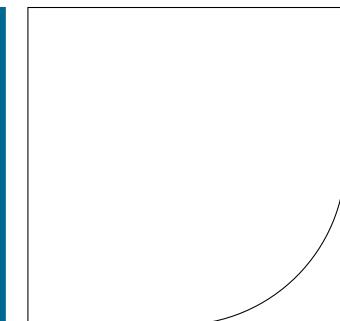


ENSTA Bretagne
2 rue Francois Verny
29200 Brest Cedex
France
tel +33 (0)2 98 34 88 00
www.ensta-bretagne.fr



Rapport de projet Multi-Cam

23 avril 2024

T. El Hajjar A.Dermouche R. Geta R. Ghanem T. Thuillier
tamara.el_hajjar@ensta-bretagne.org
alexandre.dermouche@ensta-bretagne.org
rayane.geta@ensta-bretagne.org
rayen.ghanem@ensta-bretagne.org
theo.thuillier@ensta-bretagne.org

Encadrants: A. Toumi J.C. Cexus

Table des matières

1	État de l'art	5
2	Ingénierie-Système et Gestion de Projet	6
2.1	Ingénierie-Système	6
2.1.1	Exigences du projet	6
2.1.2	Architecture externe	7
2.1.3	Exigences techniques du projet	8
2.1.4	Architecture interne	9
2.2	Gestion de projet	10
2.2.1	Méthode Agile	10
2.2.2	Diagramme de Gantt	11
2.2.3	Mise en place du Git et de Jira	11
3	Développement du dispositif	12
3.1	Réseau Wi-Fi local	12
3.2	Caméras ESP-32 Cam	12
3.2.1	Choix du matériel et caractéristiques	12
3.2.2	Programmation des cartes ESP32-CAM	13
3.2.3	Mise en place d'un circuit électronique de développement pour les ESP32-CAM	13
3.2.4	Programme embarqué de capture et diffusion d'image sur flux	14
3.2.5	Prise en charge de plusieurs flux vidéos d'ESP32 Cams en simultané sur le logiciel PC	14
3.3	Détection des personnes et pose estimation	15
3.3.1	Haar Cascade	15
3.3.2	OpenPose	15
3.3.3	YOLO	16
3.3.4	Première méthode d'implémentation de la multipose (Semestre 3)	17
3.3.5	Deuxième méthode d'implémentation de la multipose (Semestre 4)	18
3.3.6	Obtention de la liste des humains présents	19
3.3.7	Première méthode de reconnaissance d'état d'activité	20
3.3.8	Deuxième méthode de reconnaissance d'état d'activité basée sur un modèle de markov caché	20
3.3.8.1	Définition des états	22
3.3.8.2	Graphe des états	22
3.3.8.3	Matrice de transition	23
3.3.8.4	Définition des observations	24
3.3.8.5	Matrices d'observation	24
3.3.8.6	Estimation de séquences d'états par l'algorithme de Viterbi	25
3.3.9	Ré-entraînement de YOLOv7	25
3.3.9.1	Constitution du jeu de données	25
3.3.9.2	Ré-entraînement sur PC Windows avec support de CUDA	28
3.3.9.3	Ré-entraînement sur Mac avec puce Apple Silicon et support MPS	28
3.4	Correspondance Stéréoscopique : Identification unique des personnes sur 2 flux vidéos	29
3.5	Localisation des personnes dans une salle	30
3.5.1	Localisation par unique caméra	30

3.5.2	Localisation 3D par 2 caméras : hypothèse de caméra non calibrée	32
3.5.3	Localisation 3D par 2 caméras : hypothèse de caméra calibrée	33
3.5.4	Implémentation de la méthode de localisation.	33
3.6	Calcul du plus court chemin pour diriger l'utilisateur	34
3.6.1	Introduction	34
3.6.2	Algorithme de recherche A*	34
3.6.3	Fonctionnement de base de l'algorithme A* :	34
3.6.4	Implémentation de l'algorithme du plus court chemin	35
3.6.5	Test de l'algorithme du plus court chemin	35
3.6.6	Conclusion	35
3.7	Application Android	36
3.7.1	L'IDE Android Studio	36
3.7.2	Architecture espérée de l'application	36
3.7.3	Difficultés rencontrées	38
3.7.4	Acquisition des flux vidéos	38
3.7.5	Traitement des informations d'aide à la décision	38
3.8	Refonte de l'architecture logicielle de l'application PC	38
4	Dialogue avec les pompiers	39
5	Tests unitaires	40
5.1	Test du code de captures video par l'ESP32 Cam	40
5.2	Test de notre circuit de développement pour l'ESP32 Cam	40
5.3	Test de la méthode de multi-detection de postures avec Yolov7 et OpenPose	40
5.4	Test de la réception du flux vidéo sur appareil Android	41
5.5	Amélioration du processus de sélection du réseau de l'ESP32-Cam	41
5.6	Test de la reception de plusieurs flux vidéos provenant de plusieurs ESP32 Cams	41
5.7	Test de la nouvelle méthode de multi-detection de postures avec Yolov7	41
5.8	Test de l'algorithme d'extraction des humains à partir des poses détectés par YoloV7	42
5.9	Test de l'obtention de mesures cinématiques sur la posture détectée	44
5.9.1	Test de l'estimation d'états le plus probable	44
5.10	Tests de fonctionnement de localisation 3D par 2 caméras.	44
6	Impact environnemental	47
6.1	Analyse de l'empreinte carbone du projet	47
6.2	Analyse des composants de la caméra	47
6.3	Co2 émis lors de l'utilisation du dispositif	47
7	Conclusion	49

Résumé

Le projet consiste à réaliser un système multi-caméra déployable dans un bâtiment permettant à des pompiers chefs de bataillon de coordonner facilement et efficacement une équipe en cas d'incident survenant dans ce bâtiment grâce à des fonctionnalités d'aide à la décision implémentées dans ce système. Ce dispositif pourra aussi déclencher l'alerte et informer automatiquement les pompiers.

L'observation d'une scène est faite à l'aide de plusieurs capteurs de vision (ESP32 CAM) installés sur le site. Elles envoient leur flux vidéo respectif en Wi-Fi sur un PC central installé dans le site qui se charge du traitement des images et de générer des indications pour assister la prise de décision (déclenchement d'alerte, liste de priorité d'interventions selon la gravité, etc...).

Ces résultats ainsi que les flux vidéos des caméras sont retransmises sur l'appareil Android du chef de bataillon pouvant alors visualiser en direct, sur toute la durée de l'incident jusqu'à la fin de l'intervention, les informations utiles pour coordonner son équipe.

Abstract

The project aims to develop a deployable multi-camera system within a building, enabling battalion chief firefighters to coordinate teams easily and efficiently in the event of an incident within the building. This is achieved through decision-support functionalities implemented in the system. The device is designed to trigger alerts and automatically inform firefighters.

Scene observation is conducted using multiple vision sensors (ESP32 CAM) installed on the site. These sensors transmit their respective video feeds via Wi-Fi to a central PC located on the site, responsible for image processing and generating decision-support results (alert triggering, prioritized intervention lists based on severity, etc.).

These results, along with the camera video feeds, are relayed to the Android device of the battalion chief. This allows real-time visualization of pertinent information throughout the duration of the incident until the completion of the intervention, facilitating coordination of the firefighting team.

Introduction

En France en 2023, plus de 300 000 incendies domestiques ont nécessité l'intervention des pompiers, selon le gouvernement. Une augmentation de 14 % par rapport à 2022 (263 000), soit un incendie domestique toutes les deux minutes.[4]

Les Incendies peuvent causer des dommages graves aux personnes, aux biens et à l'environnement. C'est un danger que nous souhaitons non seulement prévenir à l'avance, mais aussi réagir rapidement en termes d'intervention d'urgence. Il est donc nécessaire de définir un plan d'évacuation efficace en cas d'incendie pour sauver la vie des résidents, protéger les pompiers et minimiser les pertes de propriétés.

Le projet consiste à réaliser un système multi-caméra déployable dans un bâtiment permettant à des pompiers chefs de bataillon de coordonner facilement et efficacement une équipe en cas d'incident surveillant dans ce bâtiment grâce à des fonctionnalités d'aide à la décision implémentées dans ce système. Ce dispositif pourra aussi déclencher l'alerte et informer automatiquement les pompiers.

Ce projet rend donc service à trois parties prenantes qui sont les pompiers, les occupants des bâtiments qui se voient "protégés" par cette sécurité passive que nous mettons à disposition mais aussi du propriétaire du bâtiment pouvant proposer cette sécurité accrue dans son bâtiment.

Notre motivation principale a été de proposer un système d'assistance aux personnes en danger. La présence régulière des pompiers à proximité de l'ENSTA Bretagne nous permet aussi de réaliser un projet où nous sommes proche du client et des utilisateurs principaux(ici les pompiers).

1 État de l'art

Lee et al.[10] ont proposé une méthode de détection des victimes entourées de fumée dans une situation d'incendie dans un bâtiment en adoptant une méthode « 1-stage detector » et un modèle HHD (Hybrid Human Detection) combinant 2 algorithmes de détection : YOLOv3 et RetinaNet, et en se servant d'images CCTV. Ils ont essayé 3 approches : YOLO seul, RetinaNet seul et le HHD. Quatre étapes définissent la méthode HHD : Définition de zones candidates d'objets – Post-traitement – Redétection avec RetinaNet – Perte focale (focal loss). Le YOLO est performant quand le corps de la victime était complètement visible, alors que lorsque la concentration de fumée augmente et que le corps n'est plus clairement visible, l'accuracy de YOLO diminue. Le RetinaNet a une accuracy supérieure à celle de YOLO, mais beaucoup plus lent. De plus, certaines victimes n'ont pas été détectées. La méthode HHD proposée combine la rapidité de YOLO et l'accuracy de RetinaNet. Comme métrique d'évaluation, il se sont basés sur IoU (Intersection over Union), et ont essayé 3 valeurs : 0.3, 0.5 et 0.7 Pour une concentration de fumée de 70%, l'accuracy des trois méthodes, et pour les 3 valeurs du IoU était similaire, mais pour une concentration de 75% et plus, la méthode HHD était plus précise, avec une accuracy valant respectivement 72%, 72% et 49% pour IoU = 0.3, 0.5 et 0.7 La valeur de la métrique precision pour cette même méthode vaut 1 pour les 3 valeurs de IoU, et celles de recall valent 1.4, 1 et 0.9 pour les IoU = 0.3, 0.5 et 0.7 respectivement.

Jaradat et al.[12] ont proposé un réseau de neurones convolutif pour détecter les victimes dans les bâtiments en feu. La base de données est développée pour effectuer la détection d'individus en infrarouge. Le but est de classifier les images en entrées dans l'une des trois classes : « personnes », « animaux » ou « aucune victime ». Pour le prétraitement des images, ils ont adopté le CLAHE. Deux styles de CNN différents ont été implémentés : modèle CNN en une étape" et un modèle CNN en deux étapes en cascade. Les deux méthodes se sont révélées très performantes dans la détection d'humains et d'animaux ; cependant, la performance moyenne de 96,3% pour le modèle CNN en une étape a surperformé la moyenne 94,6% du modèle CNN en deux étapes.

Tsai et al.[14] ont proposé un modèle YOLOv4 DL avec l'imagerie thermique pour la détection de personnes dans des environnements où la fumée est dense. Trois bases de données ont été mises en jeu : FLIR ADAS, des images Kaggle AAU TIR, et des images thermiques à 360 degrés prises par Fluke Ti300+ représentent les différentes postures que peut prendre une personne : debout, assise, allongée et accroupie. Pour l'entraînement, 4000 epochs étaient suffisante pour entraîner le modèle sans overfitting. La détection en temps réel à 30,1 images par seconde est réalisée par le modèle. Comme métrique d'évaluation, ils se sont basés sur le IoU, qui donne une accuracy de 95% lorsqu'elle est supérieure à 50%. Ils ont obtenu une détection supérieure à 97% en termes de precision et de recall.

Zubairi et al.[18] ont développé un algorithme, MLOCATE, pour localiser et sauver les victimes enfermées dans un immeuble à plusieurs étages partiellement effondré, en précisant leur nombre et leur emplacement approximatif. Afin de déterminer l'emplacement approximatif de chaque victime, ils ont estimé la force du signal électromagnétique émis par leur téléphone portable. Après la localisation des victimes, les chemins prioritaires sont présentés sous forme visuelle. Leur idée consiste à fixer des « points repère », et de regrouper les victimes autour de ces points. En utilisant les distances des points repère par rapport à la base des équipes de secours, ainsi que le nombre de victimes par point de repère, un graphique pondéré est dessiné pour représenter tous les chemins possibles depuis la base jusqu'aux groupes de victimes présents dans le bâtiment. Le plus court chemin est ensuite calculé de la base à chaque groupe de victimes. Les priorités des étages sont calculées en tenant compte des poids attribués aux groupes de victimes présentes dans chaque étage.

2 Ingénierie-Système et Gestion de Projet

2.1 Ingénierie-Système

Afin de mener à bien notre projet, nous avons appliqué les méthodes abordées en ingénierie système.

2.1.1 Exigences du projet

Nous avons commencé par déterminer les exigences auxquelles notre système devra répondre pour être pertinent.

Pour cela nous nous sommes dans un premier temps mis dans la peau d'un pompier chef de bataillon et tenté d'imaginer les informations utiles dont il aurait besoin pour guider son bataillon. Cela a permis de dégager les premières fonctionnalités principales du dispositif qui sont :

- Déterminer en fonction de l'évolution des postures des personnes (debout, couché, état inerte...) si une personne est victime en déterminant un niveau de gravité.
- Déterminer l'état d'une pièce (accessibilité, présence de feu).
- Déclencher une alerte au niveau des pompiers si victime/danger grave détecté.
- Informer les pompiers en temps réel via le dispositif Android des victimes et transmettre une liste d'alertes temps réels ainsi qu'une liste des pièces où aller classées par ordre de priorité.
- Éventuellement transmettre des propositions de trajectoires pour faire les pompiers réaliser le plus court chemin vers les pièces où il faut intervenir en respectant les ordres de priorités pour une coordination des équipes toujours plus efficace.

Dans un second temps, ayant désormais une idée plus concrète du projet nous avons organisé une rencontre avec des pompiers directement à proximité du site de l'Ensta Bretagne pour leur exposer nos idées d'exigences et besoins auxquels notre dispositif devra répondre. Afin d'étoffer ces derniers nous avons réaliser un mini sondage dont les réponses nous ont surtout confirmé la pertinence de nos idées pour les fonctionnalités ainsi que pour l'ensemble du projet en lui-même.

Nous avons aussi réalisé des entretiens avec nos encadrants techniques afin d'également vérifier que notre direction s'alignait toujours avec leurs exigences projet et techniques.

Nous avons alors enfin abouti à ce tableau d'exigence.

ID	Description	Méthode de vérification
1.1	Le système doit être capable de détecter les individus.	Faire des tests avec le modèle sur différentes scènes
1.2	Le système doit identifier les blessés des pompiers dans la zone de secours.	Mettre en place un essai où les pompiers et un blessé sont positionnés dans une pièce, vérifier que l'identification fonctionne.

ID	Description	Méthode de vérification
1.3	Le système doit identifier l'état des victimes et les classer dans une la catégorie adaptée, debout, allongées ou inertes.	Mettre une personnes debout, une autre allongée au sol et une dernière qui s'est évanouie dans le champ de la même caméra et vérifier que chaque personnes est dans la bonne catégorie.
2.1	Le système doit localiser les pompiers et les victimes dans une zone de secours.	Mettre en place un essai où des victimes ou des pompiers sont positionnés dans une pièce, vérifier que les positions coïncident.
3.1	Le système doit être capable de générer une liste de priorité des lieux où il faut intervenir à partir de la classification des victimes.	Mettre plusieurs personnes debout dans une pièce sans danger et mettre une victime qui s'est évanouie dans une autre pièce et vérifier que la personne qui s'est évanouie est bien première dans la liste d'attente.
4.1	L'appareil android doit recevoir les flux vidéos des caméras.	Mettre en marche le système et s'assurer que l'appareil android reçoit bien les informations souhaitées.
4.2	Le système doit transmettre la liste des priorités d'interventions à l'appareil android.	Mise en marche du système et vérification que l'appareil reçoit bien la liste des priorités.
4.3	Les caméras doivent chacune envoyer un flux vidéo à un serveur central réalisant les tâches de traitement d'image.	Tentative de fonctionnement nominal et analyse des données reçues par le serveur.
5.1	Le système doit respecter les réglementations vidéos (droit à l'image).	Consultations des réglementations.
6.1	Les pompiers doivent pouvoir se connecter au système instantanément, par le clic d'un bouton sur l'application.	Essai logiciel, test du QPushButton.
6.2	L'application doit permettre à l'opérateur de changer de vue principale (caméra).	Test du QPushButton permettant de changer de vue.

TABLE 1: Exigences du système et méthodes de vérification.

2.1.2 Architecture externe

La liste des exigences ayant été rédigée nous pouvons passer à la réalisation de l'architecture externe du système. Pour cela nous réalisons un *use case diagram*, ce diagramme nous permet d'identifier 4 fonctions principales qui correspondent aux 4 étapes qu'il est nécessaire d'effectuer pour que le système fonctionne bien. (identifier, localiser, générer une consigne et transmettre)

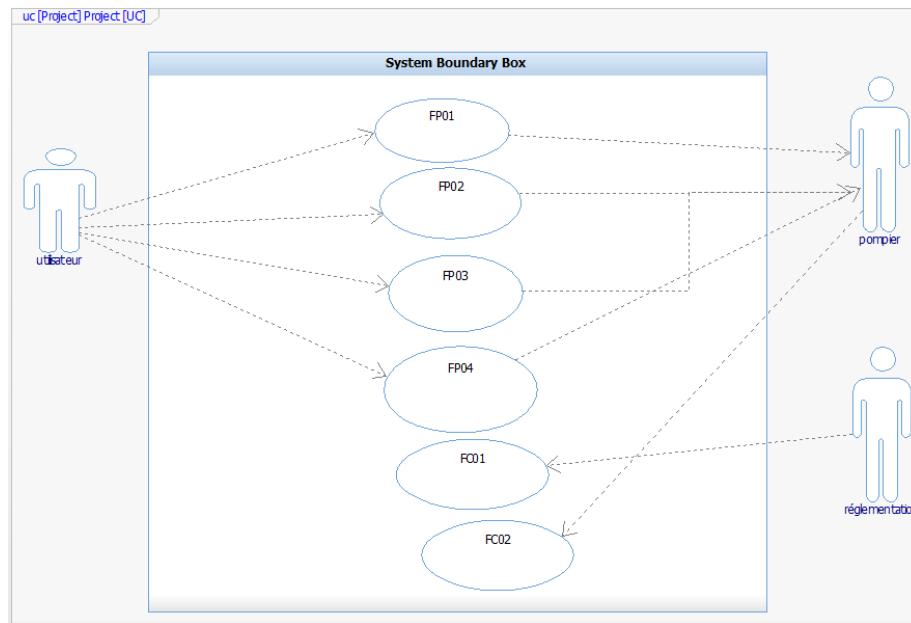


FIGURE 1 – Use Case Diagram

Les fonctions principales et contraintes que nous avons identifiées sont issues des exigences formulées dans le tableau (Figure 1). Afin de s’assurer que ses exigences sont bien prises en considération, il nous faut des critères. C’est pourquoi on réalise un tableau avec les fonctions principales et les fonctions contraintes imposées par l’environnement.

Fonction	Objectif	Critère	Niveau	Flexibilité	Exigences
FP01	Identifier	Distinction décor/cibles Distinction des pompiers/victimes Identification de l'état des victimes	Oui Oui Debout, allongé, inerte	Aucune Aucune Acune	1.1 1.2 1.3
FP02	Localiser victimes et secours	connaitre la position des personnes sur la carte du bâtiment	erreur < 2 mètres	Moyenne	2.1
FP03	Générer des consignes d'aides à la décision	générer une liste de priorité d'interventions en fonctions de l'état des victime	Oui	Aucune	3.2
FP04	Transmettre	transmettre les flux vidéos des caméras vers le dispositif Android transmettre la liste de priorité d'interventions en fonctions de la gravité transmettre les flux vidéos des caméras vers le serveur	Oui Oui Oui	Aucune Aucune Aucune	4.1 4.2 4.3
FC01	Respecter les RGPD	effacer les données non pertinentes	Oui	Aucune	5.1
FC02	connectivité au dispositif	Les pompiers doivent pouvoir se connecter facilement au système Les pompiers doivent pouvoir changer de caméra facilement	temps < 1 minutes bouton sur l'application	Moyenne Aucune	6.1 6.2

FIGURE 2 – Tableau des exigences

2.1.3 Exigences techniques du projet

En plus des exigences projet relatives aux parties prenantes, viennent s’ajouter les exigences techniques dont la plupart viennent de nos professeurs encadrants dans un but pratique et pédagogique.

De la part de nos professeurs encadrants, il nous a alors été exigé que le système utilise plusieurs caméras ESP32Cam pour observer une scène souhaitée, de traiter les flux vidéos venant de ces caméras et de rediriger

les données importantes vers un dispositif Android.

Pour le traitement des flux vidéos, on a choisi d'utiliser le langage Python car de nombreuses bibliothèques de traitement d'image et d'intelligence artificielle tels que PyTorch et TensorFlow existent pour ce langage. Enfin pour la réalisation de l'application Android, nous avons jugé adéquat d'utiliser l'IDE Android Studio et de programmer en langage Java, ces outils et langage étant très popularisés dans le monde du développement Android.

2.1.4 Architecture interne

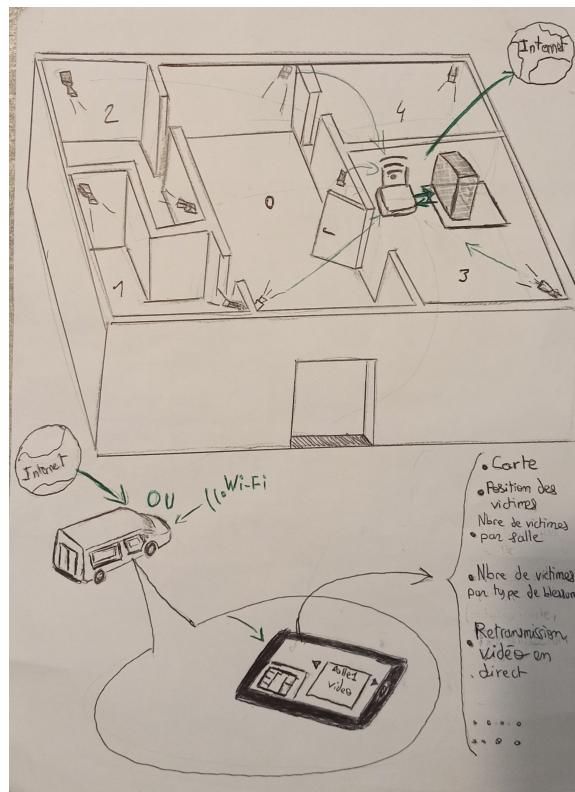


FIGURE 3 – Schéma de principe

Notre dispositif sera constitué d'un réseau de caméras faisant l'observation continue de la scène qui propageront leur flux vidéo sur un réseau Wi-Fi local au sein du bâtiment qui relayera ces flux à un PC central situé dans ce même bâtiment. Ce dernier est chargé du traitement d'images et de la génération d'indications d'aide à la décision. Il faut ensuite déployer un moyen qui permettra de communiquer les informations pertinentes au chef de bataillon.

Pour cela nous avons décidé de faire une application mobile qui tournera sur une tablette Android dont les pompiers sont propriétaires. L'idée retenue est de transmettre toutes les informations (vidéos, indications...)

sur une plateforme IOT en ligne afin que les pompiers puissent s'y connecter avec l'application depuis n'importe où.

2.2 Gestion de projet

2.2.1 Méthode Agile

La méthode Agile, une approche de gestion de projet dans le domaine de l'ingénierie logicielle, se distingue par son engagement envers la souplesse, la collaboration humaine et la communication continue. Fondée sur des pratiques telles que la méthodologie Scrum, elle favorise une progression pas-à-pas, mettant l'accent sur la participation active du client tout au long du développement du produit. L'agilité implique une gestion adaptative, un travail en équipes auto-organisées et pluridisciplinaires, ainsi que des tests récurrents pour détecter et rectifier les erreurs rapidement. En privilégiant la planification adaptive, la livraison précoce et l'amélioration continue, la méthode Agile encourage des réponses flexibles au changement, faisant de la collaboration entre les parties prenantes une priorité cruciale dans la réussite des projets.

Dans le cadre ce projet, l'utilisation de la méthode Agile est particulièrement adaptée dans la mesure où elle simplifie le suivi du projet. De plus, c'est une méthode très utilisé dans les entreprises, il est alors important de saisir cette opportunité pour la maîtriser.

2.2.2 Diagramme de Gantt

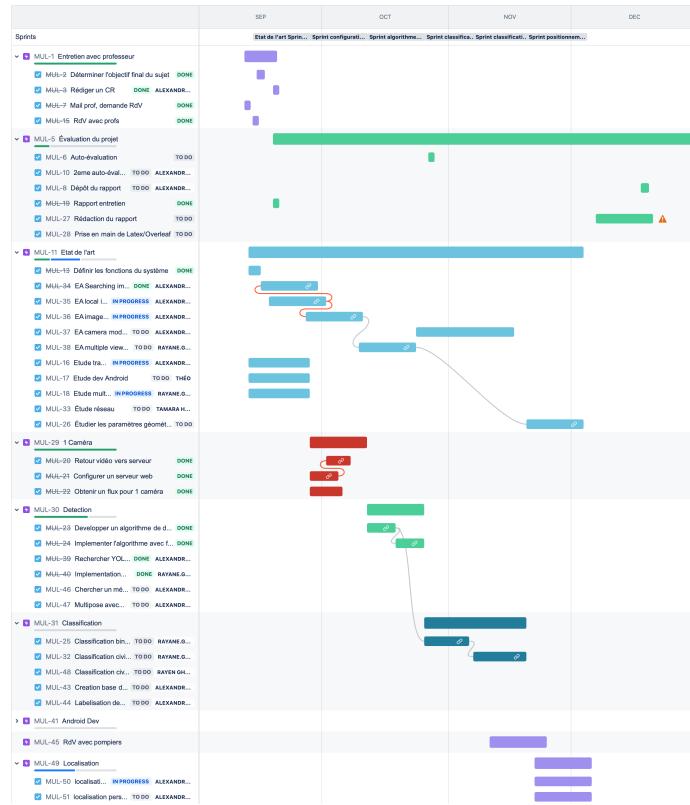


FIGURE 4 – Diagramme de Gantt, semestre 3

Lors des premières séances du projet, il nous a été demandé de réaliser un diagramme de Gantt pour planifier notre avancement sur le projet. Cependant, nous ne comprenions pas l'utilité de ce diagramme, car nous avions réparti les tâches entre chaque membre du groupe pour travailler de manière autonome. Au cours du semestre, lorsque nous avons dû commencer à faire interagir nos différentes parties entre elles. Nous avons compris comment ce diagramme nous permettait de planifier des objectifs à l'avance et ainsi de travailler efficacement en équipe.

L'un des piliers de la réussite du développement d'un projet informatique est la précision et la bonne organisation. En tant qu'étudiants ingénieurs, nous sommes censés avoir un esprit de collaboration et de rigueur scientifique, afin de faciliter le partage de l'avancement de chacun des membres de l'équipe. Pour ce faire, nous avons choisi deux outils, bien reconnus dans le domaine informatique ainsi que dans la gestion de projet : Git et Jira

2.2.3 Mise en place du Git et de Jira

Git est un logiciel de gestion de version et de collaboration bien connue des développeurs logiciels. Elle permet, entre autres de produire du code en autonomie avant de le faire interagir avec le travail réalisé

par les autres développeurs. Nous avons donc migré notre projet sur cette plateforme afin de bénéficier de ces avantages. Cela n'a pas été une mince affaire en raison de la complexité de notre vaste arborescence de fichiers. Nous avons tout de même consacré le temps et les efforts nécessaires pour faire fonctionner notre projet avec Git, notamment en adaptant et configurant nos IDE. Par exemple, pour le développement des ESP32Cams, nous avons choisi de migrer sur VSCode en installant le plugin Platform.io plutôt que de continuer à utiliser Arduino IDE. Ces efforts sont maintenant récompensés aujourd'hui par les bénéfices de Git que nous expérimentons.

Jira est une plateforme de travail collaboratif dédié au développement en mode Agile. Cette plateforme nous permet d'établir un diagramme de Gantt contenant les différents sprints de travail. Elle intègre de même un backlog product pour assigner les tâches à réaliser à chacun des partenaires. Cette plateforme nous a été fortement utile au début du projet pour établir la chronologie du travail mais elle nécessite un approfondissement et un investissement important pour en tirer pleinement avantage.

3 Développement du dispositif

3.1 Réseau Wi-Fi local

Conformément à l'architecture interne présentée plus haut il est nécessaire de disposer d'un réseau Wi-Fi local pour permettre la communication entre les divers composants (caméra, ordinateur central, etc). Pour nos tests nous avons commencé à utiliser la fonctionnalité Modem (Wi-Fi Hotspot) de nos smartphones.

Cependant afin d'avoir un réseau local disponible à tout moment au sein de tout l'établissement et de s'éviter la contrainte d'utiliser nos smartphones, nous utilisons aussi un réseau local Wi-Fi mis à disposition dans l'établissement pour les projets IOT. En cas de problèmes avec ce réseau nous pouvons à tout moment réutiliser nos smartphones si besoin.

3.2 Caméras ESP-32 Cam

Afin de réaliser l'observation temps réel de la scène en permanence, il nous faut des dispositifs peu encombrant, légers et faciles à installer permettant à la fois de capturer des images mais aussi de les envoyer en direct sur l'ordinateur central de traitement.

3.2.1 Choix du matériel et caractéristiques

Conformément aux exigences techniques des professeurs, nous nous sommes rabattus sur les caméras ESP32-Cam pour réaliser cela. mais aussi un module Wi-Fi. En plus d'embarquer un petit processeur, les ESP32-Cam sont donc de véritables petites webcams toutes-en-un largement adaptées pour notre projet.

ESP32-CAM est une carte de développement basée sur ESP32 à faible coût avec caméra intégrée, de petite taille. Ces composants embarqués sont caractérisés principalement par un processeur 32 bits basse consommation, qui peut également servir le processeur d'application, un module Wi-Fi, et donc un téléchargement d'images Wi-Fi, ainsi par un module UART avec un baudrate 115200 permettant un téléversement de et vers la caméra, ce qui la rend une solution idéale et conforme avec nos objectifs.

3.2.2 Programmation des cartes ESP32-CAM

Les cartes ESP32-Cam peuvent se programmer en Micro-Python ou en langage C/C++ avec les mêmes méthodes compatibles Arduino. Nous avons opté pour cette deuxième option, afin d'avoir un contrôle assez précis de ce que fait le programme , assurer une bonne vitesse d'exécution et afin de bénéficier de nombreux exemples de démarrage écrits en langage C++.

Nous avons initialement commencé à coder sur Arduino IDE ; et nous avons pu téléverser notre code sur les caméras. Cependant, cet IDE ne supporte pas le GIT, et le partage du code entre les membres du groupes est devenu donc difficile.

Une solution plus pratique était d'adopter un autre IDE : Platform.io à travers VSCode ; une cross-plataforme dédiée à l'écriture d'applications pour les modules embarqués.

Platform.io nous a permis de profiter de l'ensemble des outils de développement que propose VSCode (notamment la prise en charge des commandes Git, non disponibles sur l'IDE Arduino).

Les cartes ESP32-Cam nécessitent un protocole précis à suivre afin d'y charger un programme (il faut connecter un programmeur externe, connecter deux broches et appuyer sur le bouton RESET au bon moment). Bien que cela semble être de l'ordre du détail, cela est assez fastidieux sur le long terme et nous faisait perdre beaucoup de temps lors de nos premiers chargement de programme.

3.2.3 Mise en place d'un circuit électronique de développement pour les ESP32-CAM

Ainsi, afin d'optimiser le temps de développement et les tests nous avons pris l'initiative de développer un petit circuit de développement nous permettant à la fois, de simplifier les manipulations pour charger un programme et de changer facilement le réseau WI-FI auquel la carte se connectera sans qu'il soit nécessaire de modifier le code à nouveau.

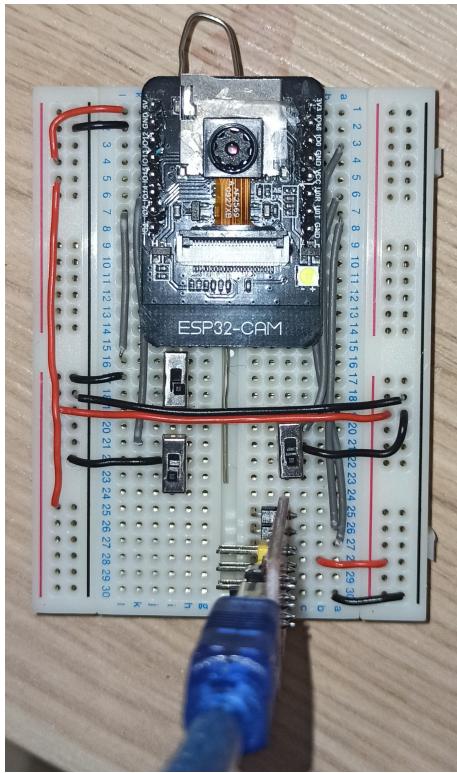


FIGURE 5 – Circuit électronique de développement

Le choix d'un réseau Wi-Fi se fait à l'aide des deux boutons switch disposés à la verticale sur la partie gauche, ce qui permet $2^2 = 4$ combinaisons, soit le nombre maximum de choix de réseaux Wi-Fi.

Le bouton sur la partie droite permet de basculer la carte sur le mode de chargement et permettre la mise à jour du logiciel embarqué.

Bien entendu, cette carte reste dédiée uniquement au développement et ne représente pas les caméras dans leur produit final.

3.2.4 Programme embarqué de capture et diffusion d'image sur flux

Dans ce programme est donc enregistré une liste des 4 réseaux Wi-Fi que nous utilisons le plus fréquemment pour nos tests (réseau de l'école, smartphones), ainsi que du système qui permet de basculer de réseau selon la configuration actuelle des boutons.

Le reste du programme se charge ensuite de configurer et initialiser la caméra ainsi que le serveur Web de retransmission pour la transmission Wi-Fi des images, puis rentre dans une boucle infinie pour capturer les images et les transmettre sur le serveur Web ainsi créé.

3.2.5 Prise en charge de plusieurs flux vidéos d'ESP32 Cams en simultané sur le logiciel PC

Dans un premier temps, il a fallu corriger la façon dont les flux vidéo des caméras étaient récupérés pour rendre le programme compatible avec les flux vidéos de nos ESP32 Cams. Le programme fonctionnait pour

des caméras pour lesquelles il n'étaient pas nécessaire d'envoyer une requête à chaque fois que l'on souhaite récupérer l'image suivante, ce qui n'est pas notre cas. En effet notre logiciel embarqué sur l'ESP32 Cam est tel qu'il faut envoyer une requête si on veut récupérer la dernière image capturée.

Nous avons également rendu possible l'exécution en continu du programme indépendamment de la disponibilité des caméras répertoriées : si plusieurs caméras se déconnectent pour diverses raisons (batterie faible, etc...) le programme continue de tourner ; ce qui n'était pas possible avec la version d'exemple. Les caméras déconnectées peuvent également se reconnecter et continuer à retransmettre leur flux.

3.3 Détection des personnes et pose estimation

Différentes technologies ont apparues pour permettre la détection de personnes dans une image, ou la pose estimation. La détection de personnes sur une image repose sur des algorithmes de Machine Learning et Deep Learning. Pour choisir l'approche la plus convenable pour notre projet, nous avons testé 3 algorithmes de détection : Haar Cascade, OpenPose et YOLO

3.3.1 Haar Cascade

Même après plus de deux décennies de son invention, Haar Cascade [7] reste l'un des piliers de la détection d'objets, parce qu'il est léger principalement en termes de ressources informatiques et d'utilisation de la mémoire. Que ça soit pour la reconnaissance de visage ou la détection d'objet en temps réel, Haar Cascade a révolutionné la détection d'objet. Les Haar features sont extraites de zones rectangulaires dans une image. La valeur de la fonctionnalité est basée sur les intensités des pixels. Habituellement, il est calculé à l'aide d'une fenêtre glissante et la zone à l'intérieur de la fenêtre est divisée en deux zones rectangulaires ou plus. Les Haar features sont la différence dans la somme des intensités de pixels entre ces zones.

Généralement, les zones rectangulaires sont parallèles aux bords de l'image plutôt qu'inclinées. Cependant, nous pouvons utiliser plusieurs tailles et formes de rectangles pour capturer différentes caractéristiques et variations d'échelle d'un objet. L'idée principale de cet algorithme est de conserver juste un petit nombre de pixels en relation avec l'objet à détecter, au lieu de retenir tous les pixels de la totalité de l'image, parmi lesquels un grand nombre s'avère non pertinent à l'objet concerné. Pendant le processus de détection, le Haar Cascade scanne l'image à différentes échelles et à différents emplacements pour éliminer les régions non pertinentes.

3.3.2 OpenPose

OpenPose [13] est un système open source pour l'estimation de pose humaine 2D de plusieurs personnes. L'estimation de la pose humaine est la tâche de vision par ordinateur consistant à localiser les articulations/parties du corps humain dans une image/vidéo (par exemple, le nez, l'épaule gauche, le coude droit).

Ces articulations sont également appelées points clés ou repères et sont représentées par l'emplacement 2D de chaque articulation dans l'image. La sortie du système est un ensemble de coordonnées (x, y) représentant les points clés, ainsi que la confiance du modèle dans chaque point clé. Le pipeline du modèle est le suivant :

1. Génération de caractéristiques : prend une image RGB en entrée et génère un ensemble de cartes de caractéristiques (F) (Feature Maps).
2. CNN à plusieurs étapes : obtient F de l'étape précédente en entrée et en prédit conjointement :

- (a) Part Confidence Maps
 - (b) Part Affinity Fields (PAFs)
3. Analyse : effectue un ensemble de correspondances bipartites pour associer les parties du corps candidates à partir des résultats de l'étape précédente.
 4. Assemblage : assemble les résultats dans des poses du corps entier pour toutes les personnes présentes sur l'image

3.3.3 YOLO

You only look once(YOLO) [8] est un système de détection d'objets en temps réel de pointe. Cet algorithme est si rapide qu'il est devenu le moyen standard de détection d'objets dans le domaine de vision par ordinateur

L'algorithme fonctionne sur la base des quatre approches suivantes :

1. Blocs résiduels (Residual blocks)
2. Régression de la boîte englobante (Bounding box regression)
3. Intersection Over Unions ou IOU
4. Suppression non maximale (Non-Maximum Suppression)

1-Blocs résiduels Cette première étape commence par diviser l'image originale en cellules de grille NxN de forme égale. Chaque cellule de la grille est chargée de localiser et de prédire la classe de l'objet qu'elle couvre, ainsi que la valeur de probabilité de correspondance.

2-Régression de la boîte englobante (Bounding box regression) La régression du cadre englobant est une technique utilisée dans les tâches de détection d'objets pour prédire les coordonnées d'un cadre englobant qui entoure étroitement un objet d'intérêt dans une image.

YOLO détermine les attributs de ces cadres englobants à l'aide d'un seul module de régression au format suivant, où Y est la représentation vectorielle finale de chaque cadre englobant.

$$Y = [pc, bx, by, bh, bw, c1, \dots, cn]$$

Ceci est particulièrement important pendant la phase de formation du modèle.

- **pc** correspond au score de probabilité de la grille contenant un objet.
- **bx, by** sont les coordonnées x et y du centre du cadre de délimitation par rapport à la cellule de la grille enveloppante.
- **bh, bw** correspondent à la hauteur et à la largeur de la boîte englobante par rapport à la cellule de la grille enveloppante.
- **c1,...cn** correspondent aux différentes classes

3-Intersection Over Unions ou IOU L'intersection sur l'union, communément appelée IOU, est une métrique utilisée pour évaluer le chevauchement entre deux cadres de délimitation ou régions d'intérêt. Il quantifie la similarité ou l'accord entre la boîte englobante prédictive et la boîte englobante de vérité terrain. IOU est calculé comme le rapport entre la zone d'intersection et la zone d'union des deux cadres englobants. Il est souvent utilisé comme critère pour évaluer les performances des algorithmes de détection d'objets, où une IOU plus élevée indique une meilleure précision de détection.

4-Suppression non maximale (Non-Maximum Suppression) Définir un seuil pour l'IOU n'est pas toujours suffisant car un objet peut avoir plusieurs cases avec IOU au-delà du seuil, et laisser toutes ces cases peut inclure du bruit. C'est ici que nous pouvons utiliser NMS pour conserver uniquement les cases avec la probabilité de détection la plus élevée.

Les itérations successives de YOLO ont permis d'obtenir un algorithme de plus en plus performant en réduisant le temps nécessaire pour réaliser une inférence (une estimation de la nature d'un objet nouvellement présenté).

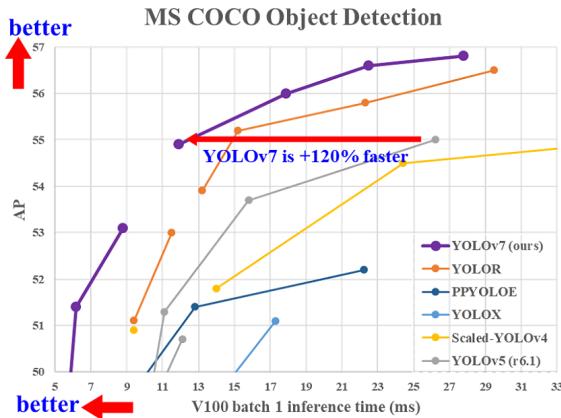


FIGURE 6 – Performance de YOLOv7 par rapport aux générations précédentes

Ce temps d'inférence dépend de la machine utilisée ainsi que du type de hardware utilisé. L'utilisation de puissant GPU permet de réduire considérablement le temps d'inférence par rapport à l'utilisation de CPU. La pose estimation correspond à l'estimation de la position du corps d'une personne au sein d'un environnement. YOLO permet à travers l'utilisation d'un réseau de neurones dédié d'effectuer la pose estimation de plusieurs personnes sur une image. Ces technologies ont pu être déployées dans un contexte proche du nôtre, pour la détection de feu de forêt. Ces technologies se retrouvent donc de plus en plus cruciales pour agir dans des situations où l'homme ne peut se rendre ou là où il n'est pas présent. C'est dans ce contexte que nous avons choisi d'étudier l'implémentation d'algorithme de détection et de pose estimation pour répondre aux exigences de notre système.

Afin de satisfaire les exigences 1.1, 1.2 et 1.3, nous avons choisi d'utiliser un ordinateur central, qui après avoir reçu les flux vidéos des caméras est chargé de faire tout les traitements d'image nécessaires et transmettre les indications d'aide à la décision à l'appareil Android du chef pompier

Pour cela nous développons une application PC en Python répondant à ces besoins.

3.3.4 Première méthode d'implémentation de la multipose (Semestre 3)

Notre objectif est de faire de la reconnaissance d'activité pour chacun des individus que l'on détecte sur une scène afin d'établir une liste de priorité de visite des victimes en fonction de la gravité de l'état d'activité estimé.

Pour cela nous avons donc besoin de faire des observations précises sur les mouvements de l'individus.

Cela signifie donc d'observer non seulement l'évolutions des paramètres cinématiques de la boite englobante (bounding box) mais aussi les paramètres cinématiques de chaque membre ou articulations de l'individu détectés. Par paramètres cinématique nous entendons (mesure de position, vitesse et accélération).

Dans un premier temps (dans la première partie du projet au semestre 3), nous avons le modèle de YOLOv7 parmi les méthodes citées précédemment afin de réaliser la détection des boites englobantes des individus. Notre choix s'est construit sur des critères de rapidité et de robustesse : Tout en détectant une variété importante d'objets YOLOv7 est l'un des modèles permettant de traiter un nombre important d'images par secondes (environ 5 à 30 fps sur nos machines).

Pour assister efficacement les pompiers dans leur prise de décision, on estime qu'un temps de réponse inférieur à 1 seconde est largement suffisant pour détecter d'éventuels incidents et victimes. Ainsi avoir un traitement d'image fonctionnant à plus d'une image par seconde est suffisant.

COCO, le dataset utilisé pour entraîner YOLO contient 80 classes d'objets. La capacité de YOLO a détecter plusieurs objets sur une même image nous a aussi guidé dans notre choix. En effet dans certains contexte, notre système peut être amener à fonctionner dans un environnement (école, atelier d'usine, bureaux d'entreprise) comportant un nombre important de personnes mais aussi d'objets pouvant renforcer nos observations en donnant davantage de contexte à une situation ou activité en cours d'estimation...

Nous avions commencé au semestre 3 notre programme PC sur la base d'un projet "sample" de prise en main de Yolov7 disponible sur internet permettant d'acquérir plusieurs sources de flux vidéo en renseignant les adresses Ip des caméras connectées à notre réseau local et de réaliser de la détection d'objets sur ces images grâce à YOLOV7.

Cette première étape nous a servi à fonder les premières briques logicielles de notre solution.

Par la suite nous avons implémenté l'estimation de pose pour chaque humain détecté par YOLOV7 ce qui nous a permis par la même occasion de faire du multi-pose estimation (estimation de posture pour tous les humains visibles sur une scène). Cela s'est obtenu en effectuant un détourage de l'image là où YoloV7 détecte une personne, et d'y appliquer sur cette image détournée l'estimation de pose, puis de transformer les coordonnées des points articulaires locales sur cette région d'image sur l'image globale d'origine (simples changements de repères).

3.3.5 Deuxième méthode d'implémentation de la multipose (Semestre 4)

La première méthode d'estimation de pose posait des soucis de performance du à une utilisation non-optimisée d'OpenPose pour faire la posture sur chaque humain détourné à l'aide de YoloV7.

De plus cette ancienne méthode utilisant du détourage n'est pas très fiable dans les situations où deux personnes sont quasiment superposées.

Nous avons donc cette fois utilisé un autre modèle de poids Yolov7 permettant de faire directement l'estimation de pose.

Nous utilisons donc désormais uniquement Yolov7 pour ce traitement (voir les tests unitaires correspondants plus bas).

3.3.6 Obtention de la liste des humains présents

Afin de faire de la reconnaissance d'activité. Il est nécessaire de faire des observations qui sont propres à chacun des humains et qui sont mémorisés sur un certain nombre de capture d'images (frames). Nous faisons effectivement de l'estimation de posture mais à ce stade il n'y a pas d'associations de posture à un humain à proprement parlé au fil des frames. Or cet étape d'association est primordiale pour les objectifs d'observation cités.

Ainsi, nous avons imaginé un algorithme conçu pour extraire des représentations humaines à partir d'une scène capturée par une caméra, en utilisant des points clés détectés sur les personnes. L'objectif principal est d'associer une identité à chaque estimation de pose afin de distinguer les humains détectés dans une scène, en particulier en cas de chevauchement. Voici un résumé de son fonctionnement :

- Initialisation : L'algorithme initialise différentes listes pour stocker les humains dans la scène, y compris ceux qui sont visibles, cachés et en train de sortir. Il prend également en compte les zones de sortie de la scène ainsi que les limites de la vue de la caméra.
- Mise à jour des humains : Cette fonction reçoit les poses des humains détectées et les mises à jour du cadre actuel. Elle commence par vérifier s'il existe déjà des humains détectés dans la scène. Si ce n'est pas le cas, un nouvel objet humain est créé pour chaque pose détectée.
- Association des poses aux humains existants : Pour chaque pose détectée, l'algorithme cherche à associer la pose au meilleur humain correspondant en fonction d'un score de correspondance. Il ne considère que les poses dont le score dépasse un certain seuil. Les poses associées sont retirées de la liste des poses disponibles.
- Gestion des nouveaux humains et des humains cachés : Les poses non associées à des humains existants sont considérées comme appartenant à de nouveaux humains dans la scène. Les humains qui ne sont pas détectés dans cette étape mais qui étaient présents dans la scène précédente sont considérés comme cachés.
- Suppression des humains sortant de la scène : Les humains qui sortent de la scène, définis par leur position par rapport aux limites de la vue de la caméra, sont retirés de la liste des humains.
- Calcul des scores de correspondance : Les scores de correspondance entre les poses et les humains existants sont calculés en fonction de plusieurs critères, notamment la distance euclidienne entre les keypoints, la corrélation entre les keypoints, la distance entre les centres des poses et d'autres facteurs comme les histogrammes de couleur.

En résumé, cet algorithme permet de suivre les humains dans une scène en associant les poses détectées aux humains existants et en ajoutant de nouveaux humains au besoin, tout en gérant les cas où les humains sortent de la vue de la caméra ou deviennent temporairement invisibles.

3.3.7 Première méthode de reconnaissance d'état d'activité

Cet algorithme vise à classifier les poses humaines détectées en associant une posture connue à chaque pose, en utilisant une approche symbolique. Voici un résumé plus détaillé de son fonctionnement :

1. **Initialisation des structures de données** : L'algorithme initialise diverses structures de données pour stocker les informations sur les poses, telles que les mesures, les caractéristiques et les seuils de classification. Ces structures sont utilisées pour évaluer et classifier les poses ultérieurement.

2. **Calcul des caractéristiques des poses** : Pour chaque pose humaine détectée, l'algorithme calcule plusieurs caractéristiques, notamment le rapport largeur/hauteur du cadre englobant, la vitesse globale de déplacement, la vitesse des keypoints et d'autres caractéristiques pertinentes. Ces mesures sont essentielles pour déterminer la posture associée à chaque pose.

3. **Comparaison des caractéristiques avec les seuils prédéfinis** : Les caractéristiques calculées pour chaque pose sont comparées à une matrice de seuils prédéfinis. Cette matrice contient des plages de valeurs pour chaque caractéristique, déterminant ainsi les critères de classification pour chaque posture.

4. **Classification symbolique des poses** : En utilisant les valeurs des caractéristiques et les seuils prédéfinis, l'algorithme classe chaque pose dans une posture prédéfinie. Il effectue cette classification en comparant les valeurs de chaque caractéristique avec les plages de valeurs spécifiées dans la matrice de seuils, attribuant ainsi une étiquette de posture à chaque pose.

5. **Analyse des résultats de classification** : En plus de classifier les poses, l'algorithme fournit également des informations supplémentaires telles que la distance euclidienne entre les keypoints, la corrélation entre les keypoints, etc. Ces informations aident à analyser les résultats de classification et à comprendre la qualité de la classification pour chaque pose.

En résumé, cet algorithme utilise une approche symbolique pour classifier les poses humaines détectées en évaluant les caractéristiques de chaque pose par rapport à des seuils prédéfinis, puis en attribuant une posture prédéfinie à chaque pose en fonction de ces évaluations.

3.3.8 Deuxième méthode de reconnaissance d'état d'activité basée sur un modèle de markov caché

Bien que les états d'activités humaines peuvent être déduites d'observations cinématiques, les états d'activités pour un humain donné s'enchaînent en suivant des séquences probables. Par exemple, une observation donnée peut donner lieu à prédire deux états possibles (concurrents). Pour trancher entre ces deux états l'idée est donc de savoir l'état le plus probable compte tenu des anciens états cachés. L'idée est donc de s'appuyer sur la notion de séquence d'état probable, d'où l'utilisation d'un modèle de Markov caché.

Ce modèle sera alimenté par les observations cinématiques de l'individu, projetées dans le plan de la caméra. Ces observations sont obtenues grâce à une analyse préalable d'poseestimation obtenue à chaque image.

Plus précisément, nous utiliserons la combinaison de deux variables d'observations : la vitesse moyenne des membres de l'individu, discrétisée sur 3 niveaux, et l'accélération du centre de gravité de l'individu,

discrétisée sur 2 niveaux.

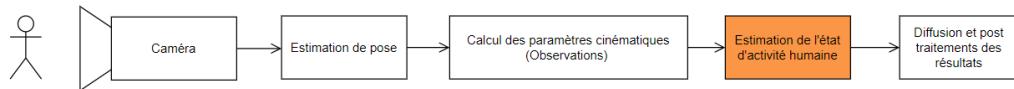


FIGURE 7 – Chaine de traitement d’image. En orange figure la partie traitée dans ce projet

Dans un premier temps, il s'agit de définir une représentation de notre problème sous une chaîne de Markov simple $\{\pi, A\}$ avec π le vecteur d'état initial et A la matrice de transition.

3.3.8.1 Définition des états L'objectif à terme est de surveiller le comportement d'un opérateur pour détecter s'il se trouve dans un état qui nécessite ou non, une alerte ou une attention particulière et retracer l'ensemble des comportements qu'il a traversé pour un instant donné.

Les transitions entre chaque activités d'états se fera toutes les deux secondes soit donc pendant $2 \times \text{FPS}$ frames. (Attention ceci impacte fortement la distribution des probabilités)

Pour des raisons de simplicité on définit les états d'activités suivants :

Cette approche permet d'éviter toute fusion indésirable d'états pour garantir la distinction entre les différents états.

L'état de panique étant propre à chaque individus, on considérera pour la construction du graphe d'état (et la matrice d'état associée), des probabilités de passage "moyennées"

Pour la suite, on considérera la correspondance suivante entre les numéros et les états mentionnés :

Id	Etats
E0	Conditions normales
E1	Handicapé
E2	Inconscient non handicapé
E3	Inconscient handicapé
E4	Panique non handicapé
E5	Panique handicapé

TABLE 2 – Liste des états avec leur numéro associé

3.3.8.2 Graphe des états Le graphe des états ci dessous retrace les probabilités de transition entre les états :

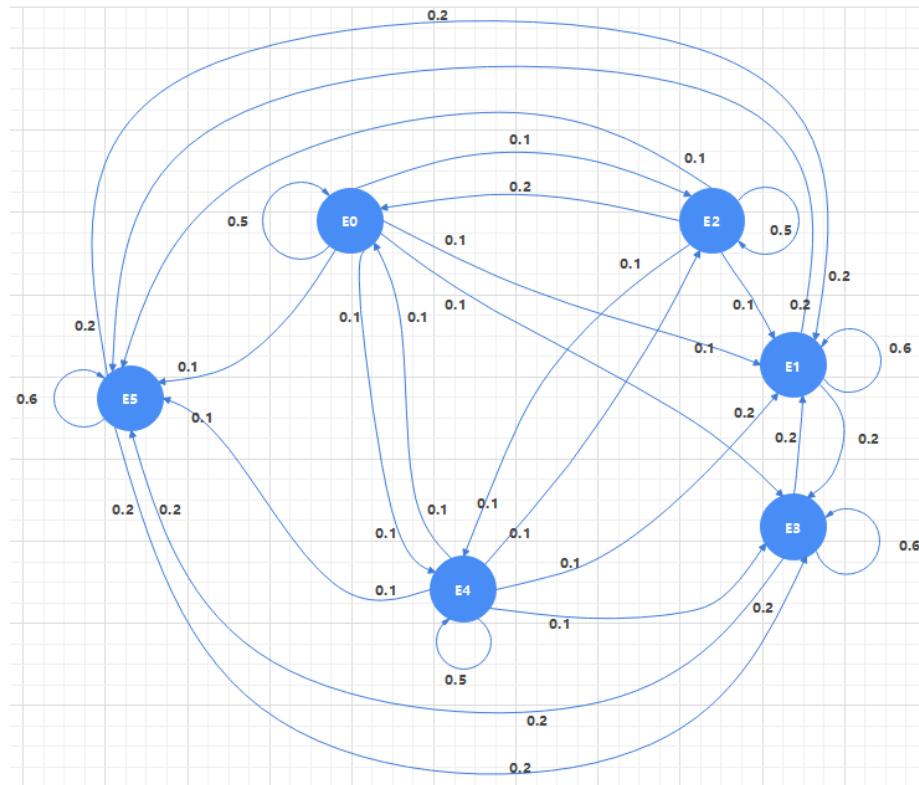


FIGURE 8 – Graphes des états

3.3.8.3 Matrice de transition A partir du graphe précédent on peut construire la matrice de transition (Les lignes et colonnes sont bien sur respectives à la numérotation des états).

$$A = \begin{pmatrix} 0.5 & 0.1 & 0.1 & 0.1 & 0.1 & 0.1 \\ 0 & 0.6 & 0 & 0.2 & 0 & 0.2 \\ 0.2 & 0.1 & 0.5 & 0 & 0.1 & 0.1 \\ 0 & 0.2 & 0 & 0.6 & 0 & 0.2 \\ 0.1 & 0.1 & 0.1 & 0.1 & 0.5 & 0.1 \\ 0 & 0.2 & 0 & 0.2 & 0 & 0.6 \end{pmatrix}$$

On vérifie bien que la somme des coefficients de chaque ligne vaut bien 1.

Pour prédire les états d'activité humaine inconnus, nous introduisons les variables d'observation. Ces variables, organisées en séquences, permettent au modèle de Markov caché de faire des prédictions sur les états. Le problème est modélisé par une chaîne de Markov cachée définie par le système π, A, B , où il est nécessaire de définir B comme la matrice d'observation.

3.3.8.4 Définition des observations

Nous nous baserons sur deux variables d'observations :

- La moyenne des vitesses en valeur absolue des articulations et du centre de gravité sur 3 niveaux (faible, moyenne, haut) : cette caractéristique à tendance à décrire l'état inerte ou non de l'individu
- L'accélération en valeur absolue du centre de gravité sur 3 niveaux (faible, moyenne, haut) : cette caractéristique à tendance à décrire si l'individu a subi un choc ou effectue des gestes brusques.

Nous allons donc considérer pour la suite une seule variable d'observation qui sera la combinaison de ces deux variables ce qui aboutit à $3^3 = 9$ valeurs possibles.

Id	Moyenne des vitesses articulaires	Acceleration maximale
O0	faible	faible
O1	faible	moyenne
O2	faible	forte
O3	moyenne	faible
O4	moyenne	moyenne
O5	moyenne	forte
O6	forte	faible
O7	forte	moyenne
O8	forte	forte

TABLE 3 – Liste des observations avec leur numéro associé

3.3.8.5 Matrices d'observation

On considère donc la matrice d'observation associant le couple vitesse/acceleration avec les probabilités d'états pour chaque lignes.

$$B = \begin{pmatrix} 0.9 & 0.1 & 0 & 0 & 0 & 0 \\ 0.6 & 0.3 & 0.1 & 0 & 0 & 0 \\ 0.3 & 0.4 & 0.3 & 0 & 0 & 0 \\ 0.1 & 0.3 & 0.5 & 0 & 0 & 0 \\ 0 & 0.1 & 0.6 & 0 & 0.3 & 0 \\ 0 & 0 & 0.1 & 0 & 0.4 & 0.5 \\ 0.1 & 0 & 0 & 0.3 & 0.6 & 0 \\ 0 & 0 & 0 & 0.1 & 0.3 & 0.6 \\ 0 & 0 & 0 & 0 & 0.1 & 0.9 \end{pmatrix}$$

3.3.8.6 Estimation de séquences d'états par l'algorithme de Viterbi L'algorithme de Viterbi est un algorithme de programmation dynamique utilisé pour trouver la séquence la plus probable d'états cachés dans un modèle de Markov caché, en utilisant des observations.

3.3.9 Ré-entraînement de YOLOv7.

Afin de permettre la classification entre pompiers et civils, un ré-entraînement de YOLO à été étudié. Cet entraînement nécessite notamment la mise en place d'un jeu de données d'entraînement, de test et de validation. De plus afin de rendre l'entraînement rapide, plusieurs approches ont été envisagées : sur nos machines personnels (PC Windows avec support de CUDA, Mac avec puce Apple Silicon) ou sur les calculateurs de l'ENSTA Bretagne.

3.3.9.1 Constitution du jeu de données. Afin de distinguer civils et pompiers, nous avons accordées de l'importance à ce que les images constituants le jeu de données représentent des pompiers et des civils. Afin de ne pas obtenir un résultat trop restrictif dans sa capacité de détection, nous avons veiller à ce que les images représentent les pompiers et services de secours dans leurs différentes tenues :



FIGURE 9 – Pompiers en tenue de caserne



FIGURE 10 – Pompiers médecin en tenue d'intervention



FIGURE 11 – Pompiers en tenue d’intervention

De plus, afin de s’assurer que YOLO fonctionne de manière optimale dans les conditions d’utilisations prévus pour le système, des images présentant les services de secours en contre-champ, vue de haut, en basse luminosité, et entourés de flammes, ont été ajoutées au jeu de données.



FIGURE 12 – Pompiers perçus en basse luminosité et entourés de flammes.



FIGURE 13 – Service de secours vue de haut.

Ces détails nous permettent de renforcer la robustesse de YOLO vis-à-vis des conditions dans lesquelles auront lieu la détection, conditions parfois difficiles pour la détection des secours ou des civils. Nous avons appliquer les mêmes règles pour la représentation des civils dans notre jeu de données.



FIGURE 14 – Personnels civils vue de haut

Pour procéder à l'étiquetage des données de manière efficace, nous avons premièrement utilisé YOLO non-entraîné pour réaliser simplement la détection de personnes dans notre jeu de données. Nous avons ensuite avec l'aide de l'application Roboflow, pu changer les labels des bounding-box de notre dataset, pour créer deux classes, "Pompiers" et civils.

Avant de procéder au ré-entraînement, nous avons ajouté un autre type de photographies dans notre jeu de données. Afin de faciliter la démonstration de notre système lors de sa présentation à l'ENSTA Bretagne, nous avons fait le choix de porter des gilets de visibilité fluorescents afin d'incarner des pompiers. Nous avons alors ajouté des photos représentants des personnes vêtus de ce type de tenues, en suivant le même protocole que précédemment.



FIGURE 15 – Personnes en tenues de haute visibilité, pour incarner des secours

La séparation du jeu de donnée à été assurée par Roboflow avec un taux de 75% des données utilisées pour la base d'entraînement. La description du jeu de donnée est le suivant :

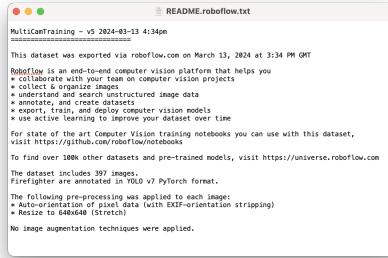


FIGURE 16 – Description du datset

Nous avons ensuite procéder au ré-entraînement.

3.3.9.2 Ré-entraînement sur PC Windows avec support de CUDA. CUDA (Compute Unified Device Architecture) est une plateforme de calcul parallèle développée par NVIDIA. Elle permet d'accélérer les calculs en exploitant la puissance de traitement des GPU. CUDA fournit un ensemble d'outils, de bibliothèques et de directives qui permettent aux développeurs de créer des applications hautement performantes en tirant parti de la puissance de calcul parallèle des GPU NVIDIA. supporté par PyTorch, CUDA permet d'accélérer l'entraînement des modèles PyTorch. Afin de préciser à YOLO la possibilité d'utiliser le GPU Nvidia pour l'entraînement, nous ajoutons la commande `device=torch.device("cuda")`

3.3.9.3 Ré-entraînement sur Mac avec puce Apple Silicon et support MPS. L'arrivée des puces Apple Silicon sur Mac permet de tirer l'avantage d'une grande puissance de calcul pour un faible coût énergétique. Dans cette objectif nous avons chercher à étudier la possibilité de ré-entraînement sur Mac, efficace énergiquement et rapidement Le ré-entraînement de YOLO peut être réalisé sur CPU ou sur GPU grâce au support GPU de PyTorch, bibliothèque Python sur laquelle se base YOLO. L'entraînement étant plus rapide sur GPU, nous avons chercher une méthode permettant d'utiliser le GPU d'un Mac avec une puce Apple Silicon M1 Pro. Après consultation de différents articles[6], nous avons mis en place un environnement virtuel sur la machine, contenant la version Nightly de PyTorch. Issue de la collaboration d'Apple et PyTorch, cette version permet de tirer parti des Metal Performance Shaders d'un Mac[1]. Ce support permet alors l'utilisation du GPU d'un Mac.

L'utilisation du GPU sur Mac a nécessité la modification du code de YOLO, notamment du fichier `train.py`, utilisé pour le ré-entraînement. Nous avons précisé le type de device torch, par la commande :
`device=torch.device("mps")`

Cependant nous avons rencontré plusieurs problèmes, notamment des problèmes de typages de variables dues aux différences de versions de PyTorch entre celles utilisées pour éditer YOLOv7 et PyTorch Nightly. Ces problèmes ont pu se résoudre relativement facilement. Le problème le plus important reste un problème de demande RAM. Sur notre Mac, équipé de 16Go de RAM, lors de la première epoch d'entraînement, YOLO affiche un Warning expliquant que la RAM allouée de 17,89 Go n'est pas suffisante. Une ligne de commande permet alors d'utiliser la mémoire disk : `PYTORCH_MPS_HIGH_WATERMARK_RATIO=0.0`

à noter que selon la documentation, cette commande peut provoquer l'arrêt de l'ordinateur. Après utilisation, nous constatons que la totalité de la RAM est occupée ainsi que 100Go de disk, mais le ré-entraînement refuse toujours de se lancer pour cause de mémoire insuffisante.

La version Nightly de PyTorch est une version bêta, toujours en phase de développement et présentant des instabilités. Pour cette raison nous avons choisi de nous tourner vers un ré-entraînement sur PC Windows avec support de CUDA.

3.4 Correspondance Stéréoscopique : Identification unique des personnes sur 2 flux vidéos.

Afin de localiser une personne dans une pièce à partir de deux flux vidéos, nous avons besoins d'effectuer une correspondance stéréoscopique. En effet dans le cas courant où plusieurs personnes sont présentes sur une même image, nous avons besoins d'identifier chaque personne de manière unique puis de faire une correspondance entre les coordonnées du sujet dans les deux flux vidéos. Sans cette correspondance, il serait impossible d'associer les coordonnées du sujet, une confusion pourrait alors avoir lieu dans le cas où plusieurs sujets seraient présents.

Afin de résoudre ce problème, nous avons dans un premier temps utilisé la méthode de Brute Force Matching implantée dans la bibliothèque Python OpenCV. Le Brute-Force Matching est une méthode simple mais efficace pour faire correspondre des descripteurs entre deux ensembles d'éléments, tels que des points d'intérêt dans des images[2]. Ainsi, nous déterminons dans un premier temps les descripteurs d'une image, dans notre cas les descripteurs SIFT[15]. Nous appliquons ensuite l'algorithme de Brute Force Matching implanté par OpenCv afin d'obtenir les correspondances entre les images issus de nos deux flux vidéos.

Une fois les correspondances établis, notre stratégie consiste à déterminer la correspondance la plus proche de notre sujet, par calcul d'une distance, pour évaluer sa correspondance dans l'autre image.



FIGURE 17 – Illustration de principe de la stratégie de correspondance stéréoscopique.

La figure ci-dessus illustre notre stratégie. Les ronds rouge représentent nos sujets que nous souhaitons localiser à partir de leurs observations sous 2 angles de vues. Les carrés verts représentent les descripteurs SIFT les plus proches pour chaque sujet, déterminés en recherchant la distance la plus faible (représentée en bleu) de chaque descripteur au sujet. Nous pouvons ainsi créer des paires de coordonnées des sujets dans les images des deux flux.

Le principe de localisation spatiale présenté ci-après repose sur cette construction de paire de coordonnées.

3.5 Localisation des personnes dans une salle.

Dans l'objectif de satisfaire l'exigence 2.1, (en plus de pouvoir localiser des personnes dans une vue), le système doit aussi pouvoir localiser une personne au sein d'une pièce (carte). Différentes approches ont été mises en place, sous plusieurs hypothèses : - Localisation 2D par une caméra : la position de la caméra dans le repère du bâtiment est parfaitement connue - Localisation 3D par deux caméras : avec des caméras supposées non calibrées. - Localisation 3D par deux caméras : avec des caméras supposées calibrées. Les algorithmes de localisation seront alimentées par les coordonnées des points d'intérêts fournis par un algorithme de pose-estimation.

3.5.1 Localisation par unique caméra

On s'intéresse dans cette section à la localisation d'une personne dans une pièce avec l'utilisation d'une unique caméra. La localisation se faisant avec une seule caméra.

On définit les repères :

$$R_0 = (\vec{a}_0, \vec{b}_0, \vec{z}_0), \text{ repère absolu.}$$

$$R_1 = (\vec{a}_1, \vec{b}_1, \vec{z}_0), \text{ repère de la caméra}$$

$$R_2 = (\vec{a}_2, \vec{b}_2, \vec{z}_0), \text{ repère de l'image}$$

On considère l'équation caméra permettant de relier un point 2D x présent sur le plan image de la caméra

dans le repère R_2 , au point 3D X dans le repère R_0 :

$$x = PX \quad (1)$$

où P est la matrice caméra, f est la distance focale de la caméra.

$$P = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2)$$

On notera que dans le cas de notre système, on cherche à déterminer X connaissant x . Il faut donc déterminer la pseudo-inverse de P .

Pour prendre en compte le fait que l'image et la caméra ne partagent pas le même repère, on prend en considération le centre optique de coordonnées (c_x, c_y) dans P :

$$P = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3)$$

De même, pour assurer l'expression de X dans R_0 , on détermine la rotation et la translation de la caméra par rapport à R_0 . P devient ainsi un produit matriciel entre la matrice K dépendant des paramètres intrinsèques de la caméra et $[R|t]$ dépendant des paramètres extrinsèques, en coordonnées homogènes.

$$P = K[R|t] \quad (4)$$

avec

$$K = \begin{pmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (5)$$

et

$$R = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{pmatrix} \begin{pmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{pmatrix} \begin{pmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (6)$$

On obtient ainsi les coordonnées de X en déterminant la pseudo inverse de Moore-Penrose de P :

$$X = P^+x \quad (7)$$

L'algorithme utilisant cette méthode correspond au fichier CameraCalibration.py . Des tests et des estimation d'erreur de cet algorithme seront entrepris au semestre prochain. A noter que l'utilisation d'une unique caméra impose, de placer la caméra au niveau du plafond de la pièce. Cela permet de localiser la personne par ses coordonnées x et y .

3.5.2 Localisation 3D par 2 caméras : hypothèse de caméra non calibrée

Afin de conserver la notion de profondeur, nous nous intéressons dans ce paragraphe à la géométrie épipolaire, appliquée à une observation par deux caméras. L'implantation de la géométrie épipolaire, nécessite la mise en place d'une phase d'initialisation des caméras.

Soient x et x' les points images respectivement des caméras n°1 et n°2 correspondant au même point réel X . Ces points vérifient l'équation de Longuet-Higgins :

$$x'^T F x = 0 \quad (8)$$

Où F est une matrice 3×3 , appelée matrice fondamentale.

Le calcul des coefficients de F se fait par application de l'algorithme des 8 points

Soit $M \geq 8$ et $m \in [1, M]$.

On considère les correspondances (x_m, x'_m) , chacune de ces correspondances vérifie l'équation de Longuet-Higgins soit :

$$\forall m \in [1, M], \quad x_m'^T F x_m = 0 \quad (9)$$

D'où en coordonnées homogènes :

$$(x'_m \quad y'_m \quad 1) \begin{pmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{pmatrix} \begin{pmatrix} x_m \\ y_m \\ 1 \end{pmatrix} = 0 \quad (10)$$

soit sous forme linéaire :

$$\begin{pmatrix} x_1 x'_1 & x_1 y'_1 & x_1 & y_1 x'_1 & y_1 y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots \\ x_m x'_m & x_m y'_m & x_m & y_m x'_m & y_m y'_m & y_m & x'_m & y'_m & 1 \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{pmatrix} = 0 \quad (11)$$

où la matrice ($M \times 9$) est noté A .

Connaissant F , on cherche à appliquer l'équation caméra afin de déterminer X .

On se place dans l'hypothèse où la caméra n'est pas calibrée et on suppose que la position d'une des caméras correspond à l'origine d'un repère par rapport à la seconde, i.e sa matrice caméra est $P = [I|0]$. La seconde matrice caméra est obtenue par :

$$P = [Ad(e)F|e'] \quad (12)$$

où e et e' sont les éipoles associés respectivement aux caméras.

Les éipoles correspondent aux projections des centres des caméras dans le plan image de l'autre caméra
Les éipoles vérifient :

$$Fe = 0 \quad \text{et} \quad e'^T F = 0 \quad (13)$$

Connaissant P , P' on peut alors trianguler le point 3D X à partir de 2 observations de ce point x et x' en utilisant l'équation caméra. On obtient les équations :

$$x = \alpha PX \quad \text{soit} \quad \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \alpha \begin{pmatrix} -p_1^T - \\ -p_2^T - \\ -p_3^T - \end{pmatrix} \begin{pmatrix} | \\ X \\ | \end{pmatrix} \quad (14)$$

et

$$x' = \alpha P'X \quad \text{soit} \quad \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \alpha \begin{pmatrix} -p_1'^T - \\ -p_2'^T - \\ -p_3'^T - \end{pmatrix} \begin{pmatrix} | \\ X \\ | \end{pmatrix} \quad (15)$$

x et PX étant deux vecteurs collinéaires, leur produit vectoriel est nul. On peut alors montrer que X vérifie :

$$\begin{pmatrix} yp_3^T - p_2^T \\ p_1^T - xp_3^T \\ y'p_3'^T - p_2'^T \\ p_1'^T - x'p_3'^T \end{pmatrix} X = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (16)$$

Qui se résous par décomposition en valeur singulière.

3.5.3 Localisation 3D par 2 caméras : hypothèse de caméra calibrée

On fait l'hypothèse dans cette section que les caméras sont calibrées. Une calibration des caméras permet de conserver les propriétés métriques des espaces Euclidiens, fournissant donc de meilleurs résultats de localisation. Le calcul de P' se fait en résolvant l'équation

$$E = K'^T FK \quad (17)$$

où E est appelée matrice essentiel et K est la matrice des paramètres intrinsèques.

On obtiens par une décomposition en valeurs singulières 4 solutions [9]. La sélection de l'unique solution se fait en vérifiant que le point X se trouve devant les deux caméras. Une fois la matrice P' déterminée, on procède à la localisation de manière identique à celle décrite dans la section précédente.

3.5.4 Implémentation de la méthode de localisation.

Les étapes suivis par notre algorithme implémentant la méthode de localisation 3D à partir des flux vidéos de 2 caméras supposées non-calibrées sont les suivantes :

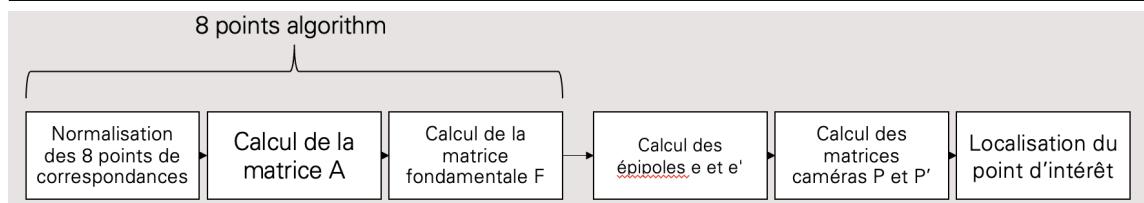


FIGURE 18 – Fonctionnement de notre implémentation de localisation 3D par géométrie épipolaire.

3.6 Calcul du plus court chemin pour diriger l'utilisateur

3.6.1 Introduction

Pour guider l'utilisateur vers un emplacement spécifique à l'intérieur de l'environnement, nous devons calculer le plus court chemin entre la position actuelle de l'utilisateur et l'emplacement cible. Ce chemin peut être calculé à l'aide d'algorithme basés sur des graphes, tels que l'algorithme A*.

3.6.2 Algorithme de recherche A*

La recherche A* est un algorithme de recherche informée qui trouve le plus court chemin entre le noeud de départ et le noeud d'arrivée dans un graphe. Il utilise une fonction heuristique pour estimer le coût du chemin du noeud courant vers le noeud d'arrivée. Cette fonction heuristique permet de guider la recherche vers les chemins les plus prometteurs.

3.6.3 Fonctionnement de base de l'algorithme A* :

1. **Initialisation :** On définit le noeud de départ et le noeud d'arrivée, ainsi qu'une liste ouverte contenant le noeud de départ et une liste fermée initialement vide.
2. **Itération :** Tant que la liste ouverte n'est pas vide :
 - On sélectionne le noeud ayant le coût $f(x) = g(x) + h(x)$, où $g(x)$ représente le coût réel du chemin parcouru jusqu'au noeud courant et $h(x)$ représente l'estimation du coût restant pour atteindre le noeud d'arrivée).
 - On déplace le noeud sélectionné de la liste ouverte vers la liste fermée.
 - On explore les voisins du noeud courant :
 - Pour chaque voisin, on calcule son coût $g(x)$ en additionnant le coût $g(x)$ du noeud courant et le coût de l'arête reliant les deux noeuds.
 - On estime son coût $f(x)$ en calculant $g(x) + h(x)$.
 - Si le voisin n'est pas dans la liste fermée et qu'il est absent de la liste ouverte ou que le nouveau coût $g(x)$ calculé est inférieur à l'ancien coût $g(x)$ associé au voisin dans la liste ouverte, on met à jour les informations du voisin ($g(x)$ coût, parent) et on l'ajoute à la liste ouverte.
3. **Arrêt :** Si le noeud d'arrivée est trouvé dans la liste fermée, on a alors identifié le plus court chemin en retracant le chemin à partir du noeud d'arrivée en suivant les parents jusqu'au noeud de départ. Si la liste ouverte est vide et que le noeud d'arrivée n'est pas trouvé, aucun chemin n'existe entre le départ et l'arrivée.

3.6.4 Implémentation de l'algorithme du plus court chemin

Nous avons implémenté l'algorithme de recherche A* en Python pour calculer le chemin optimal entre le point d'entrée de l'utilisateur et l'emplacement de la personne détectée. L'algorithme prend en entrée un graphe, les coordonnées de l'entrée du bâtiment et une personne détectée. Il renvoie une liste de nœuds représentant le plus court chemin permettant à l'utilisateur d'atteindre l'emplacement cible.

3.6.5 Test de l'algorithme du plus court chemin

Avec l'autorisation de l'administration du bâtiment, nous avons pu accéder aux fichiers AutoCAD du "Grand Hall". Cela nous a permis d'exploiter la structure détaillée de la salle à des fins de test. À l'aide d'OpenCV, nous avons converti cette structure en une représentation matricielle adaptée à l'algorithme A*. Nous avons ensuite appliqué l'algorithme A* à cette matrice et vérifié qu'il calcule correctement le plus court chemin entre l'entrée et divers points du Grand Hall, simulant ainsi des destinations potentielles de l'utilisateur. Nous avons également confirmé que l'algorithme fonctionne efficacement même pour des environnements comportant un grand nombre de nœuds (représentant différents emplacements à l'intérieur du bâtiment).

3.6.6 Conclusion

L'implémentation de l'algorithme A* permet de calculer efficacement le plus court chemin à l'intérieur d'un environnement représenté sous forme de graphe. En intégrant cette fonctionnalité à notre système, nous pouvons guider l'utilisateur vers la personne détectée en lui indiquant le chemin optimal à suivre. Les tests réalisés sur le Grand Hall ont démontré la validité de notre approche et sa capacité à s'adapter à des environnements complexes.

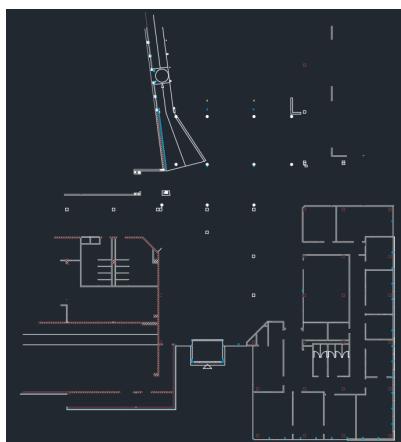


FIGURE 19 – Autocad du bâtiment.

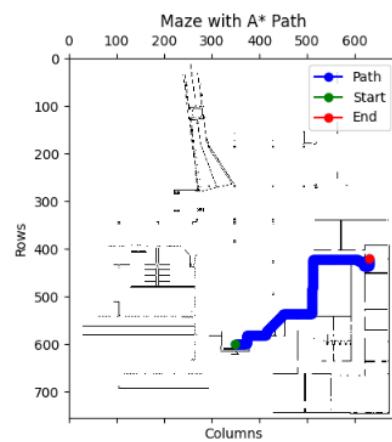


FIGURE 20 – Application de A* sur le bâtiment.

3.7 Application Android

Pour réaliser notre application Android, nous avons d'abord réfléchi à quel environnement de développement nous allions utiliser. Après avoir comparé les principaux (Android Studio, Eclipse IDE, IntelliJ IDEA), nous avons choisi Android Studio car il est gratuit et qu'il existe de nombreuses ressources disponibles sur internet. De plus, l'un des membres de notre groupe l'avait déjà utilisé.

3.7.1 L'IDE Android Studio

Android Studio est un environnement de développement intégré (IDE) [3], créé par Google. L'un de ses plus grands atouts est la possibilité de simuler le résultat du projet sur un grand nombre d'appareils et la capacité de modifier la version d'Android utilisée par ces appareils. De plus, il offre également la possibilité de développer en langage Java ou Kotlin. Lorsque l'on crée un projet avec Android Studio, il est séparé en trois dossiers (manifest, java, res) et un fichier Gradle. Nous avons cherché à comprendre comment fonctionnent et interagissent ces différents fichiers entre eux afin de pouvoir développer de manière efficace.

Le fichier Gradle est un système de construction utilisé dans Android Studio pour automatiser et rationaliser le processus de construction des projets Android.

Le dossier manifest contient le fichier AndroidManifest.xml qui permet de gérer les autorisations que l'application doit avoir pour fonctionner correctement. Dans notre cas, nous devons lui donner accès à Internet car nous souhaitons recevoir les informations provenant des caméras par Wi-Fi. Il sert également de gestionnaire lors de l'exécution de l'application pour savoir dans quel ordre les activités doivent être lancées.

Le dossier res est un dossier de ressources qui contient des sous-dossiers contenant des fichiers .xml qui gèrent la partie graphique du projet. Le sous-dossier layout contient les mises en page des activités et l'ensemble des éléments interactifs. Le sous-dossier values permet de stocker des chaînes de caractères et des couleurs qui peuvent être utilisées dans le code. Le sous-dossier mipmap permet de définir l'icône du lanceur de l'application. Le sous-dossier drawable permet de stocker des images ou des fonds que l'on veut utiliser dans le code.

Le dossier java contient les fichiers qui rendent l'interface graphique générée par les fichiers .xml dynamique. Les fichiers représentent la partie logique des activités, ils se connectent aux éléments interactifs tels que les boutons et agissent sur le code. Ces fichiers peuvent être rédigés en langage java ou kotlin, ils sont en java dans notre cas.

3.7.2 Architecture espérée de l'application

Afin de développer le plus efficacement possible, il est nécessaire de réaliser une maquette de l'application qui une fois validée par les pompiers nous permet de nous concentrer uniquement sur le développement et la résolution des problèmes techniques.

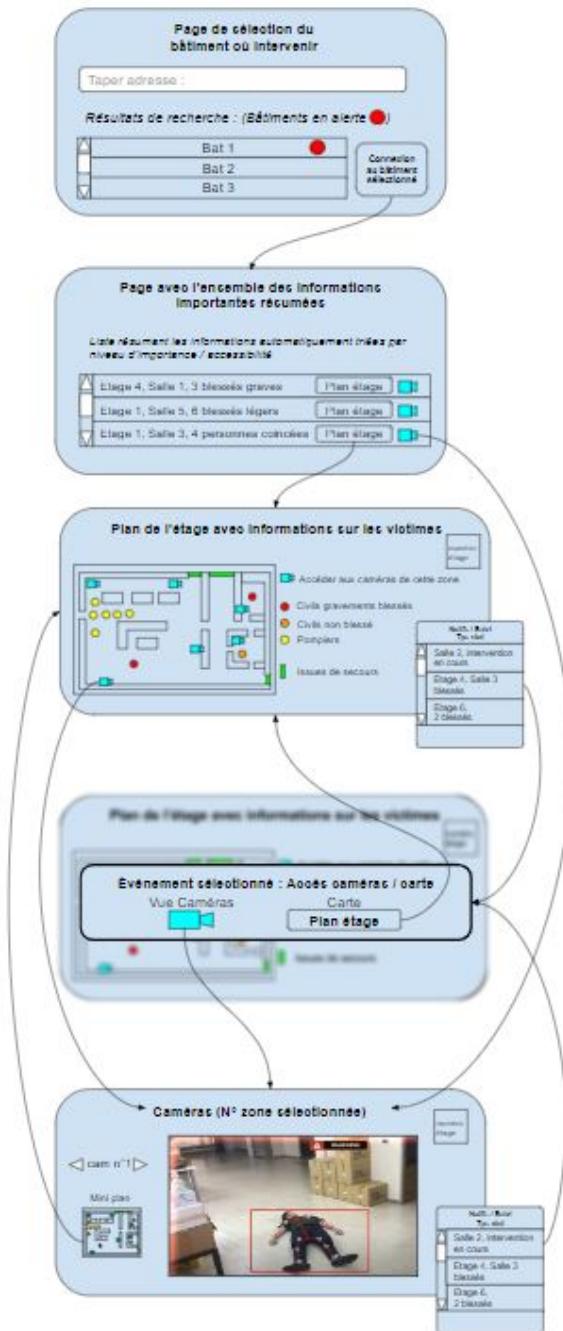


FIGURE 21 – Maquette de l'application Android

Comme illustré sur la figure 21, nous souhaitons que l’application dispose de trois activités différentes. Une première pour choisir le bâtiment sur lequel nous voulions nous connecter, une seconde pour choisir la caméra du bâtiment qui nous intéresse, et enfin une dernière activité qui afficherait la vue des caméras.

3.7.3 Difficultés rencontrées

Nous avons rencontré un problème majeur lors du développement de l’application dans l’activité 2. Il était impossible de cliquer sur un élément de cette activité, ce qui a grandement freiné le développement du projet. Nous n’avons pas trouvé la cause de cette erreur car l’objet *OnClickListener* était bien instancié lors de l’analyse du code au débogueur. Nous avons été contraints de renoncer à produire un code SOLID et avons rendu l’application finale moins générale. Cependant, nous avons conservé les lignes directrices que nous nous étions fixées :

- une interface "user-friendly"
- une application utilisable rapidement
- une possibilité de changer de vue de caméra facilement

3.7.4 Acquisition des flux vidéos

Nous avons implémenté dans notre application l’ensemble des méthodes permettant de récupérer et afficher un flux vidéo transmis par une caméra ESP32 Cam. Pour cela, nous avons utilisé des *Handler* pour générer des requêtes asynchrones et actualiser l’image affichée dans un conteneur *ImageView* 60 fois par seconde. Lorsque l’utilisateur arrive dans l’activité, les données relatives à la caméra sélectionnée sont passées à la méthode *Update*, qui utilise le *Handler* pour afficher l’image ainsi que les aides à la décision.

3.7.5 Traitement des informations d’aide à la décision

Afin de rendre l’utilisation de l’application la plus intuitive possible, l’application utilise des marqueurs pour guider les pompiers vers la caméra qui présente un danger. Dans la première activité, un système de couleur indique si le bâtiment contient des victimes ou non, grâce à une pastille de couleur verte si tout va bien et rouge s’il y a un danger.

3.8 Refonte de l’architecture logicielle de l’application PC

Afin de rendre le logiciel PC plus robuste face à des modifications et ajouts des fonctionnalités nécessaires à la réalisation de notre projet, nous avons réécrit le code fait au fait au semestre 3 quasiment entièrement et l’avons découpés en multiples classes en appliquant les principes SOLID du développement logiciel. Ces classes représentent les éléments de la chaîne de traitement de l’information depuis la caméra jusqu’à l’envoi des données d’aides à la décision.

On peut répertorier les principales :

- Classe Camera : chaque instance de Camera possède son propre Thread d’acquisition d’image. Elle est généralisée et peut donc permettre de recueillir soit les images de la webcam du PC (pour effectuer des tests), soit les images d’une caméra distante sur réseau (ex : ESP32Cam). Elle peut également être une caméra virtuelle permettant de lire des vidéos enregistrées (très utile pour les tests).

- Classe PoseEstimator : La responsabilité de détection de posture à partir des frames capturées par l'objet caméra est déléguée à toute instance de cette classe.
- Classe HumanExtractor : La classe ayant la responsabilité de fournir la liste d'humains à partir de toutes les poses détectées dans une scène filmée par une caméra.
- Classe Human : Cette classe représente un humain détecté. Elle contient essentiellement sa dernière posture détectée (ainsi que les n anciennes postures, n étant la taille du buffer de l'historique de pose). Chaque instance d'humain calcule ses attributs cinématiques (positions, vitesses, accélérations des articulations et de la bounding box) en fonction de l'historique des postures dans son buffer au travers de la méthode updateHuman().
Ainsi par la suite, cette même méthode estime l'état d'activité de l'humain à partir de ces observations cinématiques.
- Classe BiCam : Est la classe qui représente un groupe de caméra et effectue les calculs de localisation 3D.
- Classe Map : Est la classe qui représente la carte qui contient les informations de positions des individus dans le bâtiment, déterminés à l'aide de l'algorithme de localisation 3D par caméra et les classes précédentes.
- Classe FrameStream : Est la classe qui permet d'envoyer les flux vidéos vers l'appareil Android du pompier.
- Classe InfosStream : Est la classe qui permet d'envoyer les flux d'informations d'aide à la décision vers l'appareil Android du pompier.
- Classe CoreApp : Est la classe qui régit du fonctionnement global du logiciel. Il fait en quelque sorte l'assemblage de toutes les instances de classes citées plus haut pour construire la chaîne d'acquisition d'image vers l'obtention des données d'aide à la décision.
- Classe LaunchCmd : Elle fait l'interface entre l'utilisateur et la classe CoreApp pour le lancement du logiciel à l'aide de commandes bash (console) en fournissant diverses options notamment celles relatives aux caméras auxquelles se connecter.
- Classe LaunchGui : Elle fait l'interface entre l'utilisateur et la classe CoreApp pour le lancement du logiciel à l'aide d'une IHM (basée sur PyQt5). Elles permettent de visualiser avec plus d'aisance de multiples vidéos provenant de plusieurs caméras et donne un aperçu du logiciel pour le client installant notre dispositif dans son bâtiment.

4 Dialogue avec les pompiers

Afin de développer une application qui soit la plus utile possible pour les pompiers, nous nous sommes confrontés à leurs exigences. Nous les avons rencontrés une première fois afin de recueillir leurs avis concernant notre projet. Lors de cette première entrevue, nous avons voulu savoir s'ils utilisaient déjà des dispositifs

technologiques similaires lors de certaines de leurs interventions. Ensuite, nous leur avons demandé quelles informations ils souhaiteraient avoir sur le bâtiment et les différentes salles dans notre application. Cette première rencontre a été un succès, car ces questions visaient à nous aider à élaborer un questionnaire plus formel, que nous distribuerons à l'ensemble des pompiers, afin que notre échantillon de réponses soit le plus proche possible de la réalité. Nous y avons inclus un ensemble de questions nécessaires pour que nous puissions développer une application Android répondant au mieux à leurs besoins.

5 Tests unitaires

5.1 Test du code de captures video par l'ESP32 Cam

Dans un premier temps nous avons validé la connexion des ESP32 Cams à un réseau local et la diffusion des flux vidéos au travers de ce dernier. En effet nous pouvons voir par l'intermédiaire d'un navigateur Web sur un PC connecté à ce même réseau que le flux vidéo est bien disponible et retransmis. Cette démarche est régulièrement répétée lors du développement de notre dispositif pour vérifier rapidement la disponibilité d'un flux vidéo en particulier.

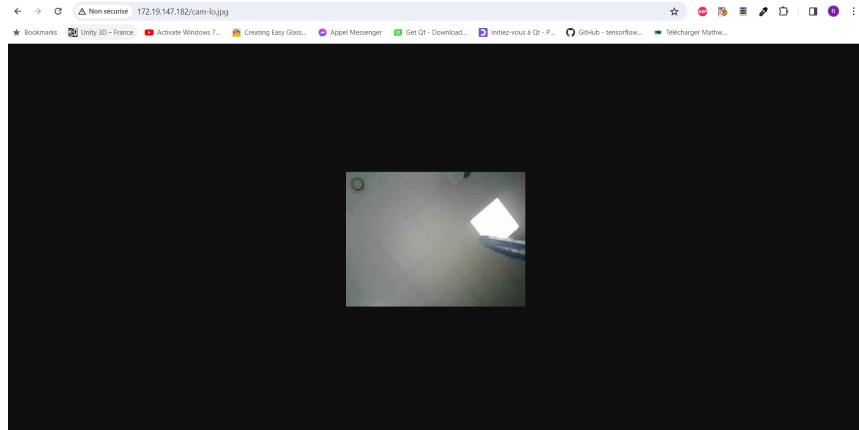


FIGURE 22 – Affichage d'une image capturée par une ESP32-CAM sur navigateur Web

5.2 Test de notre circuit de développement pour l'ESP32 Cam

Entre autres, le circuit de développement nous permet désormais de manipuler plus efficacement et plus rapidement les ESP32 Cams lors des manipulations de téléchargement et changement de réseau Wi-Fi.

5.3 Test de la méthode de multi-detection de postures avec Yolov7 et OpenPose

Pour les tests nous utilisons un PC Windows avec CPU Intel Core i5 10300H, 2.5 GHz et GPU Nvidia GeForce RTX 3060 Mobile 4Gb.

Nous avons validé la possibilité de faire de la détection de personnes (multipose estimation et yolov7) et ce sur plusieurs flux vidéos en simultané. Le taux de rafraîchissement est cependant bas (en dessous de

5 FPS lorsqu'une personne est détectée en raison de l'estimation sur un autre réseau de neurone (basée sur OpenPose) qui lui ne bénéficie pas de l'accélération GPU). Aussi cette méthode est peu robuste face aux situations où deux personnes se recouvrent partiellement.

5.4 Test de la réception du flux vidéo sur appareil Android

Enfin nous avons validé la réception vidéo, d'une caméra à la fois, sur un dispositif Android.

A présent, les tests suivants présentent les avancées faites au cours de la deuxième partie du projet (Semestre 4)

5.5 Amélioration du processus de sélection du réseau de l'ESP32-Cam

Pour faciliter le processus de développement et de démonstration nous avons fait en sorte de pouvoir changer de réseau non plus uniquement au démarrage de la carte mais également lorsque la carte est en runtime (capture vidéo et envoie sur le réseau).

Ce nouveau processus fonctionne et nous nous sommes fournis des gains de temps pour effectuer les autres tests mettant en œuvre l'ESP32-Cam

5.6 Test de la réception de plusieurs flux vidéos provenant de plusieurs ESP32 Cams

Après les diverses mises à jour architecturales du programme tournant sur l'ordinateur central, la réception de flux est encore plus robuste face aux situations de déconnexion inopinée d'une ou plusieurs caméras.

On simule une interruption brusque de connexion (qui en situation réelle serait due à une perte de signal ou un déchargement de batterie) en débranchant une caméra. On vérifie bien que le programme PC réagit bien en continuant de fonctionner et en réactualisant l'image lorsque la caméra se reconnecte.

5.7 Test de la nouvelle méthode de multi-détection de postures avec Yolov7

Ce test unitaire présente les avancées qui ont été faites pour l'amélioration de la détection de posture. Ici on rappelle que la détection de posture se fait désormais intégralement au travers de YoloV7. Les performances ont drastiquement augmentées car nous pouvons effectivement bénéficier de l'accélération GPU sur l'estimation de pose.

Pour les tests nous utilisons un PC Windows avec CPU Intel Core i5 10300H, 2.5 GHz et GPU Nvidia GeForce RTX 3060 Mobile 4Gb.

Avec l'ancienne méthode de multi-détection de posture (basé sur un découpage de l'image sur les personnes détectées puis estimation sur la zone détournée par OpenPose d'OpenCV) des performances autour de 5 images par secondes.

Avec la nouvelle méthode nous obtenons non seulement de meilleures en termes de rapidité mais aussi de meilleures performances de détection car désormais deux personnes proches ou partiellement cachées l'une par l'autre peuvent être détectées. Nous obtenons des performances autour de 30 FPS, pour une caméra (un flux vidéo)

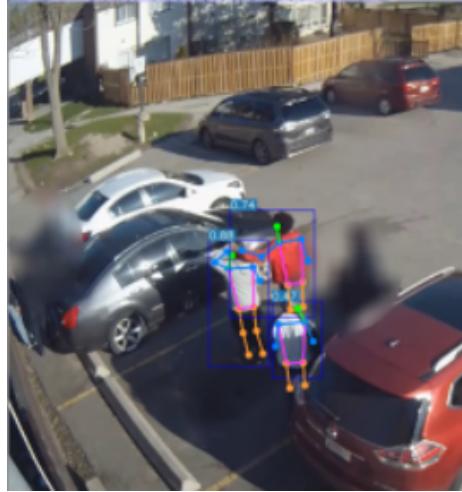


FIGURE 23 – Test de la multi-pose estimation avec la nouvelle méthode basée entièrement sur YoloV7

5.8 Test de l'algorithme d'extraction des humains à partir des poses détectés par YoloV7

Comme on le voit sur ces images notre algorithme de distinction des humains est plutot prometteur et est relativement robuste face aux croisements d'humains. Sur les vidéos de football l'algorithme parvient à ne pas confondre humain 3 et humain 2 au cours du croisement et dans la vidéo de la ville il parvient à ne pas confondre humain 1, humain 2 et humain 5 au sein du croisement. Cela permet de valider cet algorithme pour le calcul des propriétés cinématiques et des états propre a chaque individu sans confusion.

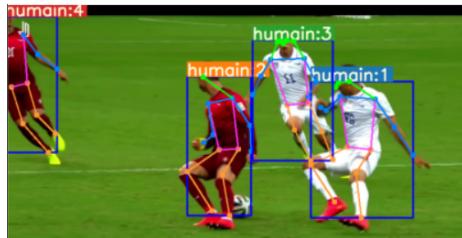


FIGURE 24 – Extraction d'humains (1)

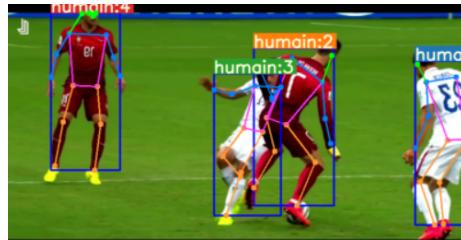


FIGURE 25 – Extraction d’humains (2)

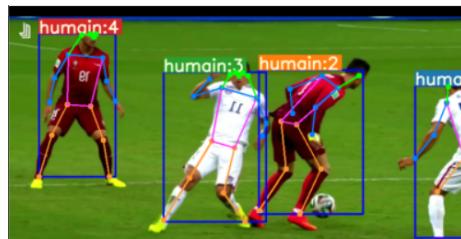


FIGURE 26 – Extraction d’humains (3)

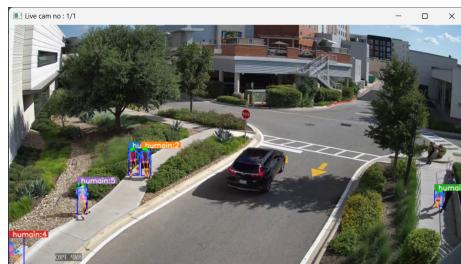


FIGURE 27 – Extraction d’humains (4)

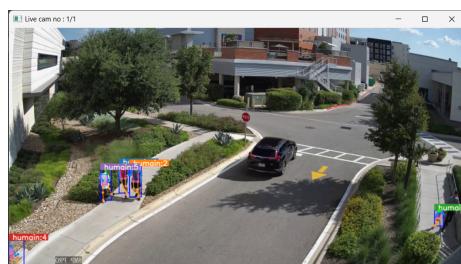


FIGURE 28 – Extraction d’humains (5)



FIGURE 29 – Extraction d’humains (6)

5.9 Test de l’obtention de mesures cinématiques sur la posture détectée

Nous mesurons bien des valeurs images de la vitesse, accélération moyenne de la posture du sujet et la vitesse moyenne globale des articulations du sujet.



FIGURE 30 – Observations cinématiques

5.9.1 Test de l’estimation d’états le plus probable

Nous avons implémentés avec succès notre méthode permettant à partir des mesures précédentes d'estimer l'état de posture d'un sujet parmi celles présentées.

```
self.labels = ["non-déterminé", "dynamique", "immobile", "choc"]
```

FIGURE 31 – Etats à estimer

5.10 Tests de fonctionnement de localisation 3D par 2 caméras.

Afin d’évaluer les performances de notre implémentation de la méthode de localisation 3D à partir du flux vidéos de 2 caméras supposées non calibrées, nous nous sommes basés sur le dataset de test Merton 1 proposé par l’université d’Oxford [17]. Ce dataset présente sous la forme d’un fichier Matlab les correspondances entre une dizaines de points en général, pour des paires d’images. Ces correspondances nous permettent d’alimenter l’algorithme des 8-points et d’effectuer des vérifications de triangulation sur les points restants.

Nous avons de même réalisé des tests sur des photographies prises par nos soins. Afin d'estimer la précision de notre algorithme, nous avons chercher à évaluer la norme d'un vecteur entre 2 points que nous localisons,

distance que nous connaissons avec une bonne précision. Cela nous permet de affranchir des erreurs de mesure de position dans le monde réelle. Le protocole suivit est le suivant :

- Prendre une photographie sous deux angles de vue.
- Utiliser un outil numérique pour récupérer les coordonnées pixels de nos 8 points d'intérêts servant à l'algorithme des 8-points.
- Les placer dans une liste Python selon le format attendu par notre programme.
- Trianguler 2 points d'intérêt situés à une distance connue l'un de l'autre.
- Calculer la distance les séparants.

Voici les 2 images sur lesquelles nous avons réaliser une mesure de distance :



FIGURE 32 – Point vue gauche.



FIGURE 33 – Point de vue droit.

Nous avons chercher à localiser les points représentant les graduations de 2cm et 12cm sur les règles, en s'attendant à obtenir une distance calculée, poche de 10cm.

Les points utilisés pour l'algorithme des 8 points sont les graduations de 0 cm, 4cm, 8cm, 25cm sur une des règles et 0cm, 8cm, 15 cm et 17cm sur l'autre règle. Après localisation des 2 points par notre algorithme Python, nous obtenons une distance de 8,8cm entre les 2 points. L'erreur est donc proche du centimètre ce qui est acceptable pour notre système, vis-à-vis des exigences listées en Figure-2.

Nous avons donc voulu réaliser une localisation à la suite de ce premier test. Nous avons procéder au protocole précédent en photographiant un objet dont la position est connu par rapport à des repères :

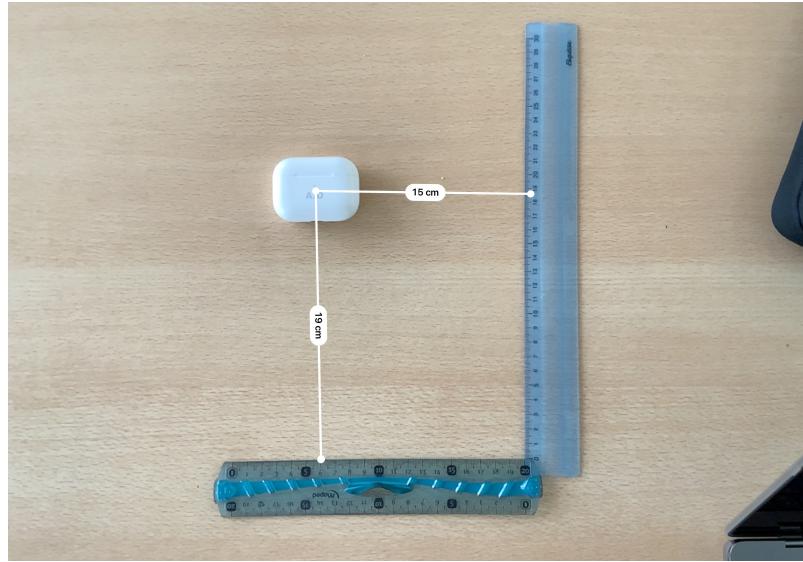


FIGURE 34 – Localisation du boîtier par rapport à des repères

photographié sous les 2 points de vues :



FIGURE 35 – Point vue gauche.



FIGURE 36 – Point de vue droit.

En cherchant à localiser le boîtier, nous obtenons des coordonnées négatives, le boîtier est donc localisé derrière la caméra de gauche (qui est utilisé comme origine du repère). Nous en avons conclut que notre algorithme commet une erreur régulière (offset) sur la localisation de sujet, ce qui explique une localisation erronée mais un calcul de distance efficace.

Afin d'améliorer notre algorithme nous avons comparer les résultats des calculs intermédiaires tels que la matrice fondamentale F et les matrices caméras P et P' avec les résultats des fonctions implantés dans OpenCV. Nous avons notamment constaté de fortes différences pour la matrice F . Nous avons donc fait le choix de recourir à OpenCV pour le calcul de F afin de rendre fonctionnel notre code.

6 Impact environnemental

6.1 Analyse de l'empreinte carbone du projet

La présente section se focalise sur une analyse cruciale dans le cadre de notre projet : l'empreinte carbone. Conscients des implications environnementales inhérentes à notre système reposant sur un réseau de caméras et un ordinateur fonctionnant en continu, nous nous penchons sur les divers composants et leur impact écologique. Ainsi, nous analyserons d'abord le cas des caméras ESP32 au travers de leur composition. Ensuite, nous étudierons l'émission de CO_2 lorsque notre système est en fonctionnement. Pour cela, nous avons considéré deux scénarios probables d'utilisation que nous expliciterons par la suite.

6.2 Analyse des composants de la caméra

Afin d'évaluer avec précision l'empreinte carbone de notre projet, il est essentiel de connaître l'ensemble des composants des systèmes que nous utilisons, notamment les caméras ESP32. Nous sommes conscients que, bien que les caméras ESP32 consomment peu d'énergie, elles nécessitent des batteries pour assurer leur autonomie. Nous reconnaissions que ces batteries ont un fort impact écologique et qu'il est impératif de les recycler en fin de vie. Cependant, nous avons décidé de ne pas traiter cette problématique, car elle peut être contournée en reliant les caméras au réseau électrique. Selon la documentation technique [5], ces caméras sont composées de nombreux éléments. Dans cette section, nous nous concentrerons uniquement sur ceux ayant le plus grand impact environnemental :

1. Microcontrôleur :

- Modèle : ESP32-D0WDQ6 (QFN48)
- Coeurs : 2 coeurs Xtensa® LX106 32 bits
- Fréquence d'horloge : Jusqu'à 240 MHz
- Mémoire RAM : 520 Ko
- Stockage flash : 4 Mo
- Connectivité : Wi-Fi 802.11 b/g/n, Bluetooth 4.2 BR/EDR et BLE

2. Capteur d'image :

- Modèle : OV2640 ou OV7670 (selon la variante de la caméra)
- Résolution : 2 mégapixels (1600 x 1200 pixels)
- Taille du capteur : 1/4 pouce
- Interface : SCCB

5. Antennes :

- Wi-Fi : Antenne PCB intégrée
- Bluetooth : Antenne PCB intégrée

6.3 Co₂ émis lors de l'utilisation du dispositif

Maintenant que nous avons évalué l'émission de CO_2 du dispositif lors de sa création, nous allons étudier son émission de CO_2 en fonctionnement. Pour cela, nous examinerons sa consommation d'énergie électrique moyenne, puis convertirons cette énergie en équivalent carbone. Pour ces calculs, nous ferons certaines hypothèses, notamment celle d'être en France et donc d'utiliser le mix énergétique français pour les calculs.

Ainsi, en France en 2023, l'émission moyenne de 54 g de CO_2 par kWh selon les données fournies par RTE France. D'après [5], la tension d'alimentation en fonctionnement normal est de 3,3 V et le courant d'alimentation normal est de 200 mA. Ainsi, la puissance consommée par une caméra est de 0,726 W. Nous

allons maintenant étudier deux scénarios : l'un où le dispositif est actif en permanence et l'autre où il ne l'est que lorsque des salariés sont présents, soit 35 heures par semaine.

Scénario 1 :

$0.726 \times 52 \times 35 \times 60 \times 60 = 4,76 \cdot 10^6 J = 1,15 kWh$ l'émission de CO_2 associé est donc de 62 g par ans par caméra pour une durée de fonctionnement de 35 heures par semaines.

Scénario 2 :

$$0.726 \times 365 \times 24 \times 60 \times 60 = 2,29 \cdot 10^7 J = 5,55 kWh$$

L'émission de CO_2 associée est donc de 300 g par an par caméra pour un fonctionnement en continu. Ce chiffre doit être comparé à l'émission sur une année d'une caméra de surveillance classique, qui est de 78,8 kg de CO_2 . Cependant, il faut aussi garder en mémoire que dans ce chiffre, on ne prend pas en considération la consommation d'énergie de l'ordinateur associé à notre dispositif.

Nous allons donc estimer la consommation électrique de l'ordinateur associé. Pour cela, nous allons supposer que l'ordinateur aura un processeur Intel Core i5-11400 et une carte graphique Nvidia GeForce GTX 1650. On considérera pour les calculs que les autres composants dissipent 20 W. Ainsi, la consommation électrique associée pour cet ordinateur, selon [11] et [16], est de $75 + 50 + 20 = 145W$. Ce qui revient, dans le cas du scénario 2, à une émission de 60 kg de CO_2 par an. On pourrait considérer que l'émission de l'ordinateur rend notre système moins avantageux vis-à-vis d'un système de surveillance traditionnel. Cependant, il faut se rappeler que nous faisons ici des calculs sur une seule caméra. Or, un ordinateur est capable de traiter les données de plusieurs caméras ESP32. Notre solution se démarque alors d'autant plus que le nombre de caméras utilisées dans le bâtiment est important. Afin de diminuer encore plus l'empreinte carbone du projet, il pourrait être intéressant d'implémenter un mode de fonctionnement en veille du dispositif lorsque cela n'est pas nécessaire.

7 Conclusion

Réaliser un dispositif multi-caméra d'aide à la décision pour faciliter l'intervention de pompier fait appels à de multiples domaines et axes de travail. Nous avons, au cours de ce semestre réussi à identifier ces axes et répartir efficacement nos tâches. La première correspond à l'ingénierie système où nous avons réussi à cadrer notre étude en délimitant le projet et en déterminant les besoins de nos clients, notamment à l'aide de sondages et réunions auprès de pompiers et de nos professeurs encadrant qui nous ont par ailleurs, confirmés l'intérêt de notre projet. Nous prévoyons de poursuivre nos sondages auprès des pompiers afin d'étoffer nos idées et avoir un maximum d'avis de la part des parties prenantes.

Les différents diagrammes d'architectures nous ont permis de délimiter les sous-systèmes tout en nous fournissant l'opportunité de nous répartir le travail. Le travail d'ingénieur en lui-même a été focalisé sur les exigences principales avant de se tourner vers les exigences secondaires au semestre 4.

Le développement embarqué des cartes ESP32-Cams permettant leur mise en réseau et la diffusion des flux vidéos est dans un stade assez avancé. Il reste à poursuivre en priorité le développement de l'application PC. Celle-ci peut désormais travailler avec différents flux vidéos en simultané et détecter des personnes et leurs postures mais ne transmet pas encore d'indications utiles sur l'application Android utilisée par le chef de bataillon. Cette application doit d'ailleurs être travaillée en priorité afin de récupérer l'ensemble des informations à communiquer et proposer une interface ergonomique et adaptée pour les interventions de pompier.

Ainsi, globalement, toutes les fondations principales de ce projet, dont la pertinence est reconnue par nos parties prenantes, ont été solidement déployées (tant du côté des outils de gestion de projet et de développement que du côté des fonctionnalités techniques du dispositif). Nous progressons donc sereinement dans le semestre 4, confiant en la réussite de notre projet.

Références

- [1] *Accelerated PyTorch training on Mac.* URL : <https://developer.apple.com/metal/pytorch/>.
- [2] *Basics of Brute Force Matching.* URL : https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html (visité le 14/04/2024).
- [3] *Découvrir Android Studio | Android Studio.* fr. URL : <https://developer.android.com/studio/intro?hl=fr> (visité le 19/12/2023).
- [4] *Départ de feu et incendie domestique,* <https://www.republicain-lorrain.fr/faits-divers-justice/2023/12/24/depart-de-feu-et-incendie-domestique-comment-prevenir-les-risques-les-bons-gestes-a-avoir>. fr. 2023. URL : <https://www.republicain-lorrain.fr/faits-divers-justice/2023/12/24/depart-de-feu-et-incendie-domestique-comment-prevenir-les-risques-les-bons-gestes-a-avoir>.
- [5] *ESPRESSIF SYSTEMS. ESP32 Series Datasheet v4.5.* Version 4.5. Espressif Systems. 2024. URL : www.espressif.com.
- [6] *Get started with PyTorch for Apple Silicon Mac.* URL : <https://pytorch.org/get-started/locally/>.

-
- [7] *Haar Cascade*, <https://machinelearningmastery.com/using-haar-cascade-for-object-detection/>. URL : <https://machinelearningmastery.com/using-haar-cascade-for-object-detection/>.
 - [8] *Haar Cascade*, <https://medium.com/@ishudey11032002/what-is-yolo-algorithm-ef5a3326510b>. URL : <https://medium.com/@ishudey11032002/what-is-yolo-algorithm-ef5a3326510b>.
 - [9] Richard HARTLEY et Andrew ZISSEMAN. *Multiple View Geometry in Computer Vision*. 2004.
 - [10] Ho-Won,L. ;Kyong-Oh,L. ;Ji-Hye,B. ;Se-Yeob,K. ;Yoon-Young,P. *Using Hybrid Algorithms of Human Detection Technique for Detecting Indoor Disaster Victims*, 3 November 2022. doi.org/10.3390/computation10110197. fr. URL : <https://www.mdpi.com/2079-3197/10/11/197>.
 - [11] *Intel® Core™ i5-11400 Processor (12M Cache, up to 4.40 GHz) - Product Specifications*. en. URL : <https://www.intel.com/content/www/us/en/products/sku/212270/intel-core-i511400-processor-12m-cache-up-to-4-40-ghz/specifications.html> (visité le 09/04/2024).
 - [12] Jaradat,F. ;Valles,D. *A Victims Detection Approach for Burning Building Sites Using Convolutional Neural Networks* January 2020. DOI : 10.1109/CCWC47524.2020.9031275. URL : https://www.researchgate.net/publication/338536071_A_Victims_Detection_Approach_for_Burning_Building_Sites_Using_Convolutional_Neural_Networks.
 - [13] *Open Pose*, <https://medium.com/@rotemsha/how-does-openpose-actually-work-11a29872f30e>. URL : <https://medium.com/@rotemsha/how-does-openpose-actually-work-11a29872f30e>.
 - [14] Pei-Fen,T. ;Chia-Hung,L. ;Shyan-Ming,Y. *Using Deep Learning with Thermal Imaging for Human Detection in Heavy Smoke Scenarios* 18 July 2022. doi.org/10.3390/s22145351. URL : <https://www.mdpi.com/1424-8220/22/14/5351>.
 - [15] *Scale Invariant Feature Transform*. URL : https://fr.wikipedia.org/wiki/Scale-invariant_feature_transform (visité le 14/04/2024).
 - [16] *Test Nvidia GeForce GTX 1650, appétit de moineau pour carte graphique à petit prix*. fr. Mai 2019. URL : <https://www.lesnumeriques.com/carte-graphique/nvidia-geforce-gtx-1650-p50681/test.html> (visité le 09/04/2024).
 - [17] Oxford UNIVERSITY. *DATA : Two-view Geometry*. URL : <http://cmp.felk.cvut.cz/data/geometry2view/index.xhtml>.
 - [18] Zubairi, J.A., Idwan, S. *Localization and Rescue Planning of Indoor Victims in a Disaster*. *Wireless Pers Commun* 126, 3419–3433 (2022). doi.org/10.1007/s11277-022-09871-z. URL : <https://www.mdpi.com/1424-8220/22/14/5351>.

Références

- [1] *Accelerated PyTorch training on Mac.* URL : <https://developer.apple.com/metal/pytorch/>.
- [2] *Basics of Brute Force Matching.* URL : https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html (visité le 14/04/2024).
- [3] *Découvrir Android Studio | Android Studio. fr.* URL : <https://developer.android.com/studio/intro?hl=fr> (visité le 19/12/2023).
- [4] *Départ de feu et incendie domestique,* <https://www.republicain-lorrain.fr/faits-divers-justice/2023/12/24/depart-de-feu-et-incendie-domestique-comment-prevenir-les-risques-les-bons-gestes-a-avoir.fr>. 2023. URL : <https://www.republicain-lorrain.fr/faits-divers-justice/2023/12/24/depart-de-feu-et-incendie-domestique-comment-prevenir-les-risques-les-bons-gestes-a-avoir>.
- [5] ESPRESSIF SYSTEMS. *ESP32 Series Datasheet v4.5.* Version 4.5. Espressif Systems. 2024. URL : www.espressif.com.
- [6] *Get started with PyTorch for Apple Silicon Mac.* URL : <https://pytorch.org/get-started/locally/>.
- [7] *Haar Cascade,* <https://machinelearningmastery.com/using-haar-cascade-for-object-detection/>. URL : <https://machinelearningmastery.com/using-haar-cascade-for-object-detection/>.
- [8] *Haar Cascade,* <https://medium.com/@ishudey11032002/what-is-yolo-algorithm-ef5a3326510b>. URL : <https://medium.com/@ishudey11032002/what-is-yolo-algorithm-ef5a3326510b>.
- [9] Richard HARTLEY et Andrew ZISSERMAN. *Multiple View Geometry in Computer Vision.* 2004.
- [10] Ho-Won,L. ;Kyong-Oh,L. ;Ji-Hye,B. ;Se-Yeob,K. ;Yoon-Young,P. *Using Hybrid Algorithms of Human Detection Technique for Detecting Indoor Disaster Victims,* 3 November 2022. doi.org/10.3390/computation10110197. fr. URL : <https://www.mdpi.com/2079-3197/10/11/197>.
- [11] Intel® Core™ i5-11400 Processor (12M Cache, up to 4.40 GHz) - Product Specifications. en. URL : <https://www.intel.com/content/www/us/en/products/sku/212270/intel-core-i511400-processor-12m-cache-up-to-4-40-ghz/specifications.html> (visité le 09/04/2024).
- [12] Jaradat,F. ;Valles,D. *A Victims Detection Approach for Burning Building Sites Using Convolutional Neural Networks January 2020.* DOI : 10.1109/CCWC47524.2020.9031275. URL : https://www.researchgate.net/publication/338536071_A_Victims_Detection_Approach_for_Burning_Building_Sites_Using_Convolutional_Neural_Networks.
- [13] *Open Pose,* <https://medium.com/@rotemsha/how-does-openpose-actually-work-11a29872f30e>. URL : <https://medium.com/@rotemsha/how-does-openpose-actually-work-11a29872f30e>.
- [14] Pei-Fen,T. ;Chia-Hung,L. ;Shyan-Ming,Y. *Using Deep Learning with Thermal Imaging for Human Detection in Heavy Smoke Scenarios* 18 July 2022. doi.org/10.3390/s22145351. URL : <https://www.mdpi.com/1424-8220/22/14/5351>.
- [15] *Scale Invariant Feature Transform.* URL : https://fr.wikipedia.org/wiki/Scale-invariant_feature_transform (visité le 14/04/2024).

- [16] *Test Nvidia GeForce GTX 1650, appétit de moineau pour carte graphique à petit prix.* fr. Mai 2019.
URL : <https://www.lesnumeriques.com/carte-graphique/nvidia-geforce-gtx-1650-p50681/test.html> (visité le 09/04/2024).
- [17] Oxford UNIVERSITY. DATA :: *Two-view Geometry*. URL : <http://cmp.felk.cvut.cz/data/geometry2view/index.xhtml>.
- [18] Zubairi, J.A., Idwan, S. *Localization and Rescue Planning of Indoor Victims in a Disaster*. *Wireless Pers Commun* 126, 3419–3433 (2022). doi.org/10.1007/s11277-022-09871-z. URL : <https://www.mdpi.com/1424-8220/22/14/5351>.