POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

Travail pratique III

Détection et suivi d'un objet d'intérêt

Élèves :
Slimane Boussafeur
Noémie Kpatenon

Enseignants :
G.A. Bilodeau
Khalil Sabri

11 avril 2025

1 Présentation de la solution

Dans le cadre de ce TP, nous avons reçu la tâche d'identifier et de suivre plusieurs objets d'intérêt à travers une séquence vidéo. Pour ce faire, **nous avons pris la décision d'utiliser DeepSORT**, un modèle d'identification et de suivi.

DeepSORT est une évolution de l'algorithme SORT (Simple Online Realtime Tracking) auquel on ajoute de l'apprentissage profond pour réduire les changements au niveau des identités de trames en trames (*identity switch)*. Pour comprendre son fonctionnement, il est, en premier, important de se familiariser avec l'algorithme SORT directement [1].

SORT comprend 4 étapes principales :

- **Détection :** Un détecteur d'objets est utilisé pour détecter les objets qu'on veut suivre. Dans notre cas, nous utiliserons YOLOv8, car YOLO est très bien entraîné sur 80 classes comprenant la classe "tasse" [1][2].
- **Estimation :** Un filtre de Kalman est utilisé pour prédire l'état des objets. Cela inclut la position, la vélocité et l'accélération. L'état est prédit pour la trame courante en fonction de l'état précédent. [1].
- **Association :** Maintenant que nous avons les boîtes englobantes, le IoU est calculé entre chaque détection de nos objets d'intérêt et les boîtes englobantes prédites. Si le IoU calculé est en-dessous d'un certain seuil (IoUmin), alors l'association est rejetée. Cette tâche est optimisée en utilisant l'algorithme hongrois [1].
- **Gestion des ID :** Des ID sont crées ou supprimés en fonction du IoUmin. Si la distance calculée est inférieur au IoUmin, cela signal que l'objet est non suivi. Le suivi d'un certain objet est annulé s'il n'est plus détecté pendant un certain nombre de trames. Si celui-ci réapparaît plus tard, alors un nouvel ID lui sera assigné [1].

SORT est très performant, mais les problèmes de *identity switch* ainsi que sa vulnérabilité face aux occlusions persistent encore. Cela est principalement dû aux métriques calculés lors de l'étape d'association. Ainsi, **DeepSort introduit de meilleurs métriques** (*deep metrics*) **qui prennent en compte le mouvement (distance de Mahalanobis) et les descripteurs d'apparence utilisant la distance du cosinus** pour calculer la similarité entre les objets présent dans les boîtes englobantes (voir figure 1) [1].

Les avantages principaux de DeepSort sont sa **performance en temps réel**, sa **robustesse face à l'occlusion** grâce à l'usage de descripteurs d'apparence et sa **capacité à discriminer les objets similaires**, toujours grâce à ces mêmes descripteurs [3].

L'**inconvénient principal** de cette méthode est le fait qu'elle soit **dépendante d'un modèle de détection externe** (YOLO dans notre cas). Ainsi, DeepSort hérite de tous les problèmes du modèle de détection avec lequel il est jumelé. Ainsi, on doit s'assurer que le modèle de détection soit robuste [4].

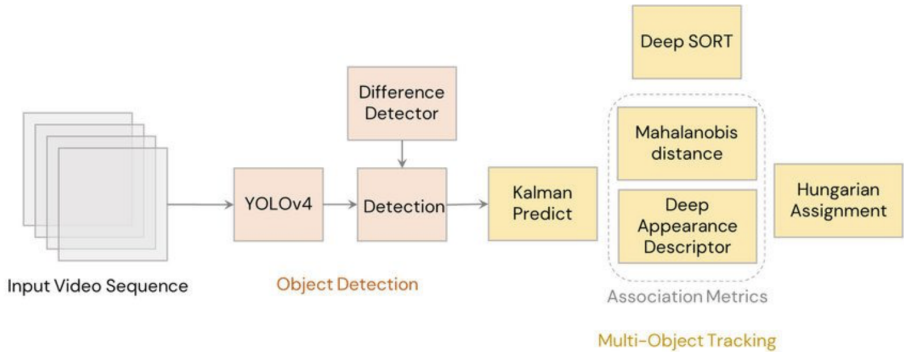

FIGURE 1 – Architecture DeepSort avec un détecteur YOLO [1]

2 Identification des difficultés dans la séquence sur Moodle

La séquence sur moodle en question présente plusieurs problèmes. Voici une liste de ceux qui ont été identifiés :

- **Présence d'occlusion** : les tasses se cachent entre elles à certains moments et sous certains angles. Cela pourrait impacter l'identification des objets occlus.
- **Disparition et réapparition** : certaines tasses sortent du champ et réapparaissent quelques trames plus tard. Cela pourrait poser un défi pour ce qui est de la réidentification avec le bon ID.
- **Niveau de zoom** : la caméra a tendance à se rapprocher et s'éloigner de certaines tasses, au point où on ne voit pas le manche. Cela pourrait faire en sorte qu'une tasse ne soit pas identifiée comme telle.
- **Flou de mouvement** : la caméra, comme mentionné, n'est pas fixe et elle bouge de gauche à droite. Cela pourrait créer du flou de mouvement et affecter le suivi.
- **Verres transparents** : vers la fin de la vidéo, nous voyons deux verres transparents à travers lesquels d'autres tasses sont visibles avec une légère déformation. Cela pourrait avoir un impact sur l'identification en associant un nouvel ID aux tasses visibles.
- **Similarité des tasses** : certaines tasses ont des apparences assez similaires, voir identiques. Ainsi, ça pourrait créer de la confusion au niveau de l'identification.

3 Justification de la méthode par rapport aux difficultés identifiées

La méthode DeepSORT a été choisie pour sa capacité à gérer les principales difficultés rencontrées dans les séquences vidéo du TP. Plusieurs facteurs justifient son utilisation :

- **Occlusions** : l'utilisation conjointe d'un filtre de Kalman et d'un descripteur d'apparence permet de prédire la trajectoire d'un objet et de le réidentifier même s'il disparaît temporairement [1].
- **Objets similaires** : grâce au descripteur d'apparence, DeepSORT peut distinguer deux objets proches en position et visuellement très semblables, comme deux tasses de même forme [1][8].
- **Changements d'échelle ou de position** : le filtre de Kalman permet d'anticiper les déplacements rapides ou les changements de taille dus à un zoom [1][3].
- **Détections bruitées** : le tracker est moins sensible à des erreurs ponctuelles de détection, grâce à l'association dynamique entre les prédictions et les détections [1][3].

Ainsi, DeepSORT répond de manière adaptée aux défis posés par la vidéo, en apportant plus de stabilité et de robustesse que des approches plus simples comme SORT.

4 Description de l'implémentation utilisée

Notre implémentation se divise en 2 sections :

- le code pour le *tracking* des tasses
- le code pour le *tracking* de personnes sur les vidéo de MOT17

Dans les deux cas, nous avons décidé de faire le *tracking* en utilisant **DeepSort avec YOLOv8** pour la détection d'objets d'intérêt. Les librairies communes aux deux implémentations sont **Ultralytics** pour l'importation du modèle YOLO, **Numpy** pour la manipulation de données, **OpenCV** pour la manipulation d'images et **os** pour nous permettre d'intéragir avec diverse fichiers. À noter que pour les deux tâches, **nous avons fait du tracking avec une valeur de IoU de 0.7** (valeur par défaut dans YOLO [2]).

4.1 Tracking des tasses

Voici les fonctions que nous avons implémentées pour nous permettre d'accomplir la tâche de tracking sur les tasses :

- **readFiles(paths)** : cette fonction prend en paramètre **une liste de chemins de fichiers**, lit ceux-ci et **retourne une liste d'images** associée à ces chemins.
- **parseResults(path, result, target_cl)** : cette fonction prend **en paramètres le *path*** d'une certaine image, **les résultats sur le *tracking*** de cette image et la **classe des objets d'intêrets** (verre dans notre cas). Essentiellement, la méthode extrait toute les informations nécessaires pour la construction du *output* qui sera écrit dans le fichier de résultat et **retourne un tableau de tuples** de tous les verres détectées de la forme suivante : (<frameid> <cupid> <xmin> <ymin> <width> <height>).

- **track(paths,imgs, target_cl = 41.0)** : cette fonction prend **en paramètres une liste de chemins de fichiers, une liste d'images et l'ID de la classe d'intérêt**. Cette fonction construit un tableau (et le retourne) de tuples contenant toutes les détections pour l'ensemble des images. YOLOv8 produit des résultats de *tracking* et ils sont envoyés dans la fonction **parseResults** pour obtenir le tableau de tuples pour une image. Pour chaque image, ce processus est reproduit et un tableau agrège l'ensemble des tableaux d'images individuelles.
- **buildString(frame_tuple)** : **cette fonction prend en paramètre un tuple** et transformes ce tuple en string avec le formatage adéquat. **Elle retourne le tuple formaté sous forme de string**.

L'accomplissement du *tracking* est très simple vu qu'on commence par lire toutes les images avec ***readFiles***. Ensuite, on fait le *tracking* sur ces images avec ***track*** et on obtient un tableau de résultats. les tuples de ce tableau sont passés dans la fonction ***buildString*** pour obtenir le formatage adéquat sous forme de string. Finalement, chaque string est retranscrit dans le fichier *result.txt*.

4.2 Tracking des vidéos de MOT17

Pour ce qui est de l'évaluation de DeepSort, nous avons décidé de le **tester sur le suivi de personnes sur les vidéo de la base de données MOT17**. Pour permettre cela, nous avons choisi d'utiliser l'outil **TrackEval**, qui permet, entre autres, le calcule de la métrique HOTA sur les vidéos de MOT17 [5].

Concretement, on a pris un code trouvé sur *github* qui permet d'intéragir avec Trackeval [6]. **Pour chaque vidéo dans MOT17, le tracking est fait** sur des personnes et on sauvegarde les résultats dans un fichier texte du même nom que la vidéo (ex : MOT17-5-SDP.txt). **Les résultats sont sauvegardés sous le format suivant : (<frameid> <personid> <xmin> <ymin> <width> <height> <1> <-1> <-1> <-1>)**. Ensuite, on créé et on rajoute des dossiers, respectant une certaine structure, dans Trackeval pour pouvoir y déposer nos fichiers textes. **Ainsi, Trackeval saura où aller les chercher** pour faire le calcule des métriques. Maintenant, il ne reste qu'à rouler la commande Trackeval comme suit :

python TrackEval/scripts/run_mot_challenge.py –TRACKERS_TO_EVAL DeepSort –METRICS HOTA –USE_PARALLEL False

On spécifie **le script à rouler** (run_mot_challenge.py), **le nom du dossier** dans lequel aller chercher nos fichier textes (DeepSort) et **la métrique à calculer** (HOTA). Trackeval reçoit ces informations et produit toutes les métrique nécessaires.

5 Présentation des résultats de validation

Dans cette section, nous allons présenter les résultats relatifs aux tests de validation effectués sur MOT17 via l'outil Trackeval.

Cependant, il est nécessaire de se familiariser avec les métriques en jeux. Premièrement, on a **LocA** qui se charge de **calculer l'alignement spatial** entre les boîtes englobantes de la prédiction et la vérité terrain. Plus les boîtes sont alignées, plus LocA sera élevé. Deuxièmement, nous avons **DetA** qui se charge de **mesurer la correspondance entre les détections obtenues et celles présentes dans la vérité terrain** (est-ce que les personnes ont été déctectées comme telles). Pour se faire, le nombre de vrais positifs est divisé par la somme du nombre de vrais positifs, de faux négatifs et de faux positifs. Troisièmement, il y a **AssA** qui **mesure à quel point le suivi des objets à été bien** fait en regardant la conservation des ID à travers le temps. C'est accompli en employant un calcule similaire que pour DetA, mais dans le contexte du suivi d'objets. Par example, nous aurons un vrai positif si on conserve le même ID pour un objet durant le même lapse de temps que dans la vérité terrain. **Finalement, on combine les 3 métriques pour obtenir HOTA** [7].

Vidéo	AssA	LocA	DetA
MOT17-2	41.9	83.3	18.6
MOT17-4	47.4	83.3	34.8
MOT17-5	47.1	79.2	44.3
MOT17-9	37.7	82.9	51.3
MOT17-10	39.9	79.1	32.6
MOT17-11	49.2	85.2	50.4
MOT17-13	38.6	79.1	24.0
Moyenne	45.2	82.4	33.5

TABLE 1 – Valeurs de AssA, LocA, et DetA pour les différentes vidéos de MOT17 ainsi leurs moyennes

Sur le tableau 1, on peut voir les valeurs de AssA, de LocA et de DetA pour chaque vidéo présente dans MOT-17. Premièrement, on constate des assez **bonnes performances** au globale **lorsqu'il est question de localisation**. En effet, tous **les scores se trouvent entre 79.1 et 83.3 avec une moyenne de 82.4**. Ensuite, on remarque une plus grande variation dans les scores d'**AssA** avec une **différence de 11.7 entre les valeurs minimums et maximums**. Cependant nous avons beaucoup plus de hauts scores que de bas scores, ce qui se traduit par **une moyenne de 45.2**. Finalement, nous avons **les scores de DetA qui sont vraiment dans tous les sens**. En effet, on voit que pour **MOT17-2 et MOT17-13, les scores sont seulement de 18.6 et 24.0** respectivement alors que pour **MOT17-9 et MOT17-11 les score sont de 51.3 et 50.4**. Ce manque de constance se reflète dans une moyenne de 33.5, une valeur vraiment au milieu

.

Le tableau 2, nous montre les scores de HOTA et de HOTA(0). **HOTA(0) est un calcule de HOTA avec un seuil de IoU mis à 0.05 lors du calcul du LocA**[7]. Ainsi, avec un IoU plus bas on devrait avoir des valeurs de HOTA supérieures, car on

Vidéo	HOTA	HOTA (0)
MOT17-2	27.8	33.9
MOT17-4	40.5	51.3
MOT17-5	45.5	63.4
MOT17-9	43.8	56.8
MOT17-10	35.9	48.0
MOT17-11	49.7	61.6
MOT17-13	30.2	39.2
Moyenne	38.8	49.5

TABLE 2 – Scores de HOTA et HOTA(0) calculés sur les vidéos de MOT17 ainsi que leurs moyennes

est plus permissible dans nos détections, donc une plus grande quantité de vrais positifs. Ainsi, le taleau 2 nous montre cela puisque **la moyenne de HOTA(0) est 49.5 face à 38.8 pour HOTA**. On remarque que, tout comme DetA, **les plus bas scores sont pour MOT17-2 et MOT17-13 alors que le plus haut score calculé est celui de MOT17-11**

Tracker	HOTA	AssA	DetA
DeepSort	38.8	45.2	33.5
MPNTrack17	46.6	47.3	46.2
eTC17	45.1	46.4	44.1
Tracktor++v2	45.1	45.0	45.3

TABLE 3 – Comparaison des performances de DeepSort avec les 3 meilleurs *trackers* présent dans l'article ayant introduit HOTA [7]

Dans le cas du **tableau 3**, nous sommes allés chercher **les performances des 3 meilleurs** *trackers* évalués par TrackEval lors des tests de la méthode HOTA dans l'article ayant introduit la métrique [7]. **Nous les avons comparés aux résultats produits par DeepSort** pour donner plus contexte ainsi qu'un ordre d'idées concernant nos résultats.

On regardant le tableau 3, on voit qu'en termes d'association, DeepSort n'a pas à rougir. En effet, **son score d'AssA est très proche de ceux des autres** méthodes de suivi. Au contraire d'AssA, **les scores d'HOTA et de DetA de DeepSort sont bien en-dessous** des autres *trackers*. En effet, DeepSort démontre **une différence moyenne de -6.8 et de -11.7** pour ses valeur de HOTA et de DetA face à sa compétition.

6 Discussion des résultats

L'analyse des résultats sur les séquences MOT17 montre les performances variables du suivi selon les métriques considérées. Dans cette section nous allons tenter de comprendre

le pourquoi derrière les chiffres obtenus suite l'usage de Trackeval.

Comme les résultats nous les ont montrés, on constate que **lorsqu'il est question de localisation, DeepSort se débrouille plutôt bien avec une moyenne de 82.4**. **Le mérite revient principalement à YOLOv8**, car, comme il faut le rappeler, DeepSort est dépendant d'un détecteur d'objets externe. Ainsi, nous savons que YOLO est très optimisé pour la reconnaissance d'objets dans le cas de 80 classes incluant la classe "personne" [2]. De ce fait, les très bons résultats de LocA obtenus par DeepSort ne sont pas si surprenants.

Comme nous l'avons mentionné précédemment, **AssA** est une métrique en lien avec la temporalité, puisqu'elle juge, notamment, si les ID ont été conservés à travers le temps. Ainsi, **les défis en rapport avec cette métrique sont la présence d'occlusions et les objets qui sortent (et reviennent) dans le cadre de la vidéo**. **On retrouve ces difficultés** effectivement dans les vidéos **MOT17-9** où des groupes de gens se croisent dans un centre commercial et, donc, se cachent et **MOT17-13** où l'on voit un rue très achalandée filmée par une caméra mobile avec des personnes qui sortent pour ensuite revenir dans le champ. **Ces deux vidéos correspondent aux deux plus bas scores d'AssA** (voir tableau 1). Ces résultats concordent aussi avec le fait que **les meilleurs scores sont obtenus sur MOT17-11 et MOT17-04**, puisque ces vidéos présentent beaucoup moins d'occlusions et dès que les gens sortent du cadre, on ne les revoient plus. Ainsi, le risque de réindentification est moins élevé. **Les points forts de DeepSort resortent dans ces cas de figure** puisque si on se réfère au tableau 3, il se compare aux autres *trackers* avec un score moyen très similaire. Comme évoqué dans la section 3, DeepSort est solide face aux occlusions grâce à l'utilisation d'un filtre de kalman et d'un descripteur d'apparence.

Dans le cas de **DetA**, on voit là où DeepSort commence à pêcher. Tel que discuté, DetA est une métrique de détection qui s'assure que nous avons la bonne sémantique pour nos objets. **Ce qui peut venir poser problème** sont **les objets lointains ou bien qui pourraient paraître petits**. De plus, **la présence de de flou de mouvement** peut faire en sorte qu'un objet ne soit pas en mesure d'être détecté. On retrouve ces challenges dans la majorité des vidéos et notamment chez **MOT17-02 et MOT17-13** où l'on voit plusieurs personnes très loin de la caméra qui sont difficilement perceptibles. En plus de cela, MOT17-13 est une vidéo filmée avec une caméra mobile, ce qui rajoute du flou de mouvement. Les scores de ces vidéos sont les plus bas d'après le tableau 1. **Dans cette exercice, DeepSort est bien en-dessous des autres détecteurs** (voir tableau 3). Cela est assez surprenant puisque YOLO est muni, dans son architecture, d'une pyramide de *features* permettant de détecter les objets d'intérêt à différentes résolutions de l'image [2]. Ainsi, on devrait s'attendre a ce que YOLO puisse mieux se débrouiller.

Dans la même veine, **les scores d'HOTA sont relativement faibles** (voir tableau 2), surtout si l'on compare la valeur moyenne à celle des autres méthodes de suivi du tableau 3. Sachant que les valeurs d'AssA et de LocA sont relativement bonnes, **on peut**

déduire que DetA à une assez grande influence sur la valeur d'HOTA. Comme évoqué dans la section précédente, **les valeurs de HOTA(0) sont plus élevées** que les valeurs d'HOTA (voir tableau 2), **car le α est mis à 0.05**. Avec un seuil d'IoU très faible, on devient très peu sévère et on accepte des chevauchements, qui en temps normal, n'auraient pas passés le filtre [7].

En prenant en considération tout se qu'on vient de discuter par rapport aux résultats de DeepSort sur les vidéos de MOT17, **essayons d'estimer la performance de notre *tracker* sur la vidéo des tasses**. Premièrement, on a vu qu'en termes de localisation et d'association, DeepSort se débrouillait bien. S'il a pu obtenir d'aussi bons résultats avec des vidéos aussi surchargées et comportant une multitude d'occlusions avec parfois la présence d'une caméra mobile, il ne serait pas choquant de dire que dans le cadre de notre vidéo de tasses (qui est bien moins complexe), **DeepSort devrait accomplir la tâche** sans trop de problèmes pour pallier à ces difficultés. Pour continuer dans cette même lignée, **les bons scores de LocA et d'AssA nous donnent espoir concernant l'identification de tasses similaires** puisqu'avec la quantité importante de personnes dans les vidéos MOT17, DeepSort arrive tout de même à en différencier une bonne partie. Quant à la **gestion des objets de tailles variables**, DeepSort ne devrait être trop affecté malgré les faiblesses démontrées par les scores de DetA. En effet, notre vidéo contient des tasses de tailles très adéquates qui ne sont pas très loin de la caméra. **Pour ce qui est du cas des verres transparents, reste à voir**.

En résumé, DeepSort fournit des résultats compétitifs sur l'ensemble des séquences, particulièrement en termes de localisation. Cependant, ses performances en détection et en association sont fortement dépendantes du détecteur YOLOv8 et de la complexité des scènes. Bien qu'efficace, il montre des limites dans les contextes très dynamiques ou fortement encombrés.

7 Bibliographie

[1] Sanyam. (2022) Understanding Multiple Object Tracking using DeepSORT. [En ligne]. Disponible : https ://learnopencv.com/understanding-multiple-object-tracking-using-deepsort/Introduction-to-DeepSORT

[2] Ultralytics. (2024) Ultralytics YOLO docs. [En ligne]. Disponible : https ://docs.ultralytics.com

[3] A. Koudri. (2023) Mastering Deep Sort : The Future of Object Tracking Explained. [En ligne]. Disponible : https ://www.ikomia.ai/blog/deep-sort-object-tracking-guide

[4] R. Kanjee. (2020) DeepSORT — Deep Learning applied to Object Tracking. [En ligne]. Disponible : https ://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104

[5] J. Luiten. (2020) TrackEval. [En ligne]. Disponible :

https ://github.com/JonathonLuiten/TrackEval

[6] kkariste. (2025) TP3. [En ligne]. Disponible : https ://github.com/kkariste/INF6804-Computer-Vision/tree/master/TP3

[7] J. Luiten. (2021) Hota : A higher order metric for evaluating multi-object tracking. International journal of computer vision, 129, 548-578.

[8] S. Raghavan. (2025) DeepSORT Algorithm For Object Tracking. [En ligne]. Disponible : https ://www.linkedin.com/pulse/deepsort-algorithm-object-tracking-shashank-v-raghavan–aaggc : :text=DeepSORT