Rapport Projet

Vision artificiel et traitement des images

Damien Santerne

Objectifs **2**
Etat de l'art **2**
Création de la base de données **2**
Détection Yolo **2**
Algorithme global **2**
Autre idée d'utilisation **3**
Limitations et remarques **3**
Faux positifs 3
Temps d'exécution 4
Contrainte réelle 4

Objectifs

L'implémentation optimal que j'aimerai faire est un algorithme pour des feux piétons en fonction des personnes qui attendent pour passer.

Etat de l'art

Il existe actuellement des passages piétons fonctionnant à une fréquence fixe et des passages piétons nécessitant une interaction pour demander de passer.
L'implémentation que je présente permettrait de surpasser des limites culturelles (certaines personnes ne connaissent pas les passages piétons à interactions), permettrait d'adapter les temps en fonction du nombre de personnes en attente.
Le tout sans le moindre effort du piéton. On notera aussi que cela aidera encore plus les personnes en situations de difficultés motrices ou souffrantes de déficiences visuelles.

Création de la base de données

Ce que j'imaginais pour une implémentation sur les passages piétons était d'avoir une caméra au-dessus des feux piétons et qui est orientée vers le feu piéton en face.
Sauf que je n'ai pas réussi à trouver de bases de données ou même suffisamment de vidéos qui présentait ces caractéristiques. J'ai donc construit une base de données qui pourrait ressembler le plus possible à ça.
Ma base contient uniquement des vidéos de tests (car j'utilise le modèle Yolo de manière brute). J'ai constitué cette base de tests avec des vidéos durant approximativement entre 10 et 30 secondes provenant principalement du site PEXELS et du PIE dataset.

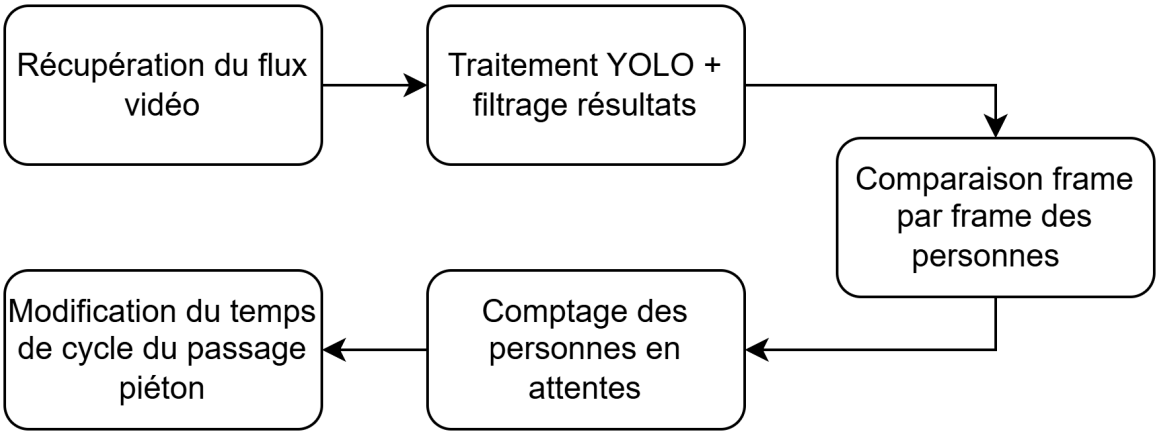
ÉTS
EXCEPTÉ
LIVRAISON
LOCALE

Détection Yolo

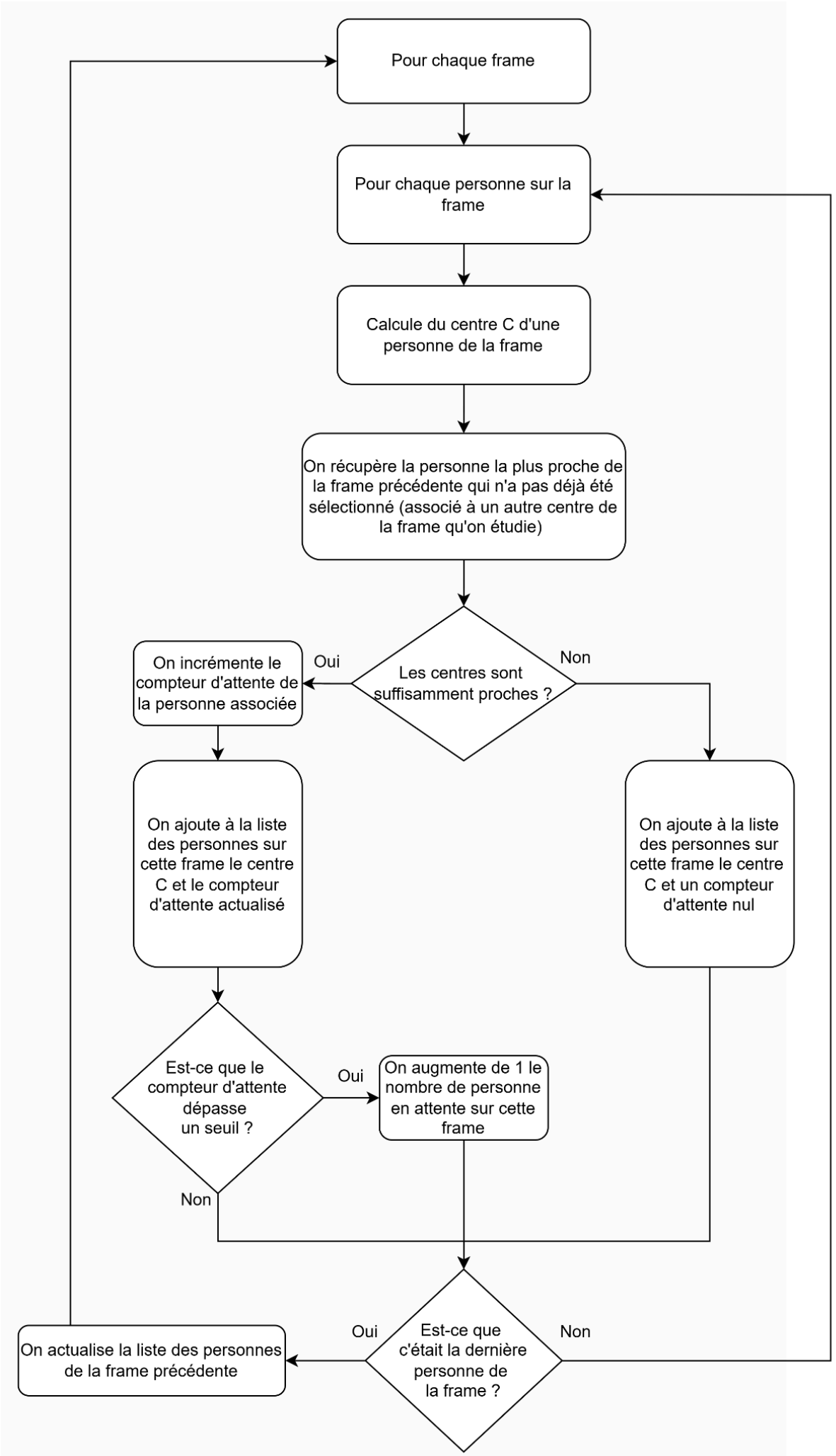
J’utilise le modèle Yolo v8 pour sa rapidité. Puis je trie les résultats pour simplement extraire les cadres des personnes détectés.

Dans mon algo je fournis directement le lien vers la vidéo mais dans le cadre d’une implémentation avec un flux continue je peux aussi utiliser un flux capture d’open cv par exemple et utiliser le modèle Yolo frame par frame.

Algorithme global



1. **Initialisation**:
    - **Paramètres** : Définition des seuils et des paramètres nécessaires pour l'algorithme, tels que le seuil de distance pour considérer une personne immobile (seuil_distance) ou encore le nombre de frames successives pour qu'une personne soit considérée en attente (seuil_attente).
    - **Dossier de Résultats** : Création d'un dossier pour stocker les résultats (avec l'heure dans le nom du fichier, cf tuto4)
2. **Détection des Personnes** :
    - **Chargement de la Vidéo** : Ouverture de la vidéo et récupération de la taille des frames.
    - **Modèle YOLO** : Utilisation de YOLO v8n pour détecter les personnes dans chaque frame de la vidéo.
    - **Filtrage des Résultats** : Extraction des bounding boxes des personnes détectées qui sont stockées dans une variable globale.
3. **Suivi des Personnes et Comptage des Personnes en Attente**:
    - Voir schéma page suivante :


Pour chaque frame
Pour chaque personne sur la frame
Calcule du centre C d'une personne de la frame
On récupère la personne la plus proche de la frame précédente qui n'a pas déjà été sélectionné (associé à un autre centre de la frame qu'on étudie)
Les centres sont suffisamment proches ?
Oui
Non
On incrémente le compteur d'attente de la personne associée
On ajoute à la liste des personnes sur cette frame le centre C et le compteur d'attente actualisé
On ajoute à la liste des personnes sur cette frame le centre C et un compteur d'attente nul
Est-ce que le compteur d'attente dépasse un seuil ?
Oui
Non
On augmente de 1 le nombre de personne en attente sur cette frame
Est-ce que c'était la dernière personne de la frame ?
Oui
Non
On actualise la liste des personnes de la frame précédente

4. :**Sortie :**
    - **Liste du nombre de personnes en attente par frame** : On renvoie la liste des personnes en attente à chaque frame de la vidéo. Dans un cas appliqué ce serait simplement un entier et non une liste.
5. **Sauvegarde et Visualisation (Non essentiel)** :
    - **Sauvegarde des Résultats** : Les résultats de la détection et du comptage sont sauvegardés dans des fichiers.
    - **Visualisation** : Génération d'images annotées montrant les personnes détectées et le nombre de personnes en attente pour chaque frame, sauvegardées dans le dossier de résultats

Suite

Évidemment il faut maintenant exploiter ces données. Pour ceci j'ai imaginé une formule décrivant le temps restant avant le changement de feu piéton en fonction du nombre de personnes en attentes:

$$t_{restant}(t, n) = t_{cycle} - t - (n * \Delta t)$$

Avec n = le nombre maximum de personnes détectées en attente pendant le cycle.

Autre idée d'utilisation

En utilisant ce système il serait aussi possible, au lieu de l'exploitation d'un flux vidéo direct / continu, d'utiliser uniquement les vidéos des caméras et cet algorithme pour déterminer une loi gaussienne. Cette probabilité permettra d'adapter les fréquences et temps de passage des piétons en fonction de l'heure et des jours.

Limitations et remarques

Faux positifs

Dans mon algorithme il est difficile de déterminer quelles personnes, qui sont en attente, le sont pour le passage piéton.

Par exemple ici mon algo renvoie une personne en attente alors que la personne détectée ne souhaite pas traverser. Et cette limite sera sûrement présente même avec des prises de vue idéales. Par contre un axe d'amélioration pourrait être de détecter le passage piéton et d'en écarter les gens n'étant pas dans sa continuité,cependant je ne développerait pas cette amélioration maintenant.
Il y a aussi un problème qui apparaît dans mes exécutions et qui ressemble à ce dont je viens de parler, Yolo détecte toutes les personnes présentes sur l'image, même celles assez éloignées. J'ai donc eu comme idée de rajouter une contrainte sur la taille des bounding boxes mais ma base de test étant trop peu uniformisée il n'existe pas de seuil satisfaisant toutes les vidéos. Je choisis donc de laisser ce problème en l'état car il faudrait une une mesure sur une caméra normée pour cette application.

Seuil de mouvement

Dans ma base de vidéos, le framerate varie. Ce qui implique qu'entre 2 frames le déplacement d'une personne n'est pas le même et donc qu'avec une vidéo ayant un framerate trop élevé mon algorithme va détecter toutes les personnes présentent comme immobile. Ce qui ne serait pas un problème car sur une implémentation avec un flux continue le framerate sera constant et donc il faudra juste ajuster seuil_distance.

Temps d'exécution

Une autre limite que je peux noter est le temps d'exécution du programme :
J'ai mesuré dans un cas : 45 secondes pour traiter 240 frames et 4min40 pour 3200 frames (avec le modèle yolo le plus rapide que j'ai trouvé = yolov8n) soit une vitesse de traitement entre 5 et 12 frames par seconde. Même si cette valeur dépend du support, cet ordre de grandeur semble pouvoir permettre une application continue à condition de gérer la mémoire.

Contraintes réelles

Cet algorithme de décisions est limité à un seul passage piéton, il faudrait donc à minima le coordonnées avec les autres dans le cas de plusieurs passages (4 voies par exemple) et encore plus compliqué de le couplés avec d'autres feux routiers d'une même route sinon on risque de créer de gros ralentissement côté voiture.
On notera aussi que cette implémentation implique l'utilisation d'au moins 2 caméras par passages piétons. Ce qui peut vite être coûteux, on pourrait imaginer par contre une utilisation plus locale que globale, uniquement pour certains passages piétons peu utilisés.