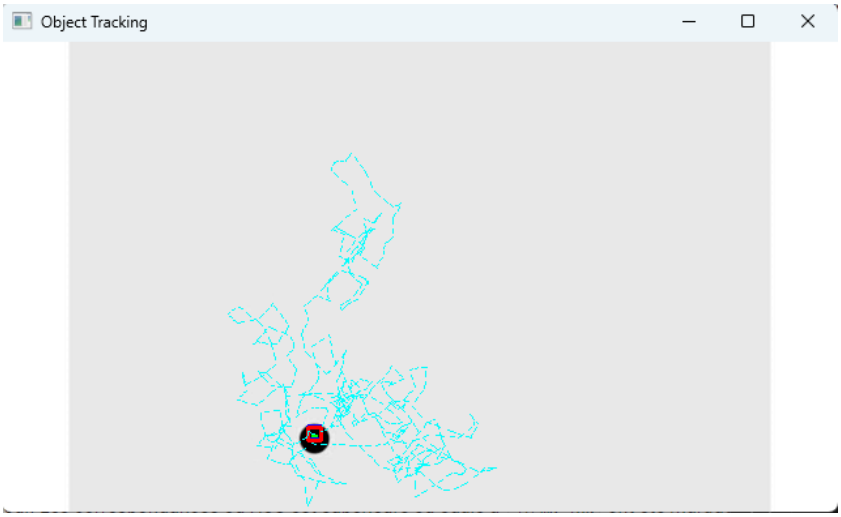
TP1

Objectif : L'objectif de ce travail pratique était de mettre en œuvre le suivi d'objets en 2D à l'aide d'un algorithme de détection d'objets préexistant et d'intégrer le filtre de Kalman pour un suivi fluide et précis. L'objet est représenté par un point (centroïde) et le suivi est effectué pour un seul objet.

Matériaux fournis : Un fichier vidéo contenant une séquence de trames avec un objet à suivre et un code de détection d'objets (Detector.py) qui utilise la détection de bord de Canny pour détecter plusieurs objets dans chaque trame et renvoie les centres des objets détectés.

Implémentation du filtre de Kalman : Le fichier KalmanFilter.py contient une classe appelée KalmanFilter qui comprend trois fonctions : __init__(), predict(), update(). Cette classe est initialisée avec six paramètres : dt, u_x, u_y, std_acc, x_sdt_meas, y_sdt_meas. Les variables d'entrée de contrôle sont définies comme u=[u_x, u_y] et la matrice d'état initiale est définie comme (𝑥⍥◌⍰)=[x0=0, y0=0, vx=0,vy=0]. Les matrices décrivant le modèle du système A,B sont définies par rapport au temps d'échantillonnage dt. La matrice de mappage de mesure H, la matrice de covariance du bruit de processus initial Q et la matrice de covariance du bruit de mesure initial R sont également définies.

Création du fichier principal : Le fichier objTracking.py est le fichier principal de ce projet qui sera exécuté pour suivre un objet. Il importe la fonction detect() et KalmanFilter. Un objet de la classe KalmanFilter est créé avec des valeurs de paramètres définies. Un objet de capture vidéo est créé. Le code de détection d'objets fourni est utilisé pour détecter un cercle noir dans chaque trame. Si un centroïde est détecté, il est suivi. La fonction de prédiction de Kalman et la fonction de mise à jour du filtre de Kalman sont appelées. Les résultats du suivi sont visualisés en dessinant un cercle détecté (couleur verte), un rectangle bleu comme position d'objet prédite, un rectangle rouge comme position d'objet estimée, et la trajectoire (chemin de suivi) dans une image.



Ce travail pratique a permis d'acquérir une compréhension approfondie de l'implémentation du suivi d'objets en utilisant le filtre de Kalman pour un suivi précis et fluide.

TP2

Objectif

L'objectif de ce travail pratique était de développer un tracker simple basé sur l'Intersection over Union (IoU) sans utiliser d'informations d'image. L'algorithme devait être étendu pour gérer le suivi de plusieurs objets simultanément. L'objet est représenté par une boîte englobante.

Données

Les détections pré-générées ont été chargées à partir d'un fichier texte. Les images sont en JPEG et nommées séquentiellement avec un nom de fichier à 6 chiffres (par exemple, 000001.jpg). Les fichiers de détection et d'annotation (dossiers det et gt) sont des fichiers CSV simples.

Méthodologie

Les détections ont été chargées à partir d'un fichier texte formaté comme le MOT Challenge. Chaque ligne représente une instance d'objet et contient 10 valeurs : <frame>, <id>, <bb_left>, <bb_top>, <bb_width>, <bb_height>, <conf>, <x>, <y>, <z>.

Un score de similarité a été calculé en utilisant l'indice de Jaccard (intersection sur union) pour chaque paire de boîtes englobantes. Une matrice de similarité a été créée pour stocker l'IoU pour toutes les boîtes.

Les détections ont été associées aux pistes de manière gloutonne en utilisant l'IoU/seuil sigma_iou. Une piste obtient la détection avec la plus grande intersection sur l'union par rapport à sa dernière position d'objet connue (c'est-à-dire la détection précédente de la piste) attribuée.

Chaque objet peut être assigné à une seule trajectoire (ID). Des listes pour les correspondances, les détections non appariées et les pistes non appariées ont été créées et mises à jour. Les correspondances où l'IoU est supérieure ou égale à sigma_iou ont été marquées comme piste correspondante. Les pistes non appariées ont été supprimées. De nouvelles pistes ont été créées pour les détections non appariées.

Une interface a été développée pour vérifier si le tracker suit correctement les objets en associant les bons ID dans le flux vidéo. Une boîte englobante rectangulaire a été dessinée autour de l'objet détecté dans les images. Un ID attribué a été dessiné pour chaque objet suivi. La trajectoire (chemin de suivi) a été dessinée dans une image.

Problèmes rencontrés

La question 5 n'a pas été réussie. Malgré la mise en place d'un système de gestion des pistes et l'association des détections aux pistes, les boîtes de détection n'apparaissent pas dans la vidéo output.avi. Plusieurs tentatives de débogage ont été effectuées, notamment la vérification des détections, la vérification du chemin des images, la vérification de la taille de la vidéo de sortie et la vérification de la gestion des pistes. Cependant, le problème persiste.

Conclusion

Ce travail pratique a permis de comprendre comment fonctionne un tracker basé sur l'IoU et comment il peut être utilisé pour le suivi de plusieurs objets. Bien que la question 5 n'ait pas été réussie, ce TP a fourni une expérience précieuse dans le développement d'un tracker et la résolution des problèmes associés. Des travaux futurs pourraient inclure la résolution du problème avec la question 5 et l'extension du tracker pour utiliser des informations d'image en plus de l'IoU.

TP3 et 4

Exercice 3

Objectif

L'objectif de cet exercice était d'étendre le tracker multi-objets basé sur l'Intersection over Union (IoU) en intégrant l'algorithme hongrois pour l'affectation des détections aux pistes.

Méthodologie

Intégration de l'algorithme hongrois : L'algorithme hongrois a été appliqué en utilisant la fonction linear_sum_assignment de la bibliothèque scipy en Python. Les valeurs précédemment calculées de la matrice de similarité (IoU) ont été utilisées en entrée pour trouver l'affectation optimale des détections aux pistes existantes.

Sauvegarde des résultats de suivi : Les résultats du suivi ont été sauvegardés dans un fichier txt. Le nom du fichier est exactement le même que le nom de la séquence. Le format du fichier est le même que celui du fichier ground truth (gt.txt), qui est un fichier texte CSV contenant une instance d'objet par ligne. Chaque ligne contient 10 valeurs. La colonne id (2ème valeur) a été mise à jour avec l'ID unique attribué à la piste. La 7ème valeur (conf) agit comme un drapeau 1.

Exercice 4

Objectif

L'objectif de cet exercice était d'étendre le tracker multi-objets basé sur l'IoU avec l'algorithme hongrois en ajoutant un filtre de Kalman.

Méthodologie

Représentation des boîtes englobantes par leurs centroïdes : Les boîtes englobantes ont été représentées par leurs centroïdes pour pouvoir appliquer le filtre de Kalman développé dans l'exercice 1.

Modification de l'algorithme de suivi : L'algorithme de suivi a été modifié selon le diagramme fourni. Le filtre de Kalman a été utilisé pour prédire l'état actuel de chaque piste lors du calcul de la matrice de coût. Ensuite, pour chaque piste correspondante, la méthode update du filtre de Kalman a été utilisée pour mettre à jour l'estimation de l'état avec la nouvelle mesure. Une liste de filtres de Kalman correspondant à chaque piste a été maintenue.

Conclusion

Ces exercices ont permis de comprendre comment fonctionne un tracker multi-objets basé sur l'IoU et comment il peut être étendu pour intégrer l'algorithme hongrois et le filtre de Kalman. Les résultats du suivi ont été sauvegardés dans un fichier txt pour une analyse ultérieure. Les exercices ont fourni une expérience précieuse dans le développement d'un tracker et la résolution des problèmes associés. Des travaux futurs pourraient inclure l'extension du tracker pour utiliser des informations d'image en plus de l'IoU, de l'algorithme hongrois et du filtre de Kalman.