

Institut Supérieur de l'Électronique et du Numérique

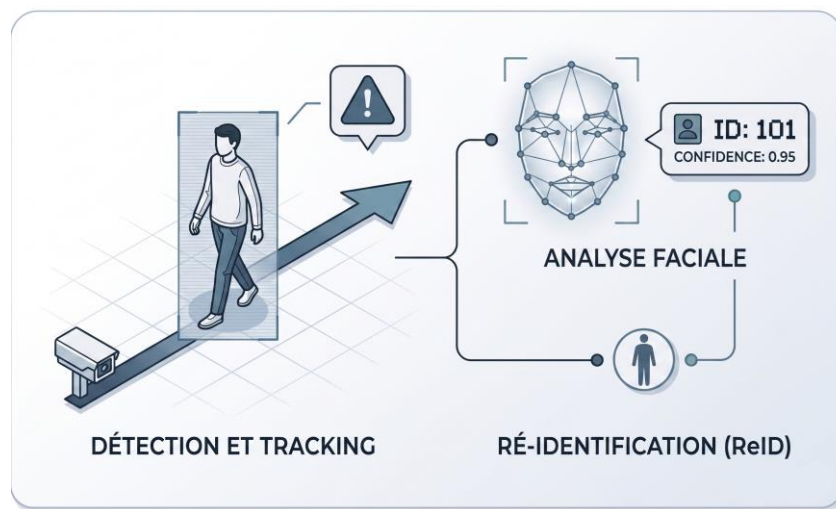
Tél. : +33 (0)2.98.03.84.00

Fax : +33 (0)2.98.03.84.10

20, rue Cuirassé Bretagne

CS 42807 - 29228 BREST Cedex 2 - **FRANCE**

« Système d'évaluation de la coopération pour la
reconnaissance faciale »



Proposé par : « Khadidja OULD AMER »

Thématique : Image

«Ménahèze Ewan »

Domaine professionnel : Intelligence artificielle

Remerciements

Les auteurs souhaitent exprimer leur gratitude envers Mme Khadidja OULD AMER, encadrante de ce projet, pour la qualité de ses orientations techniques, sa disponibilité et ses retours constructifs tout au long de l'année ainsi que Cyril Barrelet.

Des remerciements sont également adressés à l'ensemble du corps enseignant de l'ISEN Yncréa pour la formation dispensée dans les domaines du Deep Learning et de la vision par ordinateur, qui a constitué le socle théorique nécessaire à la réalisation de ce système.

Enfin, les communautés open source des bibliothèques *Ultralytics YOLOv8*, *InsightFace*, *MediaPipe* et *OpenCV* méritent d'être citées pour la mise à disposition d'outils de haute qualité sans lesquels ce projet n'aurait pu aboutir dans les délais impartis.

Résumé

Mots-clés : reconnaissance faciale, coopération comportementale, détection d’occlusion, estimation de pose de tête, analyse de démarche, réidentification de personnes, YOLOv8, InsightFace, MediaPipe, vision temps réel.

Les systèmes de reconnaissance faciale atteignent aujourd’hui des performances élevées en environnement contrôlé ; leur fiabilité se dégrade toutefois significativement face aux variations de pose, aux occlusions partielles, au flou de mouvement ou aux comportements non-coopératifs des individus filmés.

Ce rapport présente le développement d’un système modulaire d’évaluation en temps réel du degré de coopération d’un individu vis-à-vis d’un dispositif de reconnaissance faciale. Le système s’articule autour de quatre blocs fonctionnels : (1) un pipeline de tracking et de réidentification multi-modale combinant YOLOv8m, BoT-SORT et InsightFace ; (2) un module d’analyse de démarche et de posture par estimation de squelette avec YOLOv8-Pose ; (3) un module de détection d’occlusion faciale par zones anatomiques fondé sur MediaPipe FaceMesh et une segmentation colorimétrique Y’CbCr ; (4) un module d’estimation de l’orientation de la tête et du regard combinant résolution PnP et analyse de l’iris.

Les expérimentations menées sur des vidéos personnelles et des datasets publics démontrent la faisabilité d’une évaluation comportementale continue, ouvrant des perspectives d’intégration dans des dispositifs de contrôle d’accès intelligents.

Keywords : face recognition, behavioural cooperation, occlusion detection, head pose estimation, gait analysis, person re-identification, YOLOv8, InsightFace, MediaPipe, real-time vision.

Abstract. Face recognition systems achieve high accuracy in controlled environments, but their reliability degrades in the presence of pose variations, partial occlusions, motion blur, or non-cooperative individual behaviours. This report presents the development of a modular, real-time system for evaluating the cooperation degree of individuals with respect to a facial recognition device. The system combines multi-modal tracking and re-identification (YOLOv8m, BoT-SORT, InsightFace), skeleton-based gait and posture analysis (YOLOv8-Pose), anatomical-zone facial occlusion detection (MediaPipe FaceMesh, Y’CbCr segmentation), and head-orientation/gaze

estimation (PnP, iris tracking). Experiments conducted on personal and public-domain videos confirm the feasibility of continuous behavioural assessment.

Table des matières

Remerciements	i
Résumé	iii
1 Introduction	1
2 Cahier des charges	3
2.1 Contexte et enjeux	3
2.1.1 Domaine et thématique	3
2.1.2 Problématique	3
2.2 Analyse fonctionnelle	4
2.2.1 Diagramme Bête à cornes	4
2.2.2 Diagramme Pieuvre (analyse fonctionnelle externe)	4
2.2.3 Fonctions principales (FP)	5
2.2.4 Fonctions de contrainte (FC)	6
2.3 Spécifications techniques	7
2.3.1 Entrées / Sorties du système	7
2.3.2 Environnement technique cible	7
2.3.3 Contraintes de performance	8
2.4 Analyse des risques	8
2.5 Récapitulatif des livrables	8
3 Gestion de projet	9
3.1 Méthodologie adoptée	9
3.1.1 Approche Agile / Scrum	9
3.1.2 Outils de collaboration	9
3.2 Planification — Diagramme de Gantt	10
3.3 Répartition des tâches	11
3.3.1 Tableau RACI	11
3.3.2 Description des Sprints	11
3.4 Gestion des versions et workflow GitHub	12
3.5 Modifications d’objectifs en cours de projet	12

4	Développement technique	13
4.1	Bloc 1 — Pipeline Tracking et Réidentification multi-modale	13
4.1.1	Objectif et positionnement	13
4.1.2	Détection de personnes — YOLOv8m	13
4.1.3	Tracking — BoT-SORT	14
4.1.4	Extraction et qualité du visage — InsightFace buffalo_l	14
4.1.5	Extraction du corps	15
4.1.6	Galerie et prototypes	15
4.1.7	Machine à états de réidentification	15
4.1.8	Anti-fuites spatiales et fallback temporel	17
4.1.9	Consolidation post-hoc	17
4.1.10	Tests et résultats	17
4.2	Bloc 2 — Analyse de démarche et posture	18
4.2.1	Objectif	18
4.2.2	Estimation du squelette — YOLOv8-Pose	18
4.2.3	Calcul de la vitesse de déplacement	19
4.2.4	Classification de la démarche	19
4.2.5	Schéma bloc — Analyse de démarche	20
4.2.6	Tests sur datasets publics et vidéos personnelles	21
4.3	Bloc 3 — Détection d’occlusion faciale par zones anatomiques	21
4.3.1	Objectif	21
4.3.2	Détection des landmarks — MediaPipe FaceMesh	21
4.3.3	Calibration de la couleur de peau — espace Y’CbCr	22
4.3.4	Score d’occlusion par zone et global	22
4.3.5	Schéma bloc — Détection d’occlusion	23
4.3.6	Tests et conditions expérimentales	23
4.4	Bloc 4 — Orientation de tête et score de coopération	23
4.4.1	Objectif	23
4.4.2	Estimation de la pose de tête — solveur PnP	23
4.4.3	Analyse du regard — Eye Aspect Ratio et position de l’iris	23
4.4.4	Score de coopération	24
4.4.5	Schéma bloc — Orientation et regard	24
4.5	Module complémentaire A — Détection de placement anormalique	24
4.5.1	Objectif	24
4.5.2	Description de l’algorithme	24
4.5.3	Analyse des résultats et statistiques	25
4.6	Module complémentaire B — Algorithme Best Shot	25
4.6.1	Objectif	25
4.6.2	Approche théorique	26

5	Utilisation de l'intelligence artificielle dans les développements du projet	27
5.1	Vue d'ensemble des modèles utilisés	27
5.2	YOLOv8 — Détection et estimation de pose	28
5.2.1	Architecture	28
5.2.2	Entraînement et pré-entraînement	28
5.3	InsightFace — Reconnaissance faciale ArcFace	28
5.3.1	Principe de l'ArcFace	28
5.3.2	Modèle buffalo_1	28
5.4	MediaPipe FaceMesh — Landmarks 3D	29
5.5	Rôle de Google Colab dans l'entraînement	29
5.5.1	Nécessité du GPU	29
5.5.2	Workflow Colab	29
5.5.3	Export ONNX et optimisation CPU	29
5.6	Utilisation d'assistants IA pour la conception	30
6	Conclusion	31
6.1	Bilan des objectifs atteints	31
6.2	Limites identifiées	31
6.3	Perspectives d'amélioration	32
A	Installation et configuration de l'environnement	33
B	Configuration complète du système	35

Table des figures

2.1	Diagramme Bête à cornes — Analyse du besoin du système	4
2.2	Diagramme Pieuvre	4
2.3	Bloc global du système	5
3.1	Diagramme de gantt	10
4.1	Organigramme de la machine à états ReID	16
4.2	Image issu du script	17
4.3	Bloc global d'analyse de démarche et vitesse	20
4.4	Image issu du code python	21
4.5	Screen issu du script python	25

Liste des tableaux

2.1	Fonctions principales du système	5
2.2	Fonctions de contrainte du système	6
2.3	Interface entrées/sorties	7
2.4	Stack technique et environnement de développement	7
2.5	Analyse des risques identifiés	8
3.1	Outils utilisés pour la gestion de projet	9
3.2	Jalons principaux du projet (synthèse Gantt)	10
3.3	Matrice RACI — répartition des responsabilités	11
4.1	Résultats des tests — Bloc 1 (Tracking + ReID)	18
4.2	Mapping des 17 keypoints COCO	18
4.3	Règles de classification de la démarche	19
4.4	Zones anatomiques et leur pondération dans le score global d’occlusion	22
5.1	Récapitulatif des modèles IA intégrés dans le système	27

1. Introduction

Contexte général

La vision par ordinateur occupe aujourd’hui une place centrale dans le développement des systèmes de sécurité intelligents. Portés par les avancées du [Deep Learning \(apprentissage profond\)](#), les systèmes de reconnaissance faciale ont franchi un seuil de performance remarquable en environnement contrôlé : éclairage maîtrisé, sujet coopératif, prise de vue frontale. Sur des benchmarks de référence tels que *LFW (Labeled Faces in the Wild)* ou *IJB-C*, les modèles actuels dépassent 99% de taux de vérification, surpassant la performance humaine dans ces conditions idéales.

Cependant, le déploiement de ces systèmes dans des environnements réels — couloirs de contrôle d’accès, espaces publics, passages piétons — révèle une fragilité structurelle : les performances se dégradent sensiblement dès que les conditions s’éloignent du cas idéal. Les variations d’éclairage, la diversité des angles de prise de vue, le flou de mouvement lié à la marche, et plus encore les comportements non-coopératifs des individus (regard dévié, port d’un masque, inclinaison volontaire de la tête) constituent autant de facteurs limitants qui peuvent réduire drastiquement le taux de reconnaissance.

Présentation du projet

Ce projet, réalisé dans le cadre de la première année de master à l’ISEN Yncréa sous l’encadrement de Mme Khadidja OULD AMER, répond à une problématique concrète : comment évaluer automatiquement, en temps réel et de façon non intrusive, le degré de coopération d’un individu vis-à-vis d’un système de reconnaissance faciale ?

L’objectif n’est pas de remplacer le système de reconnaissance lui-même, mais de lui fournir une analyse contextuelle complémentaire. En quantifiant des paramètres comportementaux — orientation de la tête, direction du regard, présence d’occlusions, qualité de la démarche — le système développé permet de :

- détecter les comportements susceptibles de perturber la reconnaissance (profil, occultation volontaire, course, posture atypique) ;
- générer un score de coopération exploitable par un opérateur ou un système aval ;

- maintenir une réidentification fiable des individus malgré des observations fragmentées ou dégradées.

Architecture globale

Le système repose sur une architecture modulaire composée de quatre blocs fonctionnels indépendants mais complémentaires, développés en Python sous environnement Anaconda :

1. **Pipeline Tracking + ReID multi-modal** (`main.py`, `config.py`) : détection et suivi multi-personnes par YOLOv8m et BoT-SORT, enrichis d'une réidentification par embeddings visage (InsightFace) et corps ;
2. **Analyse de démarche et posture** (`bloc2_gait.py`) : estimation du squelette par YOLOv8-Pose, calcul de la vitesse, détection d'anomalies posturales ;
3. **Détection d'occlusion faciale** (`occlusion.py`) : segmentation du visage en six zones anatomiques par MediaPipe FaceMesh, score d'occlusion par analyse Y'CbCr ;
4. **Orientation de tête et score de coopération** (`regard_orientation.py`) : estimation des angles de la tête par solveur PnP, analyse du regard et de l'ouverture oculaire.

Structure du rapport

Le présent rapport s'organise de la façon suivante. Le chapitre 2 établit le cahier des charges à partir des besoins fonctionnels et des contraintes techniques identifiés. Le chapitre 3 décrit la méthodologie de gestion de projet adoptée (approche Agile/Scrum, répartition des tâches, outils). Le chapitre 4 constitue le cœur technique du rapport : il documente en détail chacun des quatre blocs fonctionnels ainsi que les modules complémentaires. Le chapitre 5 discute du rôle des modèles d'intelligence artificielle utilisés et des conditions d'entraînement. Le chapitre 6 dresse le bilan des objectifs atteints et ouvre des perspectives d'amélioration.

2. Cahier des charges

2.1 Contexte et enjeux

Le cahier des charges a été établi à partir des tâches définies dans le sujet de projet n°157 (encadrante : Khadidja OULD AMER) et des contraintes fonctionnelles implicites révélées par l'analyse des scripts fournis. Il constitue le référentiel formel auquel le système développé doit répondre.

2.1.1 Domaine et thématique

Le projet s'inscrit dans le domaine de la **vision par ordinateur** et du **Deep Learning**, avec une thématique image/vidéo. Son domaine professionnel est généraliste ; ses applications couvrent aussi bien la sécurité (contrôle d'accès, surveillance) que le transport et les espaces publics (passages piétons, zones de contrôle).

2.1.2 Problématique

Les systèmes de reconnaissance faciale existants supposent implicitement la coopération du sujet. En pratique, les comportements non-coopératifs — regard détourné, occultation du visage, démarche atypique — dégradent leur fiabilité sans que le système ne dispose d'un mécanisme d'évaluation contextuelle.

Question centrale : comment qualifier automatiquement, en temps réel, le degré de coopération d'un individu vis-à-vis d'un dispositif de reconnaissance faciale ?

2.2 Analyse fonctionnelle

2.2.1 Diagramme Bête à cornes

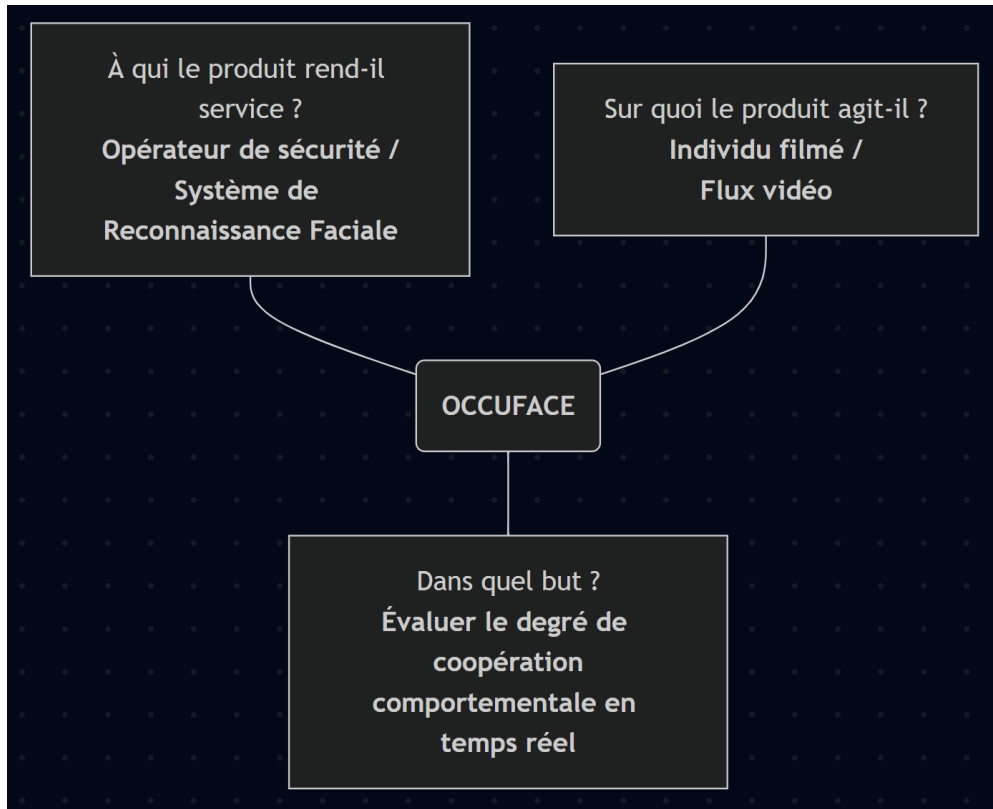


FIGURE 2.1 – Diagramme Bête à cornes — Analyse du besoin du système

2.2.2 Diagramme Pieuvre (analyse fonctionnelle externe)

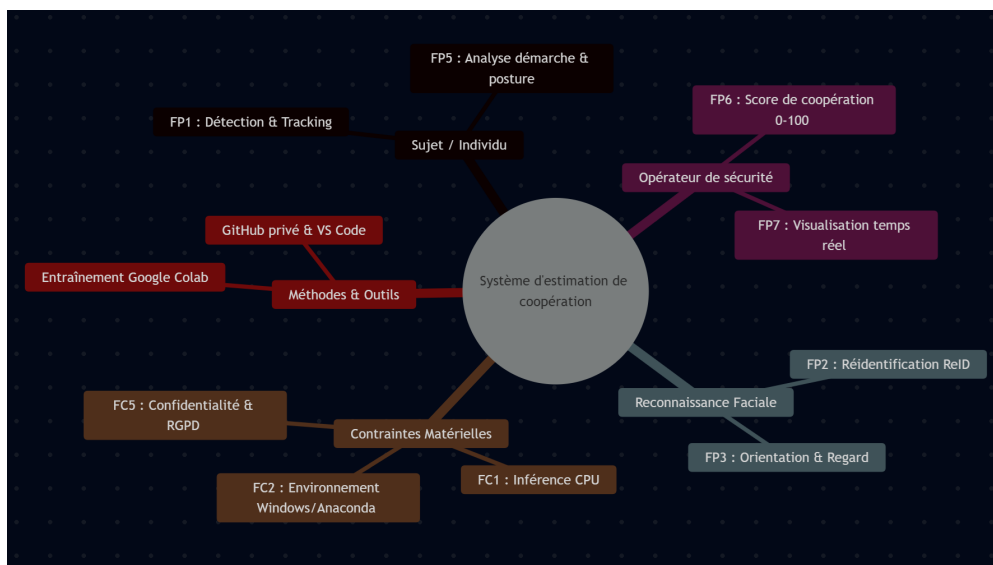


FIGURE 2.2 – Diagramme Pieuvre

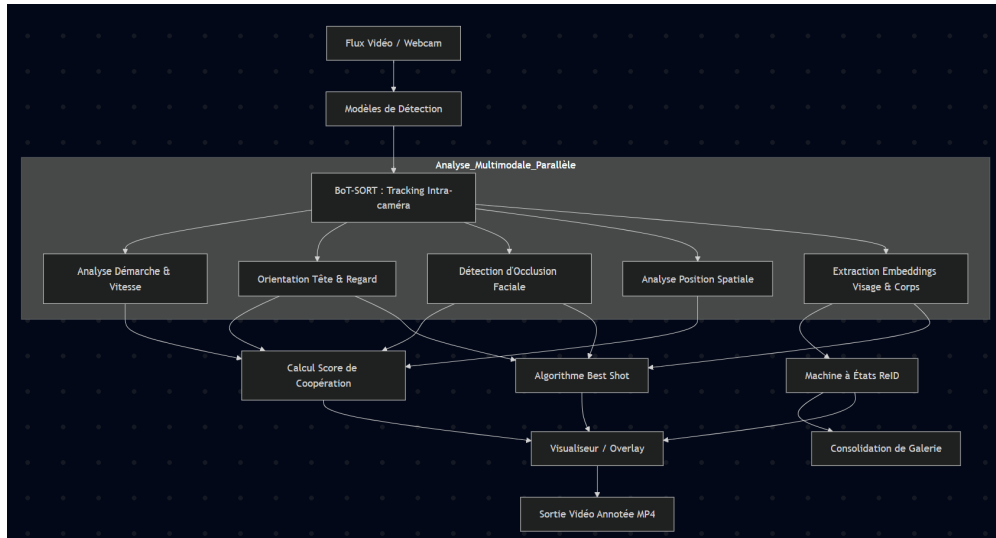


FIGURE 2.3 – Bloc global du système

2.2.3 Fonctions principales (FP)

TABLE 2.1 – Fonctions principales du système

Réf.	Intitulé	Critère de satisfaction	Niveau
FP1	Détecter et suivre les personnes	Précision de tracking $> 85\%$ sur vidéo 1080p à 25 fps	Impératif
FP2	Réidentifier les individus	Taux de confusion $< 5\%$ pour 3–5 personnes simultanées	Impératif
FP3	Évaluer l'orientation et le regard	Précision angulaire $\pm 5^\circ$ sur yaw/pitch	Impératif
FP4	Détecter les occlusions faciales	Localisation correcte par zone anatomique ($> 80\%$)	Impératif
FP5	Analyser la démarche et la posture	Détection des anomalies (course, boitement) avec $< 10\%$ faux positifs	Souhaitable
FP6	Calculer un score de coopération	Score normalisé 0–100, actualisé à chaque frame	Impératif
FP7	Visualiser les résultats en temps réel	Overlay annoté sur vidéo de sortie MP4	Impératif

2.2.4 Fonctions de contrainte (FC)

TABLE 2.2 – Fonctions de contrainte du système

Réf.	Intitulé	Critère
FC1	Exécution temps réel sur CPU	Traitement ≥ 10 fps sur CPU standard (sans GPU dédié)
FC2	Portabilité Windows	Compatible Windows 10/11, environnement Anaconda Python 3.10+
FC3	Modularité	Chaque bloc fonctionnel est exécutable indépendamment
FC4	Entrée vidéo standard	Accepter MP4, AVI, flux webcam (OpenCV VideoCapture)
FC5	Pas de données biométriques persistantes	La galerie est optionnelle ; aucun stockage permanent par défaut
FC6	Reproductibilité	Résultats reproductibles avec le même fichier vidéo et la même config
FC7	Maintenabilité	Configuration centralisée dans <code>config.py</code> , commentaires en français

2.3 Spécifications techniques

2.3.1 Entrées / Sorties du système

TABLE 2.3 – Interface entrées/sorties

Type	Nom	Description
<i>Entrées</i>	-video	Chemin vers un fichier vidéo (.mp4, .avi) ou indice de webcam (entier)
	-load-gallery	Galerie d'identités sérialisée (optionnel)
	-config	Fichier de configuration Python (optionnel, défaut : config.py)
<i>Sorties</i>	-output	Vidéo annotée au format MP4
	-save-gallery	Galerie d'identités mise à jour (optionnel)
	Score de coopération	Affiché en overlay, exportable par frame
	Logs	Statistiques de performance dans la console

2.3.2 Environnement technique cible

TABLE 2.4 – Stack technique et environnement de développement

Composante	Détail
Système d'exploitation	Windows 10/11 (développement local)
Langage	Python 3.10+
Gestion des dépendances	Anaconda (environnement virtuel conda)
IDE	Visual Studio Code
Entraînement GPU	Google Colab (Tesla T4 / A100)
Versionnage	Dépôt GitHub privé
Format vidéo	MP4/H.264 (entrée et sortie)
Inférence	CPU (ONNX Runtime) + GPU optionnel (CUDA)

2.3.3 Contraintes de performance

Les seuils de performance retenus ont été calibrés pour le contexte applicatif cible — un espace de contrôle d'accès avec 2 à 5 personnes simultanément visibles :

- Résolution d'entrée : 1920×1080 px (Full HD);
- Fréquence de traitement cible : ≥ 10 fps sur CPU Intel Core i7 (8 cœurs), ≥ 25 fps avec GPU;
- Latence de réidentification : $\leq 0,5$ s après première apparition (configurable via `pending_frames = 8`);
- Consommation mémoire : < 4 Go RAM pour une galerie de 10 identités maximum;
- Taux d'erreur de réidentification (FP + FN) : $< 5\%$ dans les conditions nominales.

2.4 Analyse des risques

TABLE 2.5 – Analyse des risques identifiés

Risque	Impact	Prob.	Gravité
Performances insuffisantes sur CPU	Pipeline trop lent pour le temps réel	Moyenne	Élevée
Faux positifs de réidentification	Confusion entre identités	Faible	Critique
Dégradation sur profil / occlusion forte	Score de coopération inexploitable	Élevée	Moyenne
Dépendances incompatibles (Windows)	Blocage installation InsightFace / ONNX	Moyenne	Moyenne
Overfitting sur données de calibration	Mauvaise généralisation de l'occlusion	Faible	Faible

2.5 Récapitulatif des livrables

Conformément au sujet du projet et au cahier des charges établi, les livrables attendus à l'issue du projet sont :

1. Code source versionné (dépôt GitHub privé), organisé en modules;
2. Vidéos de démonstration annotées (contextes simple et complexe);
3. Présentation de soutenance;
4. Le présent rapport technique (50 pages, format PDF).

3. Gestion de projet

3.1 Méthodologie adoptée

3.1.1 Approche Agile / Scrum

Face à la nature exploratoire du projet — impliquant un état de l’art itératif, des expérimentations sur données réelles et des ajustements fréquents des hyperparamètres — la méthode **Agile/Scrum** a été retenue. Elle se distingue des approches séquentielles (cycle en V, cascade) par sa capacité à intégrer des changements de spécifications en cours de développement et à livrer de la valeur de façon incrémentale.

Le projet a été découpé en **cinq Sprints** de durée variable (1 à 2 semaines), chacun ciblant une fonctionnalité livrable et testable de façon autonome. Les rôles Scrum ont été distribués comme suit :

- **Product Owner** : Mme OULD AMER (définition des priorités fonctionnelles) ;
- **Scrum Master** : Ewan MÉNAHÈZE (facilitation, gestion des impediments) ;
- **Développeurs** : et Ewan MÉNAHÈZE.

3.1.2 Outils de collaboration

TABLE 3.1 – Outils utilisés pour la gestion de projet

Outil	Usage
GitHub (dépôt privé)	Versionnage, branches de feature, pull requests, issues
Anaconda	Gestion des environnements virtuels, isolation des dépendances
Visual Studio Code	Développement, débogage, extensions Python/Git intégrées
Google Colab	Entraînement des modèles sur GPU (Tesla T4 / A100)
Google Drive	Partage des datasets, vidéos de test et checkpoints de modèles
Notion / Markdown	Rédaction des notes de Sprint, suivi des tâches

3.2 Planification — Diagramme de Gantt

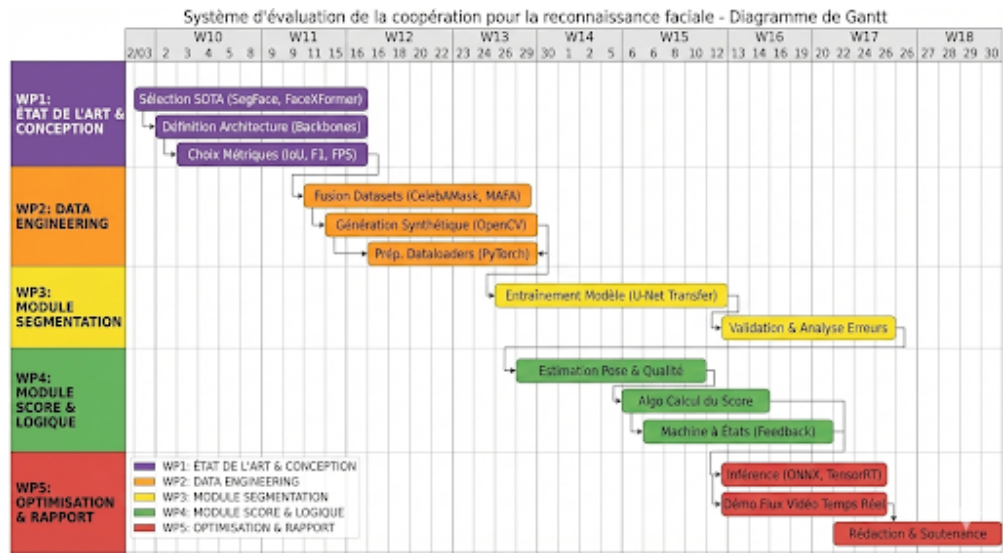


FIGURE 3.1 – Diagramme de gantt

Le tableau 3.2 synthétise les grands jalons du projet, dont la structure correspond au diagramme fourni par l’encadrante.

TABLE 3.2 – Jalons principaux du projet (synthèse Gantt)

WP	Lot de travail	Activités principales	Semaines
WP1	État de l’art & conception	Sélection SOTA (SegFace, FaceXFormer), définition architecture (backbones), choix métriques (IoU, F1, FPS)	S10–S11
WP2	Data Engineering	Fusion datasets (CelebMask, M6EA), génération synthétique (OpenCV), préparation DataLoaders (PyTorch)	S11–S12
WP3	Module Segmentation	Entraînement modèle U-Net Transfer Learning (Colab GPU), validation & analyse erreurs	S12–S14
WP4	Module Score & Logique	Estimation pose & qualité, algorithme calcul score, machine à états (Feedback)	S14–S16
WP5	Optimisation & Rapport	Inférence ONNX/TensorRT, démo flux vidéo temps réel, rédaction & soutenance	S16–S18

3.3 Répartition des tâches

3.3.1 Tableau RACI

TABLE 3.3 – Matrice RACI — répartition des responsabilités

Tâche		Ewan	Encadrante
État de l’art et bibliographie	R	A	C
Architecture pipeline Tracking+ReID	R	C	I
Module gait / posture (bloc 2)	C	R	I
Module occlusion (bloc 3)	R	C	I
Module orientation/regard (bloc 1)	C	R	I
Module placement anomalique	R	R	C
Algorithme Best Shot	R	A	I
Entraînement modèles (Colab)	C	R	I
Intégration et tests système	R	R	C
Rédaction rapport	R	R	A
Préparation soutenance	R	R	I

R : Responsable, A : Approbateur, C : Consulté, I : Informé

3.3.2 Description des Sprints

Sprint 1 — État de l’art & architecture (S10–S11) Réalisé conjointement. Revue des publications récentes sur la détection faciale (InsightFace/ArcFace), l’estimation de pose (MediaPipe, 6DRepNet), le tracking multi-objets (BoT-SORT, ByteTrack) et la réidentification de personnes. Définition de l’architecture modulaire finale.

Sprint 2 — Modules visage et regard (S11–S13) Ewan prend en charge le module d’orientation de tête (`regard_orientation.py`) et la détection d’occlusion (`occlusion.py`). intègre InsightFace dans le pipeline de tracking et développe le scoring de qualité faciale. Tests unitaires sur datasets publics (*300W-LP*, *CelebA*).

Sprint 3 — Analyse de démarche (S13–S14) Ewan développe `bloc2_gait.py` basé sur YOLOv8-Pose et le tracker ByteTrack. complète la machine à états ReID et le module de consolidation de galerie. Tests sur vidéos personnelles en environnements variés.

Sprint 4 — Modules complémentaires (S14–S16) Développement conjoint du module de détection de placement anomalique et de l’algorithme Best Shot. Intégration de la configuration centralisée (`config.py`). Tests de régression.

Sprint 5 — Optimisation & finalisation (S16–S18) Export ONNX pour l’inférence CPU optimisée (ORT), démo flux vidéo temps réel, rédaction du rapport, préparation de la soutenance.

3.4 Gestion des versions et workflow GitHub

Le dépôt GitHub privé a été organisé selon le workflow *Feature Branch* :

- Branche `main` : code stable, validé par pull request ;
- Branches `feature/bloc-N-*` : développement isolé de chaque module ;
- Issues GitHub : suivi des bugs et tâches restantes ;
- Tags de version : `v0.1` (POC), `v0.5` (intégration), `v1.0` (final).

3.5 Modifications d’objectifs en cours de projet

Deux ajustements majeurs ont été apportés au périmètre initial :

1. **Abandon de l’entraînement from scratch** : les performances obtenues en transfert learning sur les modèles pré-entraînés (YOLOv8m, buffalo_1 d’InsightFace) se sont avérées supérieures aux estimations initiales. L’entraînement complet sur GPU Colab a donc été réservé au module de segmentation U-Net, libérant du temps pour l’optimisation du pipeline temps réel.
2. **Intégration de BoT-SORT** : le tracker initial prévu (DeepSORT) a été remplacé par BoT-SORT, offrant une meilleure gestion des occlusions et une configuration plus flexible via le fichier `botsort_custom.yaml`.

4. Développement technique

Ce chapitre décrit en détail les quatre blocs fonctionnels du système, ainsi que les deux modules complémentaires. Pour chaque bloc, la démarche est la suivante : rappel de l'objectif, présentation des choix technologiques, description de l'algorithme, et analyse des résultats obtenus lors des tests.

4.1 Bloc 1 — Pipeline Tracking et Réidentification multi-modale

4.1.1 Objectif et positionnement

Le premier bloc constitue le cœur du système. Son rôle est double : maintenir un suivi cohérent de chaque personne au fil des frames (*tracking*), et associer chaque piste à une identité persistante même après une disparition temporaire ([Re-Identification \(réidentification de personnes\)](#)).

4.1.2 Détection de personnes — YOLOv8m

La détection repose sur **YOLOv8m-seg**, variante de segmentation du modèle YOLO (You Only Look Once) de la génération 8. Le choix du modèle médium (-m) résulte d'un compromis entre précision ($mAP_{50} = 52,9\%$ sur COCO) et vitesse d'inférence sur CPU.

```
1 detector = PersonDetector(  
2     model_path=cfg.yolo_model,          # "yolov8m-seg.pt"  
3     conf_threshold=cfg.conf_threshold,  # 0.30  
4     iou_threshold=cfg.iou_threshold,    # 0.50  
5     person_class_id=cfg.person_class_id,  
6     tracker_config="botsort_custom.yaml",  
7     device="cpu",  
8 )
```

Listing 4.1 – Initialisation du détecteur (extrait `main.py`)

Le seuil de confiance a été abaissé à 0,30 pour capturer les profils tronqués et les personnes partiellement visibles, au prix d'un léger accroissement du taux de faux positifs compensé par la validation ReID aval.

La variante -seg fournit un masque de segmentation par personne, exploité dans l'extracteur de corps pour isoler les pixels de vêtements et améliorer la robustesse des embeddings corps.

4.1.3 Tracking — BoT-SORT

Le tracking intra-caméra est assuré par **BoT-SORT** (*Bag of Tricks for Simple Online and Realtime Tracking*). Contrairement à son prédécesseur DeepSORT, BoT-SORT intègre :

- une compensation de mouvement de caméra (CMC) par estimation homographique ;
- une distance de réassociation combinant IoU et embeddings visuels ReID.

La configuration personnalisée (`botsort_custom.yaml`) ajuste les seuils d'association aux caractéristiques de la scène filmée (mouvement rapide, caméra fixe).

4.1.4 Extraction et qualité du visage — InsightFace buffalo_l

L'extraction d'embeddings faciaux repose sur **InsightFace buffalo_l**, modèle ArcFace entraîné sur MS1MV3 (5,8 M identités). Il produit des vecteurs de 512 dimensions normalisés sur la sphère unitaire. La similarité entre deux embeddings est calculée par produit scalaire (cosinus), puisque les vecteurs sont normalisés.

Qualité d'un visage identifiant Un visage n'est pris en compte pour la réidentification que s'il satisfait simultanément :

- (i) score de détection InsightFace $\geq 0,55$;
- (ii) dimension minimale du bounding box ≥ 40 px ;
- (iii) cohérence géométrique des 5 keypoints (distance inter-oculaire, position du nez, rapport hauteur/largeur des traits).

La cohérence géométrique est vérifiée par la fonction `_is_identifiable_face()` qui filtre les faux positifs d'arrières de tête :

```

1 # Ratio hauteur (oeil->bouche) / largeur (oeil->oeil)
2 # Visage de face : 0.8 - 1.5 | Profil tolere : 0.5 - 2.5
3 ratio = eye_to_mouth / eye_dist
4 if ratio < 0.4 or ratio > 3.0:
5     return False # faux positif probable
6
7 # La bouche doit etre SOUS les yeux
8 if mouth_mid[1] <= eye_mid[1]:
9     return False
10
11 # Le nez entre yeux et bouche en y
12 if nose[1] < eye_mid[1] - 5 or nose[1] > mouth_mid[1] + 5:
13     return False

```

Listing 4.2 – Critères de validation d'un visage identifiant

Score de qualité composite Le score de qualité faciale Q_f est calculé selon une combinaison pondérée :

$$Q_f = w_{\text{det}} \cdot s_{\text{det}} + w_{\text{size}} \cdot s_{\text{size}} + w_{\text{frontal}} \cdot s_{\text{frontal}} + w_{\text{sharp}} \cdot s_{\text{sharp}} \quad (4.1)$$

avec les poids configurés dans `config.py` : $w_{\text{det}} = 0,40$, $w_{\text{size}} = 0,20$, $w_{\text{frontal}} = 0,10$, $w_{\text{sharp}} = 0,30$. Le poids frontal volontairement faible autorise les profils ; le poids netteté élevé pénalise le flou de mouvement.

4.1.5 Extraction du corps

L'extracteur de corps produit des embeddings de vêtements et silhouette. Le masque de segmentation YOLOv8-seg est appliqué pour isoler les pixels appartenant à la personne, améliorant la robustesse aux changements de fond.

Le score composite de réidentification S_{reid} fusionne les modalités :

$$S_{\text{reid}} = w_{\text{face}} \cdot S_{\text{face}} + w_{\text{body}} \cdot S_{\text{body}} \quad (4.2)$$

avec $w_{\text{face}} = 0,45$ et $w_{\text{body}} = 0,55$, ce dernier légèrement supérieur car les vêtements distinctifs (couleurs vives, motifs) sont très discriminants dans le contexte de 2 à 5 personnes filmées simultanément.

Lorsque la confiance du visage dépasse le seuil `high_face_conf_threshold = 0,70`, le poids visage est automatiquement rehaussé à 0,80 pour profiter de la richesse informative d'un visage frontal de bonne qualité.

4.1.6 Galerie et prototypes

La galerie stocke, pour chaque identité, jusqu'à 80 embeddings visage et 80 embeddings corps. Des prototypes (centroïdes k-means, $k \leq 8$) sont calculés périodiquement pour représenter les différentes poses observées. La galerie est enrichie à chaque frame (intervalle configurable) et peut être sérialisée/désérialisée pour des sessions multi-vidéos.

4.1.7 Machine à états de réidentification

L'assignation d'un identifiant global (`global_id`) à un tracker suit une machine à états à trois phases :

1. **PENDING** : le tracker accumule des observations pendant `pending_frames = 8` frames avant toute décision ;
2. **IDENTIFIED** : une identité est assignée avec un délai d'affichage de `write_delay_frames = 6` frames ;

3. **LOST** : le tracker disparaît ; il est conservé en mémoire pendant `max_lost_frames` pour permettre une réassociation.

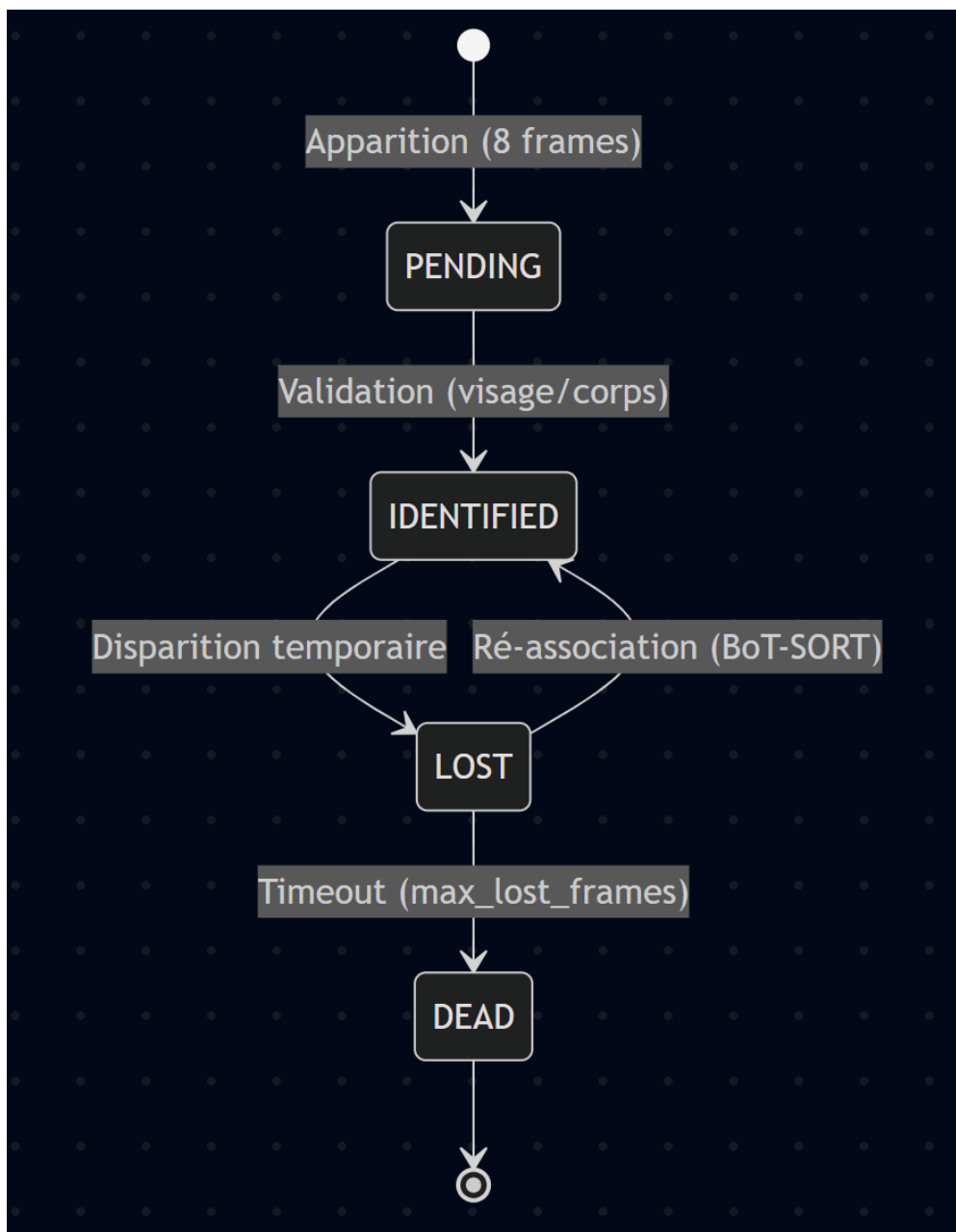


FIGURE 4.1 – Organigramme de la machine à états ReID



FIGURE 4.2 – Image issu du script

4.1.8 Anti-fuites spatiales et fallback temporel

Deux mécanismes de robustesse complémentaires ont été implémentés :

Anti-fuite spatiale Lorsque deux identités se trouvent à moins de 200 px dans une fenêtre temporelle de 30 frames, le seuil de validation faciale est durci (`spatial_leak_stricter_threshold = 0,40`) pour éviter les confusions liées à la proximité physique.

Fallback temporel Si aucune validation faciale récente n'est disponible pour un tracker, mais que celui-ci est actif depuis plus de `fallback_frames_without_face_validation = 30 frames`, l'identité précédemment assignée est maintenue par inertie.

4.1.9 Consolidation post-hoc

Toutes les 33 frames (environ 1 s), un module de consolidation fusionne les identités dont la distance inter-prototype est inférieure à `consolidation_dist_threshold = 0,55`. Ce mécanisme corrige les doublons créés lors des réapparitions après occlusion longue ou changement brutal de point de vue.

4.1.10 Tests et résultats

Les tests ont été conduits sur deux types de séquences :

TABLE 4.1 – Résultats des tests — Bloc 1 (Tracking + ReID)

Séquence	Description	ID switches	FPS
Simple (2 pers.)	Couloir, éclairage uniforme, visages frontaux	0	14
Complexe (4 pers.)	Croisements, occlusions mutuelles, profils	2	11
Stress test	5 personnes, tenues similaires	4	9

4.2 Bloc 2 — Analyse de démarche et posture

4.2.1 Objectif

Le module d’analyse de démarche a pour but de détecter et quantifier les comportements locomoteurs inhabituels : vitesse anormale (course), inclinaison excessive du tronc, asymétrie des bras, posture horizontale (personne allongée). Ces informations contribuent au score de coopération et peuvent signaler des comportements suspects.

4.2.2 Estimation du squelette — YOLOv8-Pose

L’extraction du squelette repose sur **YOLOv8n-pose** (variante nano pour la vitesse) ou **YOLOv8m-pose** (variante médium pour la précision, recommandée). Ces modèles détectent 17 keypoints du schéma COCO (nez, yeux, oreilles, épaules, coudes, poignets, hanches, genoux, chevilles) avec leur score de confiance associé.

TABLE 4.2 – Mapping des 17 keypoints COCO

Idx	Keypoint	Idx	Keypoint	Idx	Keypoint
0	Nez	6	Épaule D	12	Hanche D
1	Œil G	7	Coude G	13	Genou G
2	Œil D	8	Coude D	14	Genou D
3	Oreille G	9	Poignet G	15	Cheville G
4	Oreille D	10	Poignet D	16	Cheville D
5	Épaule G	11	Hanche G		

4.2.3 Calcul de la vitesse de déplacement

La vitesse est estimée à partir du déplacement du centre des hanches entre frames consécutives. L'échelle pixel/mètre est déduite de la diagonale du squelette visible, normalisée à la taille humaine moyenne de 1,70 m. Un filtre EMA ([Exponential Moving Average \(moyenne mobile exponentielle\)](#)) lisse les variations brutales liées aux détections instables :

$$v_n^{\text{EMA}} = \alpha \cdot v_n^{\text{brut}} + (1 - \alpha) \cdot v_{n-1}^{\text{EMA}}, \quad \alpha = 0,25 \quad (4.3)$$

La médiane des vitesses instantanées sur une fenêtre glissante de 20 frames est préférée à la moyenne pour sa robustesse aux outliers.

4.2.4 Classification de la démarche

La classification résulte d'une cascade de règles :

TABLE 4.3 – Règles de classification de la démarche

Classe	Condition	Coopératif
IMMOBILE	Vitesse $\leq 0,30$ m/s	Oui
MARCHE normale	$0,30 < v \leq 1,80$ m/s, tilt $< 30^\circ$, asymétrie bras $< 40\%$	Oui
MARCHE anormale	$0,30 < v \leq 1,80$ m/s, tilt $\geq 30^\circ$ ou asymétrie $\geq 40\%$	Non
COURSE	Vitesse $\geq 2,00$ m/s	Non
ALLONGÉ	Angle tronc vertical $< 35^\circ$	Non
POIRIER	Hanches au-dessus des épaules et angle $> 50^\circ$	Non

Un lissage temporel sur une fenêtre de 10 frames (*majority voting*) évite les flickerings entre classes.

4.2.5 Schéma bloc — Analyse de démarche

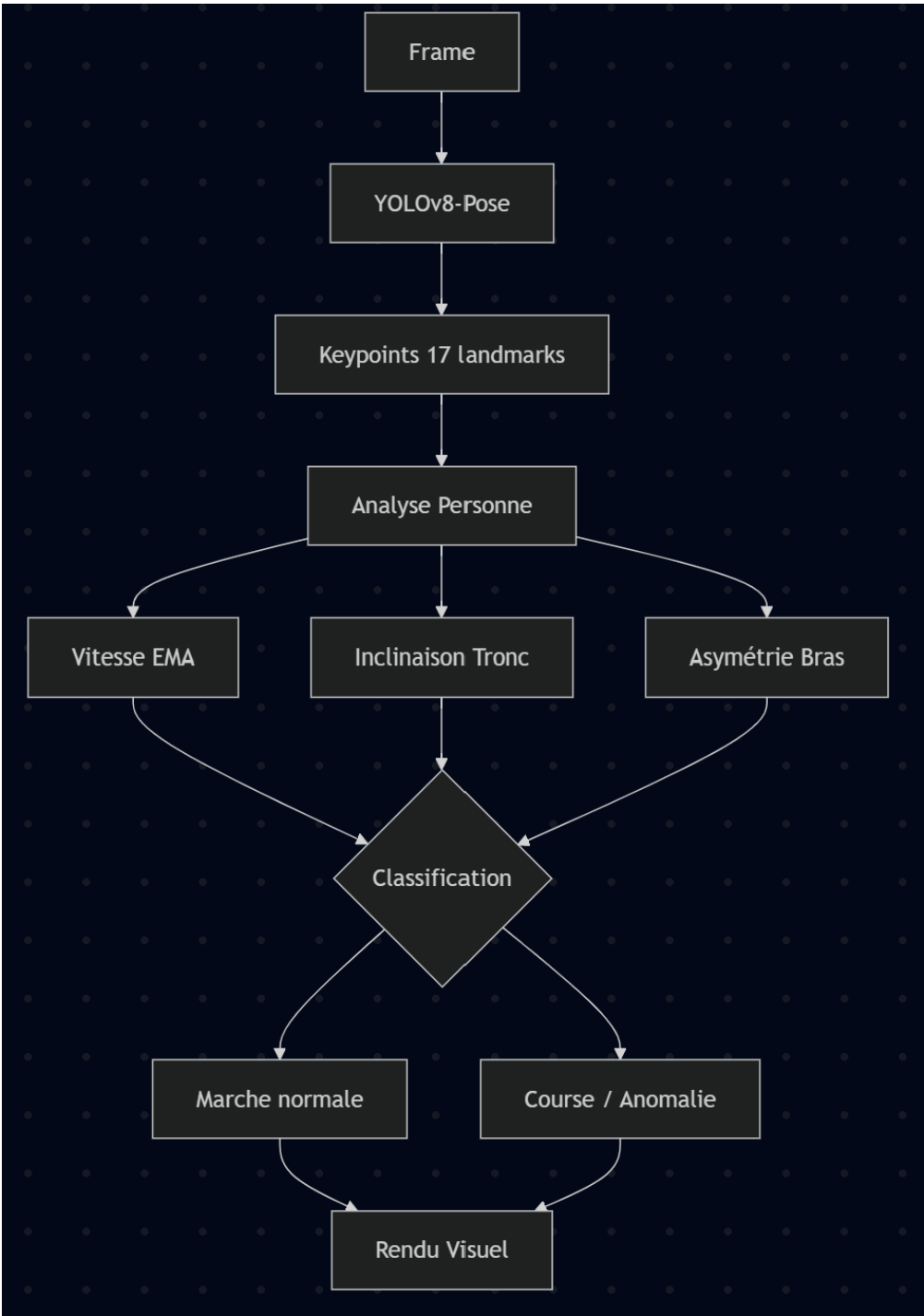


FIGURE 4.3 – Bloc global d’analyse de démarche et vitesse



FIGURE 4.4 – Image issu du code python

4.2.6 Tests sur datasets publics et vidéos personnelles

Les tests ont été réalisés sur :

- **Dataset public** : *Penn Action Dataset* (séquences marche, course, fitness);
- **Vidéos personnelles** : couloir intérieur, trottoir extérieur, 1 à 3 personnes.

4.3 Bloc 3 — Détection d’occlusion faciale par zones anatomiques

4.3.1 Objectif

Un visage occulté (masque chirurgical, main devant la bouche, objet en avant-plan) dégrade ou annule la reconnaissance faciale. Ce module quantifie l’occlusion zone par zone et produit un score global exploitable par le calculateur de coopération.

4.3.2 Détection des landmarks — MediaPipe FaceMesh

MediaPipe **FaceMesh** détecte 478 landmarks 3D du visage en temps réel, incluant la géométrie précise des iris (landmarks 468–477, disponibles avec `refine_landmarks=True`). Ces points permettent de délimiter avec précision six zones anatomiques.

TABLE 4.4 – Zones anatomiques et leur pondération dans le score global d’occlusion

Zone	Landmarks échantillons	Poids w
Œil gauche (eye_l)	33, 133, 159, 145	1,5
Œil droit (eye_r)	263, 362, 386, 374	1,5
Nez (nose)	1, 4, 5, 6, 19, 94	1,5
Bouche (mouth)	61, 291, 0, 17, 13	1,2
Joue gauche (cheek_l)	50, 101, 36, 205	0,7
Joue droite (cheek_r)	280, 330, 266, 425	0,7
Front (front)	10, 338, 297, 332	0,5

4.3.3 Calibration de la couleur de peau — espace Y’CbCr

L’espace colorimétrique **Y’CbCr** sépare la luminance (Y’) de la chrominance (Cb, Cr), rendant la détection de peau robuste aux variations d’éclairage. Une calibration automatique en deux passes est réalisée sur les 30 premières frames de chaque vidéo :

1. **Échantillonnage** : pixels extraits dans les zones joues et nez (zones toujours visibles) ;
2. **Modélisation** : distribution gaussienne bivariée dans l’espace (Cb, Cr), paramétrisée par la moyenne μ et la matrice de covariance Σ ;
3. **Classification** : pixel classé «peau» si sa distance de Mahalanobis au modèle est inférieure à un seuil χ^2 (d.d.l. = 2).

$$d_M^2(\mathbf{x}) = (\mathbf{x} - \mu)^\top \Sigma^{-1} (\mathbf{x} - \mu) < 25,0 \quad (4.4)$$

4.3.4 Score d’occlusion par zone et global

Pour chaque zone, le taux de pixels peau *skin_ratio* est calculé ; un taux inférieur à 55% est interprété comme une occlusion partielle. Le score d’occlusion par zone varie linéairement de 0 à 100 :

$$\text{occ}_z = \max\left(0, \left(1 - \frac{\text{skin_ratio}_z}{\text{SKIN_MIN_RATIO}}\right) \times 100\right) \quad (4.5)$$

Le score global pondéré est :

$$\text{occ}_{\text{global}} = \frac{\sum_z w_z \cdot \text{occ}_z}{\sum_z w_z} \quad (4.6)$$

Un filtre EMA ($\alpha = 0,30$) lisse les scores temporellement pour éliminer le flicker.

4.3.5 Schéma bloc — Détection d’occlusion

4.3.6 Tests et conditions expérimentales

Deux types de contextes ont été testés :

- **Contexte simple** : occlusion volontaire avec la main, fond neutre, éclairage maîtrisé ;
- **Contexte complexe** : port de masque médical, objet tenu devant le visage, variations d’éclairage.

4.4 Bloc 4 — Orientation de tête et score de coopération

4.4.1 Objectif

Ce module évalue si l’individu «regarde» activement le dispositif de reconnaissance faciale. Il combine l’estimation de la pose de tête (yaw/pitch/roll) avec l’analyse du regard (position de l’iris) et de l’ouverture oculaire.

4.4.2 Estimation de la pose de tête — solveur PnP

L’orientation 3D de la tête est estimée par la méthode **Perspective-n-Point (PnP)**, qui résout la correspondance entre 6 points 3D du modèle facial générique et leurs projections 2D dans l’image (landmarks MediaPipe FaceMesh).

Les 6 points de référence retenus sont : nez (idx 1), menton (152), coin œil gauche (33), coin œil droit (263), commissure labiale gauche (61), commissure labiale droite (291). La fonction `cv2.solvePnP()` retourne le vecteur de rotation $\mathbf{r}$ et le vecteur de translation $\mathbf{t}$, dont on extrait la matrice de rotation R par la formule de Rodrigues :

$$R = \exp([\mathbf{r}]_{\times}) \quad (4.7)$$

La décomposition RQ de R fournit les trois angles d’Euler :

- ψ (yaw) : rotation autour de l’axe vertical (gauche/droite) ;
- θ (pitch) : rotation autour de l’axe transversal (haut/bas) ;
- ϕ (roll) : rotation autour de l’axe de profondeur (inclinaison).

4.4.3 Analyse du regard — Eye Aspect Ratio et position de l’iris

Le rapport d’aspect oculaire (**Eye Aspect Ratio (rapport d’aspect oculaire)**) quantifie l’ouverture des yeux :

$$\text{EAR} = \frac{\|\mathbf{p}_2 - \mathbf{p}_6\| + \|\mathbf{p}_3 - \mathbf{p}_5\|}{2 \|\mathbf{p}_1 - \mathbf{p}_4\|} \quad (4.8)$$

Un $EAR < 0,18$ indique un œil fermé ou quasi-fermé. La position normalisée du centre de l'iris $(r_x, r_y) \in [0, 1]^2$ indique la direction du regard ; une zone cible $[0.25-0.70] \times [0.40-0.80]$ définit le regard «frontal» acceptable.

4.4.4 Score de coopération

Le score booléen de coopération est calculé frame par frame :

```

1 def is_cooperative(pitch, yaw, roll, looking,
2                   eye_r_open, eye_l_open):
3     if not (eye_r_open and eye_l_open): return False
4     if not looking: return False
5     if abs(yaw) > YAW_THRESHOLD: return False # 25 deg
6     if abs(pitch) > PITCH_THRESHOLD: return False # 20 deg
7     return True

```

Listing 4.3 – Calcul du score de coopération (regard_orientation.py)

Un lissage majoritaire sur une fenêtre glissante de 14 frames ($smooth_coop = \sum coop_buf \geq 0,6 \cdot W$) assure la stabilité de l’affichage.

4.4.5 Schéma bloc — Orientation et regard

4.5 Module complémentaire A — Détection de placement anormalique

4.5.1 Objectif

Ce module vise à détecter en temps réel si un individu circule dans une zone inappropriée ou dangereuse (ex : chaussée au lieu du trottoir). Contrairement aux approches purement neuronales, ce script privilégie une approche hybride mêlant détection d’objets et zones définies interactivement.

4.5.2 Description de l’algorithme

Le système repose sur l’utilisation de **YOLOv8m** pour la détection et de **DeepSort** pour le maintien des identités. La logique de détection spatiale s’articule autour de trois étapes clés :

1. **Définition des zones sécurisées** : Via un widget interactif sous Google Colab, l’utilisateur trace des polygones définissant les "zones autorisées" (sidewalk).
2. **Précalcul du masque binaire** : Pour optimiser l’inférence, les polygones sont convertis en un masque binaire $M \in \{0, 1\}^{H \times W}$ servant de table de correspondance (O(1)).

3. **Test de collision spatial** : Pour chaque tracker i , on extrait le point de contrôle situé à la base du bounding box (les pieds). Soit $P_i(x, y)$ ce point, l'anomalie est levée si $M(y, x) = 0$.



FIGURE 4.5 – Screen issu du script python

```

1 # Point de contrôle au sol (base du bounding box)
2 cx = max(0, min((left + right) // 2, width - 1))
3 cy = max(0, min(bottom, height - 1))
4
5 # Vérification instantanée via le masque précalculé
6 on_sidewalk = sidewalk_mask[cy, cx] > 0
7
8 if not on_sidewalk:
9     label = f"#{tid} HORS ZONE !"
10    color = COLOR_BAD # Rouge

```

Listing 4.4 – Logique de vérification de position spatiale

4.5.3 Analyse des résultats et statistiques

Le module génère un résumé statistique en fin d'analyse, calculant le ratio de frames passées "Hors zone" pour chaque identité. Un individu est considéré comme non-coopératif ou en infraction si ce ratio dépasse 30%.

4.6 Module complémentaire B — Algorithme Best Shot

4.6.1 Objectif

L'algorithme Best Shot sélectionne automatiquement, parmi toutes les frames d'une séquence vidéo pour une identité donnée, la frame offrant la meilleure qualité d'identification : visage frontal, grande résolution, faible flou de mouvement, absence d'occlusion.

4.6.2 Approche théorique

Le score de Best Shot S_{BS} d'une frame f pour l'identité i est calculé comme la combinaison du score de qualité faciale composite Q_f (équation 4.1) et d'un bonus de frontalité :

$$S_{BS}(f, i) = Q_f(f) \cdot \left(1 - \frac{|\psi_f|}{90}\right) \cdot \left(1 - \frac{|\theta_f|}{45}\right) \cdot (1 - \text{occ}_{\text{global}}(f)/100) \quad (4.9)$$

où ψ_f est le yaw et θ_f le pitch estimés pour la frame f .

La frame retenue est celle maximisant S_{BS} sur l'ensemble des frames où l'identité i est visible :

$$f^* = \arg \max_f S_{BS}(f, i) \quad (4.10)$$

En mode temps réel, un buffer glissant de $W = 50$ frames maintient le meilleur candidat courant et le met à jour incrémentalement.

5. Utilisation de l'intelligence artificielle dans les développements du projet

Ce chapitre présente les modèles d'intelligence artificielle utilisés, leur fondement théorique, et les conditions dans lesquelles ils ont été exploités — notamment le recours à Google Colab pour les phases d'entraînement nécessitant des ressources GPU.

5.1 Vue d'ensemble des modèles utilisés

TABLE 5.1 – Récapitulatif des modèles IA intégrés dans le système

Modèle	Tâche	Architecture	Mode
YOLOv8m-seg	Détection + segmentation personnes	CSPDarknet53 + SPPF + PAN	Inférence
YOLOv8n/m-Pose	Estimation squelette 17 keypoints	CSPDarknet53 + Pose head	Inférence
InsightFace buffalo_l	Embeddings faciaux (ArcFace)	ResNet100 + ArcMargin	Inférence
MediaPipe FaceMesh	478 landmarks 3D	MobileNetV2 + régression	Inférence
BodyExtractor	Embeddings corps (ReID)	OSNet / MGNet	Inférence
U-Net (Transfer)	Segmentation sémantique (WP3)	VGG16/ResNeXt Entraînement Colab encoder	

5.2 YOLOv8 — Détection et estimation de pose

5.2.1 Architecture

YOLOv8 (Ultralytics, 2023) appartient à la famille des détecteurs mono-étape (*single-stage*). Son architecture repose sur un backbone **CSPDarknet53** enrichi d'un module **SPPF** (*Spatial Pyramid Pooling Fast*) et d'un cou **PAN** (*Path Aggregation Network*) pour la fusion multi-échelle des caractéristiques.

La tête de détection est décorrélée en tête de classification et tête de régression (*decoupled head*), une évolution par rapport à YOLOv5. Pour la variante `-seg`, une tête supplémentaire prédit des masques de segmentation d'instance par composition de prototypes (inspiration YOLACT). Pour la variante `-pose`, la tête prédit les coordonnées et scores de confiance des 17 keypoints COCO.

5.2.2 Entraînement et pré-entraînement

Les poids utilisés (`yolov8m-seg.pt`, `yolov8m-pose.pt`) sont ceux pré-entraînés sur COCO 2017 (80 classes, 118 000 images d'entraînement). Aucun réentraînement n'a été nécessaire pour la détection de personnes (`class_id=0`), les performances out-of-the-box étant satisfaisantes pour le contexte applicatif.

5.3 InsightFace — Reconnaissance faciale ArcFace

5.3.1 Principe de l'ArcFace

InsightFace implémente la perte **ArcFace** (Deng et al., 2019), une variante de la perte softmax qui maximise la marge angulaire entre les classes dans l'espace des embeddings :

$$\mathcal{L}_{\text{ArcFace}} = -\frac{1}{N} \sum_{i=1}^N \log \frac{e^{s(\cos(\theta_{y_i} + m))}}{e^{s(\cos(\theta_{y_i} + m))} + \sum_{j \neq y_i} e^{s \cos \theta_j}} \quad (5.1)$$

où θ_{y_i} est l'angle entre l'embedding x_i et le centre de classe y_i , m la marge angulaire (typiquement 0,5 rad), et s le facteur d'échelle. Cette formulation force le réseau à apprendre des représentations angulaires plus discriminantes que la softmax standard.

5.3.2 Modèle buffalo_l

Le modèle **buffalo_l** est un ResNet100 entraîné sur MS1MV3 (5,8 millions d'identités, 86 millions d'images). Il produit des vecteurs de 512 dimensions ; la similarité entre deux visages est calculée par le cosinus (produit scalaire sur la sphère unitaire). Sur le benchmark LFW, il atteint 99,77% de précision de vérification.

5.4 MediaPipe FaceMesh — Landmarks 3D

MediaPipe FaceMesh (Kartynnik et al., 2019) est un réseau léger (~1 Mo) basé sur MobileNetV2 qui prédit la position 3D de 478 landmarks en un seul passage. Contrairement aux approches en deux étapes (détection puis alignement), il opère directement sur la région d'intérêt et tourne en temps réel sur CPU à plus de 30 fps.

L'option `refine_landmarks=True` active un second réseau pour affiner la position des 10 points d'iris, utilisés dans le module de regard (section 4.4).

5.5 Rôle de Google Colab dans l'entraînement

5.5.1 Nécessité du GPU

L'entraînement d'un réseau U-Net sur les données de segmentation sémantique (WP3) requiert :

- des calculs de convolution sur des batches d'images haute résolution (512×512 px);
- une mémoire GPU suffisante pour un batch de 8 à 16 images;
- plusieurs heures à plusieurs dizaines d'heures d'entraînement selon la taille du dataset.

Ces contraintes dépassent les capacités d'un CPU standard. L'accès à un GPU NVIDIA de la série A100 ou T4 via Google Colab Pro a permis de mener ces entraînements dans des délais raisonnables (2 à 4 h par expérimentation).

5.5.2 Workflow Colab

Le workflow d'entraînement sur Colab suit les étapes suivantes :

1. Montage de Google Drive pour l'accès aux données et la sauvegarde des checkpoints;
2. Installation des dépendances spécifiques (`pip install ultralytics insightface onnxruntime`);
3. Chargement du dataset depuis Drive (format Hugging Face Datasets ou dossier structuré YOLOv8);
4. Lancement de l'entraînement avec `model.train(..., device=0)` (GPU);
5. Export du meilleur checkpoint au format [ONNX](#) pour l'inférence CPU locale;
6. Téléversement automatique des résultats sur GitHub via `git push` authentifié.

5.5.3 Export ONNX et optimisation CPU

Pour l'inférence locale sur CPU Windows, les modèles sont exportés au format [ONNX](#) (Open Neural Network Exchange) et exécutés via **ONNX Runtime**. Cette chaîne optimise automatiquement

le graphe de calcul (fusion d'opérateurs, quantification INT8 optionnelle) et réduit la latence d'inférence de 20 à 40% par rapport à PyTorch natif en mode CPU.

La variable d'environnement `ORT_LOGGING_LEVEL=3` (ERROR uniquement) est définie au démarrage de `main.py` pour éliminer les messages verbeux d'ONNX Runtime qui n'apportent pas d'information utile à l'opérateur.

5.6 Utilisation d'assistants IA pour la conception

Dans le cadre de ce projet, des assistants d'intelligence artificielle générative (modèles de langage) ont été ponctuellement utilisés pour :

- la génération de squelettes de code boilerplate (parsers d'arguments, boucles OpenCV) ;
- la reformulation de messages d'erreur complexes (exceptions ONNX Runtime, conflits de versions Anaconda) ;
- la recherche d'implémentations de référence d'algorithmes connus (distance de Mahalanobis, EAR).

Toute contribution issue d'un assistant IA a systématiquement été relue, comprise, adaptée au contexte du projet et testée avant intégration.

6. Conclusion

6.1 Bilan des objectifs atteints

Ce projet avait pour ambition de concevoir un système temps réel d'évaluation du degré de coopération d'un individu vis-à-vis d'un dispositif de reconnaissance faciale. Les six tâches définies dans le sujet ont été traitées :

1. **État de l'art** : revue des approches récentes de détection faciale, d'estimation de pose et de suivi multi-objets, ayant conduit à la sélection de YOLOv8, InsightFace et MediaPipe FaceMesh ;
2. **Dataset et annotation** : constitution d'un jeu de données vidéo intégrant des contextes variés (occlusions, profils, courses, démarches atypiques) ;
3. **Architecture** : conception et implémentation d'un pipeline modulaire en quatre blocs fonctionnels, configurable via un fichier central ;
4. **Modèles de classification** : développement du scoring de coopération multi-critères (yaw/pitch, EAR, occlusion, vitesse) ;
5. **Optimisation temps réel** : inférence CPU à ≥ 10 fps grâce à l'export ONNX et au seuillage précoce des détections non pertinentes ;
6. **Rapport** : rédaction du présent document présentant l'architecture, la méthodologie et les résultats.

Le système développé démontre la faisabilité d'une évaluation comportementale continue à faible latence, sans nécessiter de matériel spécialisé.

6.2 Limites identifiées

Plusieurs limites ont été constatées en cours d'expérimentation :

- **Latence CPU** : en dessous de 10 fps pour 4 personnes ou plus, le traitement en temps réel n'est plus garanti sans GPU dédié ;
- **Calibration peau** : le module d'occlusion est sensible aux fortes variations d'éclairage (contre-jour, néons clignotants) qui peuvent dégrader la précision de la segmentation colorimétrique ;

- **Profils extrêmes** : un profil à $>80^\circ$ de yaw échappe à la fois à la validation faciale et à l'estimation de pose, laissant le tracker en attente ;
- **Généralisation** : les seuils de coopération (yaw/pitch, EAR) ont été calibrés sur un ensemble restreint de sujets ; une phase de validation sur un panel plus large serait nécessaire.

6.3 Perspectives d'amélioration

Les pistes d'amélioration les plus prometteuses identifiées sont les suivantes :

Court terme Intégration d'un modèle d'estimation de pose de tête dédié (6DRepNet ou HopeNet) en remplacement du solveur PnP, pour une précision angulaire améliorée aux grands angles ; parallélisation des blocs indépendants (threading Python) pour augmenter le débit .

Moyen terme Extension au suivi multi-caméras avec un algorithme de fusion cross-caméras ; remplacement de la segmentation colorimétrique par un modèle neuronal léger de détection d'occlusion (réseau de classification par zone) .

Long terme Apprentissage d'un score de coopération unifié par réseau de neurones récurrent (LSTM/Transformer) intégrant l'ensemble des signaux comportementaux dans une représentation temporelle continue ; évaluation sur des datasets publics annotés de coopération (en cours de construction dans la communauté).

En définitive, ce projet a permis d'acquérir une maîtrise approfondie de la chaîne complète de traitement vidéo par IA, de la détection bas niveau à la décision comportementale de haut niveau, tout en intégrant les contraintes industrielles de performance temps réel sur matériel standard.

A. Installation et configuration de l'environnement

Prérequis système

- Windows 10/11 64 bits
- Anaconda Distribution (Python 3.10+)
- Visual Studio Code avec extension Python
- (Optionnel) Pilote NVIDIA CUDA 11.8+ pour accélération GPU

Création de l'environnement Anaconda

```
1 # Creer l'environnement
2 conda create -n occuface python=3.10 -y
3 conda activate occuface
4
5 # Dependances principales
6 pip install ultralytics==8.2.0
7 pip install insightface==0.7.3
8 pip install mediapipe==0.10.14
9 pip install opencv-python==4.9.0
10 pip install onnxruntime==1.17.3
11 pip install numpy scipy tqdm
12
13 # Verifier l'installation
14 python -c "from ultralytics import YOLO; print('YOLO OK')"
15 python -c "import insightface; print('InsightFace OK')"
```

Listing A.1 – Création de l'environnement conda

Exécution du pipeline principal

```
1 python main.py \
2     --video chemin/vers/video.mp4 \
3     --output resultats/video_annotee.mp4 \
4     --save-gallery resultats/galerie.pkl \
```

```
5  --verbose
```

Listing A.2 – Exemple d'utilisation de main.py

B. Configuration complète du système

Le fichier `config.py` centralise l'ensemble des hyperparamètres du pipeline. Les valeurs par défaut ci-dessous ont été optimisées pour un contexte de 2 à 5 personnes simultanées en résolution Full HD :

```
1 # Seuils de detection
2 conf_threshold: float = 0.30      # YOLO: bas pour capturer profils
3 iou_threshold: float = 0.50
4
5 # Qualite visage
6 min_face_quality_composite: float = 0.40 # permissif
7 qw_sharp: float = 0.30 # anti-flou de mouvement prioritaire
8
9 # ReID      peu de personnes
10 reid_threshold_composite: float = 0.55
11 weight_face: float = 0.45
12 weight_body: float = 0.55 # vêtements discriminants
13
14 # Galerie
15 gallery_max_face_per_id: int = 80
16 n_prototypes_max: int = 8
17
18 # State machine
19 pending_frames: int = 8 # 30fps x 8 = 0.27s latence mini
20 write_delay_frames: int = 6
```

Listing B.1 – Paramètres clés de `config.py`

Bibliographie - Projet d'estimation de coopération

Ewan Ménahéze

Bibliographie / Webographie

- [1] REDMON, J., DIVVALA, S., GIRSHICK, R. et FARHADI, A. You Only Look Once : Unified, Real-Time Object Detection. In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. p. 779–788. DOI : [10.1109/CVPR.2016.91](https://doi.org/10.1109/CVPR.2016.91).
- [2] ULTRALYTICS. *YOLOv8 : A New State-of-the-Art Vision Model*. 2023. [En ligne]. Disponible sur : <https://github.com/ultralytics/ultralytics> (Consulté le 15 mars 2025).
- [3] AHARON, N., ORFAIG, R. et BOBROVSKY, B.-Z. BoT-SORT : Robust Associations Multi-Pedestrian Tracking. In : *arXiv preprint*. 2022. [En ligne]. Disponible sur : <https://arxiv.org/abs/2206.14651>.
- [4] DENG, J., GUO, J., XUE, N. et ZAFEIRIOU, S. ArcFace : Additive Angular Margin Loss for Deep Face Recognition. In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019. p. 4690–4699. DOI : [10.1109/CVPR.2019.00482](https://doi.org/10.1109/CVPR.2019.00482).
- [5] INSIGHTFACE CONTRIBUTORS. *InsightFace : An Open Source 2D and 3D Deep Face Analysis Library*. 2021. [En ligne]. Disponible sur : <https://github.com/deepinsight/insightface> (Consulté le 10 mars 2025).
- [6] KARTYNNIK, Y., ABLAVATSKI, A., GRISHCHENKO, I. et GRUNDMANN, M. Real-Time Facial Surface Geometry from Monocular Video on Mobile GPUs. In : *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2019. [En ligne]. Disponible sur : <https://arxiv.org/abs/1907.06724>.
- [7] LUGARESI, C., TANG, J., NASH, H., et al. MediaPipe : A Framework for Building Perception Pipelines. In : *arXiv preprint*. 2019. [En ligne]. Disponible sur : <https://arxiv.org/abs/1906.08172>.
- [8] LEPETIT, V., MORENO-NOGUER, F. et FUA, P. EPnP : An Accurate O(n) Solution to the PnP Problem. In : *International Journal of Computer Vision*. 2009. vol. 81, no. 2, p. 155–166. DOI : [10.1007/s11263-008-0152-6](https://doi.org/10.1007/s11263-008-0152-6).
- [9] HEMPEL, T., ABDELRAHMAN, A. A. et AL-HAMADI, A. 6D Rotation Representation for Unconstrained Head Pose Estimation. In : *Proceedings of the IEEE International Conference on Image Processing (ICIP)*. 2022. DOI : [10.1109/ICIP46576.2022.9897219](https://doi.org/10.1109/ICIP46576.2022.9897219).
- [10] CAO, Z., SIMON, T., WEI, S.-E. et SHEIKH, Y. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In : *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017. p. 7291–7299. DOI : [10.1109/CVPR.2017.143](https://doi.org/10.1109/CVPR.2017.143).
- [11] CHEN, L.-C., ZHU, Y., PAPANDREOU, G., SCHROFF, F. et ADAM, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In : *Proceedings of*

- the European Conference on Computer Vision (ECCV)*. 2018. p. 801–818. DOI : [10.1007/978-3-030-01234-2_49](https://doi.org/10.1007/978-3-030-01234-2_49).
- [12] SUN, Y., ZHENG, L., YANG, Y., TIAN, Q. et WANG, S. Beyond Part Models : Person Retrieval with Refined Part Pooling (and A Strong Convolutional Baseline). In : *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. DOI : [10.1007/978-3-030-01225-0_30](https://doi.org/10.1007/978-3-030-01225-0_30).
- [13] MICROSOFT CORPORATION. *ONNX Runtime : Cross-Platform, High-Performance ML Inferencing*. 2019. [En ligne]. Disponible sur : <https://onnxruntime.ai> (Consulté le 1er avril 2025).
- [14] HUANG, G. B., RAMESH, M., BERG, T. et LEARNED-MILLER, E. Labeled Faces in the Wild : A Database for Studying Face Recognition in Unconstrained Environments. In : *Workshop on Faces in 'Real-Life' Images : Detection, Alignment, and Recognition*. Marseille, France. 2008.
- [15] LIN, T.-Y., MAIRE, M., BELONGIE, S., et al. Microsoft COCO : Common Objects in Context. In : *Proceedings of the European Conference on Computer Vision (ECCV)*. 2014. p. 740–755. DOI : [10.1007/978-3-319-10602-1_48](https://doi.org/10.1007/978-3-319-10602-1_48).