Rapport des travaux pratiques

Auteur: Virgile Hermant

TP 1: Single Object Tracking with Kalman (Centroid-Tracker)

Objectif

Construire un traqueur d'objet en utilisant des prédictions déjà faites, puis rajouter un filtre de Kalman pour lisser le résultat et affiner les prédictions.

Réalisations

Le filtre de Kalman réalisé réussi bien à rendre fluide la vidéo, en plus de permettre la création d'une vidéo contenant le chemin parcouru par la balle durant l'animation.

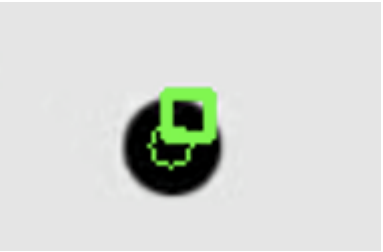
Ce filtre ce trouve dans une classe qui lui est propre `KalmanFilter` afin de le réutiliser plus tard dans les autres parties du TP

Résultats

La vidéo résultante de cette première partie se nomme `tp1.mp4` et `tp1_path.mp4`.
On remarque que le rectangle rouge (donc la position estimée) est plus fluide comparé à la position prédite (en bleue), le filtre de Kalman a donc bien fonctionné.

Problème rencontré

A la fin du premier TP, le filtre n'arrivait pas à prédire correctement la position de la balle. En comparaison avec la véritable position, le filtre donnait un rectangle décalé (voir image en-dessous). Le problème venait du calcule retour entre centroïdes et position des coins du rectangle vert.



TP 2: IOU-based Tracking (Bounding-Box Tracker)

Objectif

Développer un système de suivi simple basé sur l'IoU et l'étendre au suivi d'objets multiples.

Réalisation

Le principal de cette partie se trouve dans une fonction `track_management` qui gère le suivi d'objets tout au long de la vidéo. Elle associe des détections entre frames en utilisant la similarité (IoU).

Voici les étapes principales :

① **Initialisation** : Crée des variables pour les pistes actives, inactives, et les objets suivis.
② **Association** : Associe les détections actuelles aux pistes actives via une matrice de similarité (IoU > 0.3). Si une détection est non associées, la fonction vérifie les pistes inactives pour essayer de l'associer.
③ **Mise à jour** : Réactive les pistes inactives correspondant à des détections ou les supprime si elles dépassent un seuil d'inactivité (ex. 5 frames).
④ **Création** : Crée de nouvelles pistes pour les détections non associées.
⑤ **Résultat** : Retourne un DataFrame indiquant les objets suivis par frame (ID et coordonnées).

Résultats

La vidéo résultante de cette première partie se nomme `tp2.mp4`. On remarque que les IDs sont bien conservés d'une frame à l'autre, mais les rectangles dessinés tremblent et les identifiants sont souvent perdus puis réassociés à cause d'occlusions.

Problème rencontré

Problème: j'ai rencontré un problème lors de mon association, mon programme n'arrivait pas à associer les détections d'une frame à l'autre
Résolution: Oublie d'un `-` lorsque je donnais la matrice à `linear_sum_assignment`

Problème: lors de la création de la vidéo, les identifiants étaient tous `-1` (voir image ci-dessous)
Résolution: mauvais type qui empêchait le programme d'associer correctement les ids et mauvaise compréhension de ma part de l'algorithme permettant d'associer les identifiants entre les frames d'avant et la frame actuelle

TP 3: Kalman-Guided IoU Tracking (Bounding-Box Tracker)

Objectif

Extension du traqueur d'objets multiples basé sur l'IoU avec l'algorithme hongrois par l'ajout d'un filtre de Kalman.

Réalisation

Le principal enjeu de cette partie était de mettre ensemble la partie une et deux du projet.
Pour cela, j'ai mis à jour la fonction `track_management` pour utiliser le filtre de Kalman pour pour prédire et mettre à jour les positions des objets détectés sur plusieurs frames. Voici les étapes principales :

① **Initialisation** : Les pistes actives, inactives, et un ID unique pour chaque piste sont définis.
② **Prédictions** : Les pistes actives prédisent leur position dans le frame suivant grâce au Kalman Filter.
③ **Association** : Les détections actuelles sont associées aux pistes actives/inactives via une matrice de similarité (IoU > 0.3). Les pistes mises à jour reçoivent leurs nouvelles positions et sont marquées comme actives.
④ **Création de nouvelles pistes** : Les détections non associées génèrent de nouvelles pistes si elles ne sont pas des doublons.
⑤ **Mise à jour des pistes inactives** : Les pistes inactives restent valides pendant un certain nombre de frames (ex. 5) avant d'être supprimées.

⑥ **Résultat** : Retourne un DataFrame contenant les coordonnées des objets suivis avec leur ID par frame.

Résultats

La vidéo résultante de cette première partie se nomme `tp3.mp4`. On remarque que bien qu'un peu plus fluide, il n'y a pas de grand changement entre cette partie et la partie précédente.

Problème rencontré

Problème: réussir à lier le filtre de Kalman de la première partie, la transition entre les centroïdes et les rectangles faisaient des problèmes dans ma fonction.
Résolution: plusieurs itérations

Problème: lors de la première itération, les identifiants n'étaient plus du tout associés d'une frame à l'autre, ce qui résultait à des ids immenses puisqu'un identifiant était crée par détection pour chaque frame
Résolution: reprise de l'algorithme, je faisais mes prédictions au mauvais moment

TP 4: Appearance-Aware IoU-Kalman Object Tracker

Objectif

Extension du tracker IoU-Kalman pour inclure la ré-identification des objets (ReID) en tenant compte de la similitude d'apparence entre l'état prédit de l'objet (de l'image t-1) et les objets détectés (de l'image t).

Réalisation

La fonction `track_management` effectue cette fois le suivi d'objets détectés dans les frames de la vidéo en intégrant des caractéristiques de réidentification. Voici un résumé des étapes principales :

① **Initialisation** : Les pistes actives, inactives, et un ID unique pour chaque piste de la première frame sont définis.
② **Extraction des caractéristiques** : Les caractéristiques ReID sont extraites pour chaque détection.
③ **Prédiction des positions** : Les positions des pistes actives sont prédites via un Kalman Filter.
④ **Association des détections** : Les détections sont associées aux pistes actives/inactives en combinant des métriques d'IoU et de similarité des caractéristiques.
⑤ **Mise à jour des pistes** : Les pistes actives sont mises à jour, les pistes inactives tentent d'être récupérées, et de nouvelles pistes sont créées pour les détections non associées.

⑥ **Gestion des pistes inactives** : Les pistes inactives sont conservées pendant un certain nombre de frames avant d'être supprimées.

⑦ **Résultat** : Retourne un DataFrame contenant les objets suivis avec leurs coordonnées et IDs par frame.

Résultats

La vidéo résultante de cette première partie se nomme `tp4.mp4`. On remarque cette fois que les identifiants sont mieux conservés d'une frame à l'autre (notamment visible sur la dame au premier plan avec son haut vert qui ne perd plus son identifiant en plein milieu).

Problème rencontré

Problème: utilisation du modèle ReId avec les poids fournis, les types entre les images et ce qu'attendais le modèle n'était pas compatible
Résolution: plusieurs itérations

Problème: Pareil que lors de la partie précédante, les identifiants n'étaient plus associés d'une frame à l'autre, en effet, seuls les identifiants de la premières frame étaient conservés et l'algorithme n'affectait plus de nouveaux identifiants par la suite.
Résolution: plusieurs itérations pour régler le problème.