Laboratoire 3 INF6804 : Vision par ordinateur

PAR

Ariste Kwawe-K-Arnov (2317869) groupe : gr12

TRAVAIL PRÉSENTÉ À KHALIL SABRI

Le 12 AVRIL 2024

1 Introduction

Le présent laboratoire consiste à la réalisation de suivi multi objets sur des séquences d'images fournies par le professeur sur moodle à partir d'une solution qu'on doit implémenter. Notre solution devra en retour être appliquée sur le jeu de données publics MOT17 ou MOT20 afin de l'évaluer par la métrique HOTA. Pour arriver à cela, nous avons tout d'abord présenté notre solution en donnant ses avantages et ses inconvénients. Par la suite, nous avons identifié les potentiels difficultés en suivi de la séquence d'images sur moodle. Enfin nous avons présenté les résultats de notre solution sur MOT17 et avons discuté les résultats.

2 Description en profondeur de votre solution

En suivi multi objet (MOT), on s'intéresse particulièrement à la résolution de deux problèmes majeurs qui sont la détection et l'association. De cette façon, on cherche alors à attribuer à chaque objet présent dans une séquence d'images, une étiquette unique tout au long de cette séquence [1]. La solution proposée ci-dessous utilise Ultralytics YoloV8 pour la détection et ByteTrack pour le suivi.

2.1 YOLOv8

YOLOv8 est une version non propriétaire de YOLO (You Only Look Once), mais plutôt de Ultralytics, qui a dévéloppé récemment la version 9. Elle s'appuie sur les premières versions afin de fournir un modèle plus complet et plus efficace. Nous allons présenté la prmière version de YOLO, ensuite la version 5 développée par Ultralytics et terminer par YOLOv8.

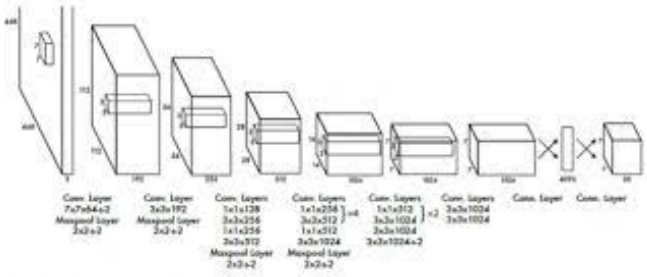

Figure 1 – Architecture YOLOv1

Dans sa version originale YOLOv1, l'architecture comprend 24 couches de réseaux convolutifs et les deux dernières couches sont totalement connectées. La base du réseau est formée par les 20 premières couches et il a été ajouté en plus un couche de mise en commun moyenne et une couche entièrement connectée. Pour optimiser les performances pendant les phases de pré-entraînement et de validation, YOLOv1 utilise la descente stochastique du gradient et il en ressort que la fonction de perte est une somme de pertes de localisation et de perte de classification. La première mesure l'erreur entre les boîtes englobantes prédites et celles de la vérité de terrain alors que la deuxième estime l'erreur entre

les classes prédites et celles de la vérité de terrain. Les deux coefficient en λ permettent de réguler l'ampleur des différentes composantes [2].

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Figure 2 – Fonction de perte

YOLOv5, plus performante et rapide que l'état de l'art est une architecture qui apporte une nouvelle structure de modèle en trois composants à savoir, la tête, le cou et la colonne vertébrale. L'architecture de réseau Darknet53 qui est sa colonne vertébrale consiste à extraire les caractéristiques. Elle se distingue par l'utilisation de petites fenêtres de filtrage et de connexions résiduelles. Les connexions partielles transversales favorisent un flux de gradient accru tout en réduisant les calculs. Le cou améliore l'information spatiale et sémantique en agrégeant et affinant les caractéristiques issues de la colonne vertébrale ; il fait le lien entre la tête et la colonne vertébrale. La tête de YOLOv5 se compose de trois branches de taille de grilles respectives 13x13, 26x26 et 52x52. Chacune des grilles prédit trois rectangles englobants alors que de chaque tête, il en résulte des boîtes de délimitation, des scores de confiance et des probabilités de classes. Les chevauchements sont filtrés grâce à la suppression non maximale utilisée par le réseau. Afin de prédire l'emplacement et la taille des objets dans une image, YOLOv5 intègre des boîtes d'ancrage, des boîtes de délimitation de taille fixe utilisées . Au lieu de prédire des boîtes de délimitation arbitraires pour chaque instance d'objet, le modèle prédit les coordonnées des boîtes d'ancrage avec des rapports d'aspect et des échelles prédéfinis et les adapte à l'instance d'objet [3].

YOLOv8 bien que possédant la même architecture que les autres, elle apporte une amélioration considérable par rapport à YOLOv5. D'après le lien suivant `https://github.com/ultralytics/ultralytics/issues/189`, elle commence d'abord par remplacer le module C3 par le module C2f ensuite, la convolution 6x6 par une convolution 3x3 dans le Backbone. Elle supprime les convolutions 10 et 14 dans l'architecture de YOLOv5 et remplace la première convolution 1x1 par une convolution 3x3 au niveau de l'étranglement. Il supprime la branche d'objectivité et utilise la tête découplée. YOLOv8 est une gamme complète pour la vision par ordinateur. Elle intègre les applications de *détection, de segmentation, d'estimation de pose, de suivi et de classification*. Elle se distingue par plusieurs modèles :

- YOLOv8n, n = nano

- YOLOv8m, m = medium
- YOLOv8s, s = small
- YOLOv8x, x = très large

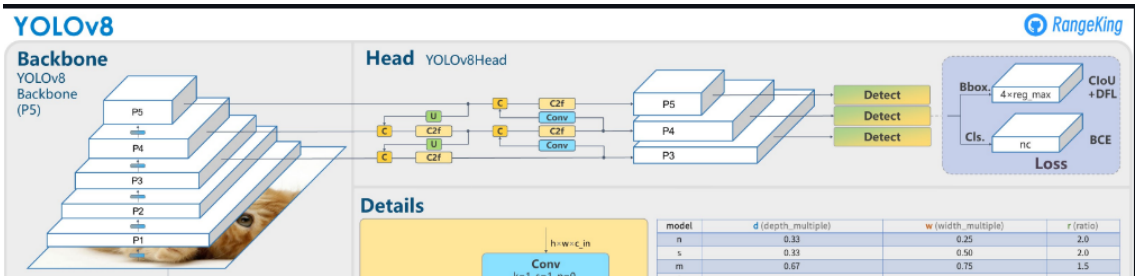

Figure 3 – Architecture YOLOv8

2.2 ByteTrack

Une fois la détection terminée suivant un certain seuil, ses résultats et la séquence vidéo sont passées en entrée au traceur ByteTrack qui lui fournit en sortie les pistes de la vidéo. Le contenu d'une piste n'est rien d'autre que le rectangle englobant et l'identifiant (ID) d'un objet dans chaque image. On distingue ici les rectangles englobants à haut et à faible niveau de confiance qui seront utiles pour les deux étapes d'association alors que les boîtes englobantes d'arrière-plan seront éliminées [4].

Première association : elle commence à comparer les boîtes de détection à score élevé de l'image actuelle avec toutes les pistes prédites perdues ou non de l'image précédente obtenues grâce au filtre de Kalman. Pour y parvenir, on calcule la distance entre les caractéristiques IoU ou ReID entre les boîtes prédites et celles détectées en utilisant une mesure d'affinité telle que la similarité cosinus. Après cela, l'appariement peut être fait suivant les scores les plus proches. Pour terminer cette première association, on applique l'algorithme hongrois en se basant sur cette similarité. Les détections qui n'ont pas pu être appariées sont alors dite à faible score et sont sauvegarder avec les pistes aussi non mises en correspondance [4][5].

Deuxième association : en utilisant cette fois un score de mise en correspondance inférieur à celui de la première association, les rectangles englobant à faible score sont reliés aux boîtes prédites restantes. En effet, on essaye dans cette partie de résoudre les problèmes d'occlusion et de fort mouvement dans les séquences d'images qui représentent les boîtes de détection à faible score. De même l'algorithme utilisé ici ne se base que sur l'IoU comme similarité de deuxième association toujours pour résoudre ces problèmes, car ReID serait trop robuste ou sévère. en faisant cela, on conserve ainsi l'identité des objets quand ils sont masqués ou peu visibles [4][5].

Gestion du cycle de vie des pistes : après les étapes d'association, les prédictions qui n'ont pas été mises en correspondance sont marquées comme des pistes perdues alors que les détections homologues sont

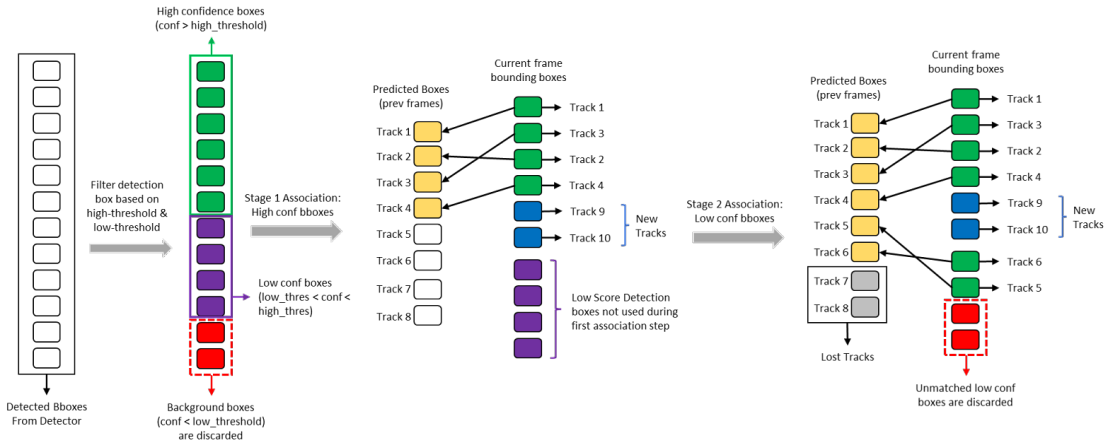

Figure 4 – Algorithme BYTETrack [5]

supprimées. Chaque piste perdue n'est conservée que si elle existe depuis environ un nombre de trames égale à 30 avant la prochaine prédiction. Enfin, de nouvelles pistes sont initialisées pour les objets détectés qui n'apparaissaient pas dans l'image précédente [1][4].

La figure ci-dessous montre la supériorité de ByteTrack par rapport aux autres trackers.

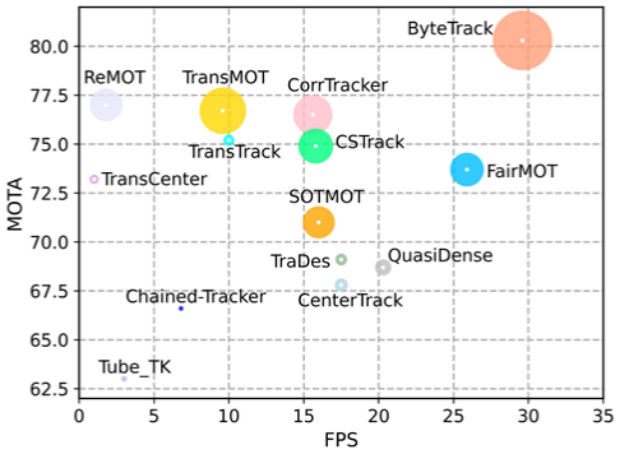

Figure 5 – Comparaisons MOTA-IDF1-FPS de différents trackers sur l'ensemble de tests de MOT17 [4]

Avantages et inconvénients : la stratégie de ByteTrack permet de gérer les situations où les objets se chevauchent et même lorsqu'ils sont peu occultés. Des objets qui sont sortis de la séquence vidéo sont ré-identifiés lorsqu'ils apparaissent de nouveau et tout nouveau objet qui n'était pas présent est suivi lors de son apparition. Cependant certaines pistes (tracklets) ne sont pas associées parce qu'ils ne correspondent pas à des boîtes de détection appropriées en cas d'occlusion sévère, de flou du mouvement et de changement d'échelle. Autrement dit, la robustesse du suivi par ByteTrack dépend aussi en majorité du détecteur utilisé [4].

3 Identification des difficultés dans la séquence sur Moodle

Dans la vidéo fournie, plusieurs difficultés subsistent :

- Ré-identification (toutes les trames) : certaines tasses qui étaient présentes dans les images précédentes disparaissent pour apparaître un peu plus tard dans les trames suivantes seront difficiles à ré-identifier à leur nouvelle apparition.
- Occlusions : lorsque les tasses se cachent entre elles ou sont cachées par d'autres objets, le suivi risque d'être plus ou moins difficile selon l'ampleur de l'occlusion.
- Le flou : les tasses qui sont floues ne seront pas détectées et par conséquent leur suivi (frame162.jpj - frame191.jpg).
- Forme : certaines tasses ne sont pas filmées en entier et parfois lorsqu'elles sortent du champ de vue, elles rentrent progressivement.
- Objets éloignés dans la scène (Ex : frame693.jpg-frame710.jpg) : Les tasses éloignées du champ de la caméra seront faiblement détectés et seront alors considérées comme arrière plan.
- Caméra mobile (toutes les images) : Lorsque la caméra se déplace, la perspective des images dans la scène change.

4 Justification de la méthode par rapport aux difficultés identifiées

Nous citons ci-dessous, les faiblesses et les forces de notre solution :

- Ré-identification : ByteTrack utilise la Re-ID lors de l'association des données et par conséquent permet la ré-identification. Cependant, elle peut avoir du mal à le faire à la nouvelle trame où l'objet réapparaît, mais plutôt après un certains nombres trames.
- Occlusions : ByteTrack est robuste aux occlusions partielles. en cas d'occlusion total, le détecteur YOLOv8 peut ne pas détecter un objet, car il sera caché par un autre.
- Le flou : il sera impossible pour le détecteur et le trackeur de poursuivre les objets, car ils sont moins nets. En effet, les réseaux qui les constituent n'ont pas été pré-entraîné sur de tels objets. Leur forme et contour ne sont pas identifiable dans ce cas.
- Résolution : YOLOv8 a été entraîné sur les images de la dataset COCO de résolution 640. La taille des images de la séquence sur moodle étant plus grande la vitesse de traitement serait alors plus grande.
- Forme : YOLOv8 est capable de détecter les objets de différentes formes similaires aux jeux de données de l'entraînement. Cependant certaines tasses partiellement visibles ou des angles de vue inhabituels affecteront négativement la détection.
- Objets éloignés : Plus les tasses sont éloignées du champ de vue de la caméra, moins elles seront détectées.

5 Description de l'implémentation utilisée

Notre code se base sur Ultralytics qui fournit à la fois le détecteur YOLOv8 et deux modèles pour le suivi : BotSORT et ByteTrack et comme mentionné nous avons utilisé ByteTrack. Nous nous sommes donc inspirés du code sur le site web ou github et en s'appuyant sur les exemples de code dans les commentaires `https://docs.ultralytics.com/` ou `https://github.com/orgs/ultralytics/discussions`.

Nous commençons par installer leur module avec la commande *pip install ultralytics* pour les dépendances. Par la suite, on charge on charge un modèle pré-entraîné en important le module YOLO grâce à la ligne *model = YOLO('yolov8n.pt')*. Enfin on démarre le tracking en appelant la méthode track sur notre modèle : *results = model.track(paramètres)*. Les paramètres importants à rappeler que nous avons utilisés sont :

- Dataset : COCO128 qui est l'ensemble de données par défaut.
- tracker : ByteTrack.
- conf : représente le score minimal de confiance pour qu'une tasse puisse être détectée. Un seuil bien choisir permettra de réduire les faux positifs. Nous l'avons laissé à *conf = 0.25* qui est le score par défaut.
- IoU : c'est le seuil d'intersection au dessus de l'union pour la suppression non maximale (NMS). Plus sa valeur est faible, plus les boîtes qui se chevauchent sont éliminées, on réduit alors les doublons. Nous utilisons là encore la valeur par défaut *iou = 0.7*
- imgsz : cette valeur représente la taille de l'image pour l'inférence. Plus le choix est meilleur plus les coordonnées des boîtes englobantes sont précises. On a préféré la valeur par défaut *imgsz = 640*.
- classes : une liste d'entier qui permet de choisir les objets d'intérêts. 0 pour person et 41 pour cup.

Afin de voir comment agir les valeurs IoU, imgsz et conf nous avons fait plusieurs tests qui sont donnés dans le tableau qui suit.

Table 1 – Conf est variable et le reste fixe.

(conf,iou, imgsz)	(0.25, 0.7, 640)	(0.15, 0.7, 640)	(0.5, 0.7, 640)
Identifications	62	58	70
Détections	4361	4726	3862

Table 2 – imgsz est variable et le reste fixe.

(conf,iou, imgsz)	(0.25, 0.7, 736)	(0.25, 0.7, 800)	(0.25, 0.7, 1024)
Identifications	56	57	46
Détections	4538	5032	5089

Table 3 – IoU est variable et le reste fixe.

(conf,iou, imgsz)	(0.25, 0.6, 640)	(0.25, 0.85, 640)	(0.25, 0.95, 640)
Identifications	59	79	166
Détections	4354	4397	4785

On termine en enregistrant les résultats de tracking dans le fichier results.txt dans le répertoire courant.

NB : La même démarche a été faite pour les vidéos du jeu de données MOT17.

Pour l'évaluation HOTA des résultats sur les séquences vidéo de MOT17, nous avons utilisé TrackEval qui doit être téléchargé localement dans le répertoire de travail disponible à l'adresse `https://github.com/JonathonLuiten/TrackEval.git`. Les évaluations ont été faites sur l'ensemble de jeu de données de MOT17.

Un dossier data `https://omnomnom.vision.rwth-aachen.de/data/TrackEval/data.zip` nécessaire à l'évluation a été téléchargé et placé dans le répertoire TrackEval. Plus loin dans ce dossier, en suivant le chemin : data/trackers/mot_challenge/MOT17-train, on remplace MPNTrack par le nom de notre de modèle de tracking ByteTrack et les fichiers dans ce dossier sont les résultats de notre méthode. La commande suivante a été appliquée pour avoir les résultats :
subprocess.run(["python3", "TrackEval/scripts/run_mot_challenge.py", "–USE_PARALLEL", "False", "–METRICS", "HOTA", "–TRACKERS_TO_EVAL", "ByteTrack"]).

NB : Les formules utilisées pour l'évaluation HOTA se trouve dans l'article de l'auteur et ne sont pas présentées ici [6].

6 Présentation des résultats de validation

Bien que nous avons effectué les tests sur l'ensemble des données de MOT-17, les résultats présentés dans le tableau ci-dessous ne fournissent que pour quelques séquences et le paramètre HOTALocA(0) n'est pas mentionné ici afin de limiter le texte.

Table 4 – Scores HOTA pour les 09 premières séquences MOT17.

Séq	HOTA	DetA	AssA	DetRe	DetPr	AssRe	AssPr	LocA	OWTA	HOTA(0)	LocA(0)
MOT17-02-DPM	25.029	14.91	42.041	15.136	82.065	43.8	83.743	84.364	25.223	30.466	80.969
MOT17-02-FRCNN	25.029	14.91	42.041	15.136	82.065	43.8	83.743	84.364	25.223	30.466	80.969
MOT17-02-SDP	25.029	14.91	42.041	15.136	82.065	43.8	83.743	84.364	25.223	30.466	80.969
MOT17-04-DPM	37.17	25.49	54.366	26.223	82.181	56.812	87.116	84.751	37.733	45.686	80.567
MOT17-04-FRCNN	37.17	25.49	54.366	26.223	82.181	56.812	87.116	84.751	37.733	45.686	80.567
MOT17-04-SDP	37.17	25.49	54.366	26.223	82.181	56.812	87.116	84.751	37.733	45.686	80.567
MOT17-05-DPM	41.39	40.1	42.911	43.333	71.331	50.234	65.987	78.941	43.105	58.744	71.554
MOT17-05-FRCNN	41.39	40.1	42.911	43.333	71.331	50.234	65.987	78.941	43.105	58.744	71.554
MOT17-05-SDP	41.39	40.1	42.911	43.333	71.331	50.234	65.987	78.941	43.105	58.744	71.554
Combinée	34.52	26.83	46.439	28.23	78.52	50.282	80.282	82.68	35.353	44.965	77.697

Pour mieux connaître les performances de détection de notre solution face au terrain de vérité, la table suivante présente notre nombre de détections et des identifications retrouvées.

Table 5 – Notre solution vs groundtruth de MOT17.

Séquences	Détections	Détections gt	Identifications	Identifications gt
MOT17-02-DPM	3427	18581	35	62
MOT17-02-FRCNN	3427	18581	35	62
MOT17-02-SDP	3427	18581	35	62
MOT17-04-DPM	15175	47557	66	83
MOT17-04-FRCNN	15175	47557	66	83
MOT17-04-SDP	15175	47557	66	83
MOT17-05-DPM	4202	6917	116	133
MOT17-05-FRCNN	4202	6917	116	133
MOT17-05-SDP	4202	6917	116	133
Combiné	68412	219165	651	834

La figure ci-dessous présente les performances en fonction du coefficient alpha choisit.

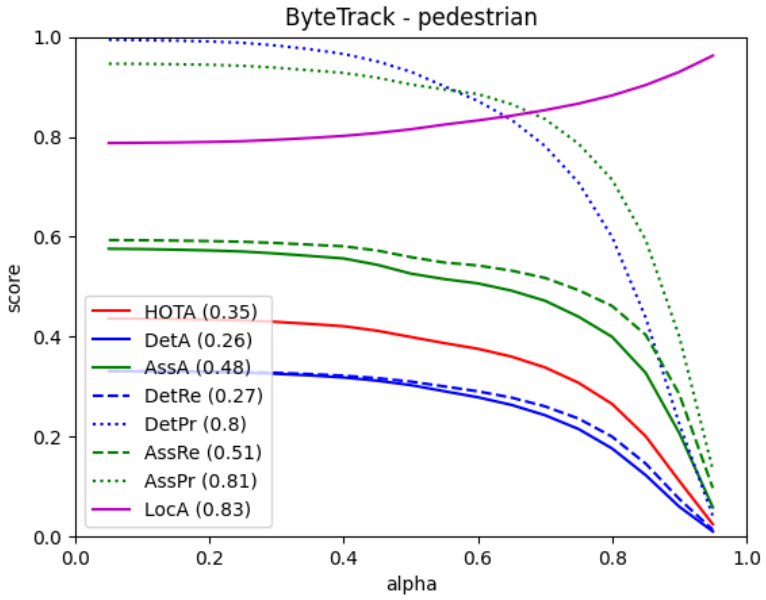

Figure 6 – Scores HOTA en fonction de alpha

7 Discussion des résultats

Le seuil ou le paramètre α permet de définir la tolérance en précision de la métrique HOTA. Pour HOTA(0), $\alpha = 0$, la précision de localisation n'est pas considérée. Autrement dit, pour admettre qu'une prédiction est correcte, les coordonnées des rectangles englobants les objets ne nécessitent pas de s'aligner totalement avec ceux du terrain de vérité (groundtruth). On dit alors que HOTA(0) est plus tolérant ou indulgent car aussi longtemps que les autres aspects comme l'association des prédictions restent correspondent à ceux du groundtruth alors les résultats HOTA(0) sont bons, malgré que dans le cas notre solution, on obtient pas de bons scores en général. Lorsque α tend à augmenter, la situation change, on augmente en précision et donc le score HOTA est en baisse d'après la figure. On comprend par là que la méthode HOTA évalue les trackers en combinant la détection, l'association et la localisation.

Pour toutes nos séquences les résultats sont catastrophiques, bien que les trois dernières semblent se rapprocher d'un score moyen. Ces résultats sont dues aux détections et donc à YOLOv8. Le nombre d'objets détectés avec notre solution pour les séquences MOT17-02-DPM, MOT17-02-FRCNN, MOT17-02-SDP ne représentent que les 1/6 des détections du groundtruth. De toute façon, la précision de détection globale DetA reste faible pour notre solution. Le rappel de détection DetRe montre que nous trouvons moins d'objets de la vérité de terrain et que nous avons beaucoup de détections erronées d'après la précision de détection DetPr.

Pour expliquer ces performances, on peut s'attarder aux difficultés cités à la section 3. Dès les premières images de la séquence MOT17-02-DPM par exemple, la plupart des piétons sont éloignés du champ de vu de la caméra, il est impossible pour YOLO de les détecter, car ils sont assimilés à l'arrière-plan. Par

contre, le détecteur identifie bien les personnes placées plus près de la caméra. On peut donc ici confirmer l'incompétence de notre modèle face à cette difficulté.

On peut encore considérer les occlusions ou les groupes de piétons qui se forment n'est pas avantageux pour la détection. Les piétons qui cachent ceux qui sont devant eux induisent le détecteur en erreur (ex. MOT-17-05-DPM, 000553.jpg - 000567.jpg), mais nous pensons que ce problème est plus attribué à la position de la caméra. En effet, si on regarde en hauteur d'une bonne distance, on peut facilement détecter les têtes ou l'ensemble du corps de chaque piéton ce qui améliorait les résultats. On peut aussi observer les voitures qui passent cachées les piétons qui se trouvent de l'autre côté de la route et qui veule traverser.

D'une façon générale, on estime que pour améliorer les résultats, il aurait été préférable d'entraîner le détecteur sur les images de MOT17 ou de la séquence vidéo sur moodle. Mais faire de cette façon est moins pratique, on suppose alors que pour que le détecteur soit plus performant, on doit entraîné YOLO sur les difficultés qu'on rencontre en suivi multi objets. Dans ce cas, il faudrait par exemple annoter les objets flous et d'entraîner le réseau sur eux pour que même en cas de cette difficulté, qu'il puisse avoir détection.

De la même façon, on peut se pencher sur les paramètres au niveau de l'implémentation. Par exemple, la taille des images pour l'inférence devrait être plus ou moins similaire à celle des images fournies ou à la résolution de la caméra. De même, le score de confiance et l'IoU devrait être ajustés de façon à obtenir les meilleurs résultats d'après les tests d'implémentation à la section 5.

On peut conclure au vu de nos résultats que notre solution ne fournira pas de bon résultats pour la séquence de moodle, il faudrait alors ajuster les paramètres, on observe qu'on a beaucoup de détections erronées et de ré-identifications mal faites.

8 Conclusion

Au terme de ce laboratoire, nous avons abordé le côté pratique du suivi multi objet en implémentant une solution et en l'évaluant grâce à la métrique HOTA. Les résultats obtenus dépendent majoritairement des difficultés observées dans les séquences d'images à notre disposition telles que les occlusions, les objets flous, la ré-identification. Notre solution qui est une combinaison de YOLOv8 et ByteTrack bien que prônée aujourd'hui comme l'une des meilleures n'a pas bien fonctionné. On peut conclure que ces performances négatives résultent de l'implémentation où les valeurs utilisées ne sont pas appropriées c'est le seuil de confiance conf, l'IoU et la taille de l'image pour l'inférence imgsz.

Références

[1] Allan Kouidry. *ByteTrack : The Future of Multi-Object Tracking in Computer Vision. ByteTrack : l'avenir du suivi multi-objets en vision par ordinateur*. https://www.ikomia.ai/blog/bytetrack-multi-object-tracking. (2023, novembre 21).

[2] Joseph Redmon et al. *You only look once : Unified, real-time object detection*. 2016.

[3] Dillon Reis et al. *Real-Time Flying Object Detection with YOLOv8*. https://arxiv.org/pdf/2305.09972.pdf. (2023, mai 17).

[4] Yifu Zhang et al. « ByteTrack : Multi-Object Tracking by Associating Every Detection Box. ByteTrack : Suivi de plusieurs objets en associant chaque boîte de détection ». In : (2022).

[5] Ben Le. *An Introduction to BYTETrack : Multi-Object Tracking by Associating Every Detection Box. Introduction à BYTETrack : Suivi de plusieurs objets en associant chaque boîte de détection*. https://www.datature.io/blog/introduction-to-bytetrack-multi-object-tracking-by-associating-every-detection-box. (2023, novembre 08).

[6] Jonathon Luiten et al. « HOTA : A Higher Order Metric for Evaluating Multi-Object Tracking ». In : *International Journal of Computer Vision* (2020), p. 1-31.