Projet Visual Object Tracking

Novembre 2024 - Décembre 2024

Écrit par

Gaspard SALIOU

GitHub: https://github.com/Gaspard-s/mlvot

Table des matières

1 TP1 : Suivi d'objet avec Filtre de Kalman

1.1 Méthode

Le TP consistait à implémenter un suivi d'objet en 2D en intégrant un algorithme de détection existant avec un filtre de Kalman. Les étapes suivantes ont été réalisées :

1. Détection d'objets

La fonction `detect` a été utilisée pour détecter le centre des cercles noirs dans les images vidéo. Cette fonction :

— Convertit les images en niveaux de gris.

— Applique un seuillage binaire inversé pour isoler les objets.

— Identifie les contours pour localiser les centres des objets.

2. Filtre de Kalman

Un filtre de Kalman a été implémenté dans `KalmanFilter.py`. Ce filtre comprend :

— Une étape de prédiction de la position et de la covariance d'erreur.

— Une étape de mise à jour basée sur les mesures obtenues de la détection.

3. Suivi et Visualisation

Le fichier principal, `objTracking.py`, gère l'intégration des deux modules. Le suivi est visualisé en :

— Dessinant le cercle détecté en vert.

— Affichant une boîte bleue pour la position prédite et une rouge pour la position estimée.

— Traçant la trajectoire complète en jaune.

1.2 Résultats

Le programme a permis un suivi précis de l'objet dans la vidéo fournie. La trajectoire suivie était cohérente avec les mouvements de l'objet détecté. Les visualisations ont montré une estimation fluide et correcte grâce à l'intégration du filtre de Kalman.

1.3 Difficultés et Solutions

1. Problème de compatibilité avec OpenCV

Initialement, la fonction `detect` utilisait une méthode incompatible avec ma version de Python et OpenCV. J'ai dû ajuster le code pour corriger ces incompatibilités, notamment dans la récupération des contours et la conversion des couleurs.

2. Implémentation de la visualisation

La superposition des différents éléments de suivi (cercles, rectangles, trajectoire) a nécessité plusieurs itérations pour garantir une visualisation claire et cohérente. Des ajustements ont été faits sur l'ordre des dessins et les couleurs utilisées.

2 TP2 : IOU-based Tracking

2.1 Méthode

Ce TP avait pour objectif de développer un tracker simple basé sur le calcul de l'IoU (Intersection over Union) et d'étendre ce système pour le suivi d'objets multiples (MOT). Les étapes réalisées sont décrites ci-dessous :

1. Chargement des détections

Les détections ont été importées à l'aide de la fonction `load_detections` qui les met dans un dataframe.

2. Calcul de la matrice de similarité

Une matrice de similarité a été créée pour associer les objets suivis aux nouvelles détections en utilisant l'IoU. Chaque entrée de la matrice représente le score IoU calculé entre une boîte suivie et une nouvelle détection.

3. Association des détections aux trajectoires

Pour trouver l'association optimale entre les trajectoires existantes et les nouvelles détections, l'algorithme de Hongrie a été appliqué (fonction `linear_sum_assignment` de la bibliothèque SciPy).

4. Gestion des trajectoires

— Les trajectoires associées ont été mises à jour avec les nouvelles détections correspondantes.

— Les trajectoires non associées ont été marquées comme perdues et supprimées après un nombre maximum de frames sans association.

— De nouvelles trajectoires ont été créées pour les détections non associées.

5. Visualisation et sauvegarde des résultats

Un système de visualisation a été fait pour afficher les boîtes englobantes et les IDs des objets suivis sur les frames vidéo. Vous pouvez trouver le résultat dans le dossier videos sur GitHub.

2.2 Résultats

Le système développé a permis un suivi efficace des objets dans la séquence vidéo. Les IDs des objets ont été correctement assignés et maintenus à travers les frames ; cependant, on peut voir qu'à certains moments, nous avons des objets qui apparaissent durant quelques frames qui ne reflètent rien dans la vidéo.

2.3 Difficultés et Solutions

1. Format des données

La compréhension du fichier det.txt a été un peu compliquée pour comprendre à quoi correspond quoi. Cette difficulté a été résolue en définissant clairement les colonnes pertinentes lors du chargement des données et en utilisant un dataframe pour les stocker.

3. Visualisation

L'implémentation de la visualisation a posé des défis pour synchroniser les données de suivi avec les frames vidéo. Ça a été résolu en structurant les données par frame et en vérifiant leur correspondance avec les fichiers image.

3 TP3 : Kalman IoU Tracking

3.1 Méthode

Ce TP visait à intégrer un filtre de Kalman au système de suivi par IoU développé dans le TP2. L'objectif était d'améliorer la précision et la stabilité du suivi grâce à une prédiction basée sur le modèle Kalman. Les étapes réalisées sont :

1. Intégration du Filtre de Kalman

Chaque trajectoire suivie a été associée à une instance du filtre de Kalman implémenté dans le tp1. Les prédictions du filtre sont utilisées pour guider l'association entre les trajectoires existantes et les nouvelles détections.

2. Association des Détéctions aux Trajectoires

Comme dans le TP2, une matrice de similarité basée sur l'IoU a été calculée. Cependant, avant de calculer l'IoU, les positions prédites par le filtre de Kalman ont été utilisées pour aligner les trajectoires avec les nouvelles détections.

3. Mise à jour des Trajectoires

Après l'association, les trajectoires ont été mises à jour de deux façons :

— En actualisant les états du filtre de Kalman avec les nouvelles détections.

— En ajustant les boîtes englobantes pour refléter les positions estimées.

4. Gestion des Trajectoires Non Associées

Les trajectoires non associées pendant plusieurs frames consécutives ont été supprimées, tandis que de nouvelles trajectoires ont été créées pour les détections non appariées.

3.2 Résultats

L'intégration du filtre de Kalman a permis une amélioration marginale des résultats par rapport au TP2. Les trajectoires étaient légèrement plus fluides, et les IDs des objets suivis restaient cohérents. Cependant, dans des cas de chevauchements significatifs ou de mouvements brusques, les bénéfices apportés par le filtre de Kalman étaient limités.

3.3 Difficultés et Solutions

1. Intégration du filtre de Kalman

L'intégration du filtre a rendu le code plus complexe, augmentant le risque d'erreurs lors du débogage et de l'implémentation.

2. Visualisation et Débogage

La visualisation des résultats a nécessité des ajustements pour refléter à la fois les prédictions du filtre de Kalman et les observations des détections.

4 TP4 : Reid Tracker

4.1 Méthode

Le TP4 avait pour objectif d'étendre le suivi par IoU-Kalman en incluant une composante de re-identification (ReID) basée sur les caractéristiques visuelles des objets détectés. Voici les étapes principales réalisées :

1. Extraction des caractéristiques d'apparence

Les caractéristiques visuelles ont été extraites à l'aide d'un modèle de deep learning léger pré-entraîné, `OSNet`, optimisé pour la ré-identification d'objets. Voici les étapes :

— **Prétraitement des patchs** : Chaque boîte englobante a été convertie en un patch d'image, redimensionnée à 64×128, normalisée, et convertie en format RGB pour correspondre aux exigences du modèle `OSNet`.

— **Calcul des vecteurs de ReID** : Les patchs d'image ont été passés dans le modèle `OSNet` pour générer des vecteurs de caractéristiques, représentant l'apparence de chaque objet détecté.

2. Combinaison de l'IoU et des similarités d'apparence

Un score combiné a été calculé pour chaque paire `(trajectoire existante, nouvelle détection)` en utilisant la formule suivante :

$$S = \alpha \cdot IoU + \beta \cdot SimilitudeNormalisée$$

— **IoU** : Intersection over Union des boîtes englobantes.

— **Similitude Normalisée** : Basée sur la similarité cosinus entre les vecteurs de caractéristiques extraits.

— Les poids α et β ont été ajustés à 0.4 et 0.6, respectivement, pour donner une importance plus grande aux caractéristiques visuelles.

3. Gestion des trajectoires

Les associations ont été réalisées en utilisant l'algorithme de Hongrie sur la matrice de coût combinée. Les trajectoires non associées ont été gérées comme suit :

— Les trajectoires non appariées pendant un certain nombre de frames (`MAX_LOST_FRAMES`) ont été supprimées.

— De nouvelles trajectoires ont été créées pour les détections sans correspondance.

— Les détections ré-identifiées ont été associées aux trajectoires précédemment perdues en utilisant les vecteurs de ReID.

4.2 Résultats

Le tracker basé sur ReID a montré une amélioration notable dans les cas où les objets détectés sortaient et réapparaissaient dans le champ de vision. La combinaison des scores IoU et ReID a permis une association plus robuste, notamment dans les scénarios de chevauchement important ou d'apparitions soudaines. Toutefois, ca ne marche pas dans tous les cas lorque l'objet disparait trop longmtemps du champ de vision.

4.3 Difficultés et Solutions

1. Ré-implémentation de la fonction de suivi

La fonction `track` a dû être ré-implémentée à partir de zéro, car l'approche des TP précédents ne permettait pas d'intégrer efficacement les prédictions avec les vecteurs de ReID. la fonction a donc du etre reimplémentée pour gérer la combinaison des scores IoU et ReID.

3. Ajustement des poids α et β

L'équilibrage entre l'IoU et les similarités visuelles a nécessité plusieurs itérations pour trouver une configuration optimale.

5 TP5 : Detector Extension

5.1 Méthode

Le TP5 avait pour objectif d'intégrer un détecteur d'objets basé sur un modèle de deep learning léger afin d'améliorer l'efficacité du système de suivi multi-objets. Voici les étapes principales réalisées :

Intégration du Détecteur YOLOv8

Un modèle `YOLOv8` pré-entraîné a été utilisé pour détecter les objets dans les séquences vidéo. Les étapes suivantes ont été suivies :

- **Préparation des données** : Toutes les images de la séquence `ADL-Rundle-6` ont été triées et traitées par le modèle YOLOv8.
- **Filtrage des détections** : Seules les détections correspondant aux personnes (classe 0 dans le modèle COCO) avec une confiance supérieure à 0.5 ont été retenues.
- **Génération du fichier** `det.txt` : Les résultats des détections ont été enregistrés dans un fichier texte au format compatible avec le système de suivi.

5.2 Résultats

L'intégration de YOLOv8 a permis une détection plus rapide et précise des objets, ce qui a significativement réduit les erreurs d'association dans le système de suivi. Les résultats obtenus montrent une réduction du nombre d'`ID Switch` grâce à une précision accrue des détections initiales et le résultat final est disponible dans le dossier videos sur github.