### **Rapport MLVOT**

**TP4**

## **1. calculate\_jaccard\_index(rect1, rect2)**

Cette fonction calcule l'indice de Jaccard entre deux rectangles donnés. L'indice de Jaccard mesure la similarité entre deux ensembles, définis ici par les zones des rectangles. Le résultat est une valeur entre 0 et 1, où 1 signifie une correspondance parfaite et 0 aucune correspondance. Cette mesure est utilisée pour comparer l'intersection et l'union des zones des rectangles, ce qui aide à évaluer la similarité entre les positions d'objets détectés dans des frames consécutives.

## **2. compute\_similarity(rect1, rect2)**

Cette fonction calcule l'indice de Jaccard entre deux rectangles et le renvoie uniquement s'il dépasse un seuil minimal (0.1). Sinon, elle renvoie 0. Elle permet de filtrer les correspondances peu probables, augmentant ainsi l'efficacité du processus de correspondance.

## **3. create\_similarity\_matrix(rectangles\_frame1, rectangles\_frame2)**

Cette fonction crée une matrice de similarité entre les rectangles de deux frames consécutives en utilisant l'indice de Jaccard.La matrice de similarité est essentielle pour l'appariement des objets entre les frames, facilitant le suivi d'objets.

## **4. match\_tracks(similarity\_matrix)**

Cette fonction utilise l'algorithme de la somme des affectations linéaires (linear\_sum\_assignment) pour déterminer les correspondances optimales entre les rectangles en fonction de la matrice de similarité. Elle permet d'assigner de manière optimale les objets détectés dans une frame aux objets suivis dans la précédente, minimisant ainsi le coût global.

## **5. calculate\_centroid(rect)**

Cette fonction calcule les coordonnées du centre d'un rectangle donné. Le calcul du centroïde est utilisé pour la mise à jour et la prédiction des positions des objets avec le filtre de Kalman.

## **6. predict\_next\_position(rect, kalman\_filters, track\_id)**

Cette fonction prédit la prochaine position d'un rectangle en utilisant un filtre de Kalman spécifique à la piste (track\_id). Le filtre de Kalman est utilisé pour prévoir la prochaine position probable d'un objet, améliorant ainsi la continuité et la précision du suivi.

## **7. update\_tracks\_and\_history(rectangles, previous\_positions, previous\_tracks, current\_tracks, track\_histories, kalman\_filters)**

Cette fonction met à jour les positions des pistes et l'historique des objets suivis en utilisant les nouvelles détections et les prédictions du filtre de Kalman. Elle assure que les positions des objets sont mises à jour correctement et maintient un historique des positions pour chaque piste.

## **8. get\_previous\_positions(track\_positions, track\_histories)**

Cette fonction récupère les positions précédentes des rectangles à partir de l'historique des pistes. Elle permet de comparer les nouvelles détections aux positions précédentes, facilitant la création de la matrice de similarité.

## **9. process\_frame(frame\_number, track\_positions, previous\_tracks, track\_histories, detections, kalman\_filters)**

Cette fonction traite une frame vidéo : elle lit l'image, crée la matrice de similarité, met à jour les pistes et affiche les résultats. Elle orchestre l'ensemble du processus de suivi pour chaque frame, assurant une mise à jour continue et cohérente des pistes.

## **10. save\_tracks\_to\_file(detections, track\_histories)**

Cette fonction enregistre les pistes dans un fichier suivant le format du fichier de vérité terrain. Elle permet de sauvegarder les résultats du suivi pour une analyse ultérieure ou pour une comparaison avec des méthodes de vérité terrain.

## **11. main()**

Cette fonction principale initialise le processus, charge les détections, traite chaque frame et enregistre les résultats.

Le résultat est affiché dans la vidéo jointe, et le process prend 0.039 secondes par frame

**TP5**

pour le TP5 j’ai essayé de rajouter OSnet de ce repo <https://github.com/KaiyangZhou/deep-person-reid>

mais je n'ai pas réussi à le faire tourner sur ma machine.