



**INF6804 - Visions par ordinateur**

**Hiver 2024**

**Travail Pratique 03**

**Groupe 01**

**Alexandre Gélinas - 2083465**

**Elizabeth Michaud - 2073093**

**Soumis à : Khalil Sabri**

**Guillaume-Alexandre Bilodeau**

**Date :**

**15 avril 2024**

## **Table des matières**

<b>1. Description détaillée de la méthode.....</b>	<b>2</b>
<b>2. Identification des difficultés dans la séquence sur Moodle.....</b>	<b>2</b>
<b>3. Justification de la méthode par rapport aux difficultés identifiées.....</b>	<b>5</b>
<b>4. Description de l'implémentation utilisée.....</b>	<b>6</b>
<b>5. Présentation des résultats de validation.....</b>	<b>7</b>
<b>6. Discussion des résultats.....</b>	<b>7</b>
<b>Références.....</b>	<b>11</b>

## 1. Description détaillée de la méthode

La méthode de *multiple object tracking* que nous avons choisi est ByteTrack. Cette méthode se base tout d'abord sur la détection des objets à l'aide d'un modèle existant comme YOLO ou Fast-RCNN. La méthode que nous avons utilisée se base sur une méthode de segmentation par instances, soit YOLOv8, pour la détection d'objets. Cette étape permet d'obtenir tous les objets détectés dans chacune des trames d'une vidéo avec leur boîte englobante (*bounding box*) et une étiquette (*label*) qui catégorise l'objet en question. De plus, chaque objet se fait attribuer un score de confiance (*confidence score*). ByteTrack prend d'abord les objets qui ont un score de confiance élevé et effectue le suivi (tracking) de ceux-ci afin de les identifier au travers des différentes trames. Ensuite, contrairement à plusieurs autres méthodes de suivi de multiples objets d'intérêt, ByteTrack prend aussi en compte les objets qui ont un score de confiance bas. Si un objet détecté avec un score de confiance bas correspond à la position et à l'apparence attendues d'une détection des trames précédentes, alors les deux détections sont associées. Cette étape est particulièrement utile pour faire le suivi d'objets qui sont occultés. ByteTrack arrive à trouver les positions attendues des objets détectés grâce à une étape de prédiction de mouvement qui utilise le filtre de Kalman.

Le but de ce travail pratique est d'évaluer les performances de la méthode ByteTrack pour le suivi de multiples objets d'intérêt en l'évaluant sur une vidéo de tasses et sur un ensemble de données public, soit MOT17.

## 2. Identification des difficultés dans la séquence sur Moodle

La première difficulté présente dans la séquence vidéo sur Moodle est le **changement d'échelle (scale)** des tasses. À plusieurs moments, la caméra se rapproche ou s'éloigne d'une ou de plusieurs tasses. C'est par exemple le cas au tout début de la vidéo et à la toute fin. Par exemple, à la fin de la vidéo, on peut voir une tasse blanche posée sur une tour et vers laquelle la caméra se rapproche graduellement.

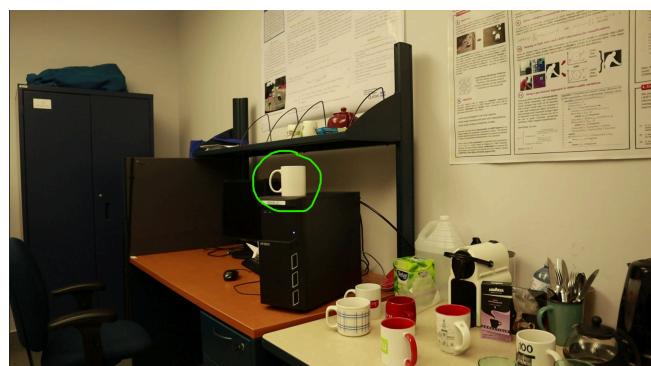


Figure 1. Trame 1170 provenant du fichier *frame1170.jpg* de l'ensemble de données fourni



Figure 2. Trame 1380 provenant du fichier *frame1380.jpg* de l'ensemble de données fourni

Les tasses encerclées en vert dans la figure 1 et la figure 2 correspondent à la même tasse à des échelles différentes. Il faut alors s'assurer que notre modèle est capable de reconnaître qu'il s'agit de la même tasse.

La deuxième difficulté présente dans la séquence vidéo fournie est **l'occlusion des tasses** entre elles. Cette difficulté est surtout présente lorsqu'on voit beaucoup de tasses en même temps. Par exemple, c'est le cas de la trame 300.

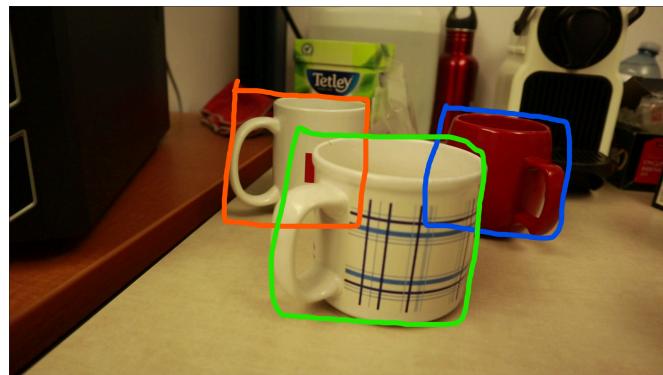


Figure 3. Trame 300 provenant du fichier *frame300.jpg* de l'ensemble de données fourni

Sur la figure 3, on peut remarquer que la tasse encadrée en vert occulte la tasse encadrée en orange et la tasse encadrée en bleu. Il est important de choisir une méthode de suivi de multiples objets d'intérêt qui permet de détecter et de suivre les objets occultés.

La troisième difficulté présente dans la séquence vidéo est la rotation dans le plan de la caméra du verre transparent lorsque la personne dans la vidéo verse le contenu du verre dans un autre verre. À ce moment-là, un des verres est penché, ce qui cause une rotation dans le plan de celui-ci.



Figure 4. Trames 700, 750 et 880 provenant respectivement des fichiers *frame700.jpg*, *frame750.jpg* et *frame880.jpg* de l'ensemble de données fourni

On peut voir aux trames présentes sur la figure 4 (et aux trames entre celles-ci) que le même verre transparent encerclé en rouge subit une rotation.

La quatrième difficulté est la présence de **frou** (*blur*) dans la vidéo. Cela peut rendre la détection des tasses plus difficile, car l'objet est moins facilement reconnaissable. Ce cas se produit aux alentours de la trame 160.



Figure 5. Trame 160 provenant du fichier *frame160.jpg* de l'ensemble de données fourni

Comme le montre la figure 5, plusieurs tasses sont présentes en arrière-plan (encerclées en vert) alors que le focus de la caméra est sur l'avant-plan. Il peut être très difficile pour un modèle de reconnaître et de suivre ces tasses surtout lorsqu'elles sont occultées en plus d'être floues.

Finalement, la cinquième difficulté est que les **tasses sortent parfois du cadre de la caméra** pour revenir un peu plus tard. Comme il s'agit des mêmes tasses, il pourrait être souhaitable qu'elles se fassent attribuer le même identifiant une fois qu'elles reviennent dans l'image. C'est une situation qui se produit entre les trames 370 et 422 environ.



Figure 6. Trames 370, 380 et 422 provenant respectivement des fichiers *frame370.jpg*, *frame380.jpg* et *frame422.jpg* de l'ensemble de données fourni

À l'aide de la figure 6, on peut comprendre que la tasse rouge encerclée en vert est soulevée, sortie du cadre de la caméra, puis ramenée dans l'image.

### 3. Justification de la méthode par rapport aux difficultés identifiées

En ce qui concerne le choix de notre solution, nous avons choisi Byte Track, car celle-ci permet de facilement changer de méthode de description des images, et donc de résoudre une grande partie de nos problèmes simplement en prenant un descripteur de meilleure qualité. En ce qui concerne le premier problème, le changement d'échelle sur l'objet sera pris en charge par YOLOv8, qui est actuellement l'un des meilleurs détecteurs. Bien que l'objet pourrait parfois ne pas être correctement identifié, la précision de YOLOv8 est assez haute pour réduire ce problème. Ainsi, le problème ne sera pas complètement résolu, mais grandement diminué.

En ce qui concerne la deuxième difficulté, celle-ci sera plus difficile à traiter, voire impossible à traiter complètement. Comme l'occlusion peut cacher une partie voir la totalité d'un autre objet, il est très difficile de garder le suivi sur l'objet alors qu'il n'est plus visible. Bien que ByteTrack utilise une méthode afin de comparer d'anciens objets comme source pour la comparaison et le suivi, il y aura toujours des moments où YOLOX ne pourrait pas détecter l'objet et empêchera le reste des méthodes de bien fonctionner. Bref, un objet non visible est impossible à détecter avec assurance.

En ce qui concerne la troisième difficulté, la rotation dans le plan est toujours difficile pour YOLOX à moins de spécifiquement tourner la vidéo par nous-mêmes. De ce fait, ce sera aussi un problème qui restera difficile à résoudre, peu importe la méthode. De plus, l'entraînement des données apporte aussi sa valeur dans ce problème. Si le modèle n'a jamais été entraîné sur des images de tasse à l'envers, il sera très difficile pour celui-ci de détecter une image de tasse à l'envers. Bien que nous aurions pu entraîner par nous-mêmes le modèle, il aurait été très difficile de trouver un jeu de données qui correspondait à nos attentes et nous avons donc voulu garder un jeu de données sensiblement fiable et accessible.

Pour ce qui est de la quatrième difficulté, la présence de flou sera assez bien résolue par ByteTrack. Comme ce modèle prend en compte les détections ayant une faible confiance pour la traçabilité des objets dans plusieurs frames, il est plus facile pour un objet flou de pouvoir être détecté correctement selon la position où il se trouve dans l'image. Le modèle utilise la dernière position de l'objet afin d'influencer la détection sur plusieurs frame de l'objet et de pouvoir bien identifier un objet même s'il est moins bien détecté par le détecteur. De ce fait, la présence de flou devrait être quand même bien résolue. Cependant, si la première apparition d'un objet contient cette présence de flou, il sera beaucoup plus difficile de détecter l'objet pour ByteTrack et si l'objet est trop intensément flou, cela apportera une difficulté supplémentaire.

Pour la cinquième difficulté, elle risque d'être difficile à traiter pour ByteTrack. En effet, lorsqu'un objet n'est plus détectable, car il ne fait plus partie de la vidéo pendant plusieurs trames d'affilée, la méthode ByteTrack aura de la difficulté à comprendre qu'il s'agit du même objet qui était présent 20 ou 30 trames auparavant. Elle sera alors portée à attribuer un identifiant différent à l'objet qui revient dans la vidéo. Cependant, si l'objet ne sort pas complètement du cadre de la caméra, il reste détectable et correctement identifiable pour ByteTrack. Le modèle choisi est capable de faire le suivi des objets entre les trames même s'il n'est pas totalement sûr qu'il a

bien identifié l'objet grâce à son étape de prédiction. Cependant, si l'objet n'est plus présent sur les trames, il n'effectuera pas de prédiction sur cet objet et ne prédira pas que l'objet puisse revenir dans la vidéo.

#### 4. Description de l'implémentation utilisée

L'implémentation que nous avons utilisée nous vient directement du GitHub de xjl-le avec son projet YOLOv9 ([https://github.com/xjl-le/yolov9\\_tracking?tab=readme-ov-file](https://github.com/xjl-le/yolov9_tracking?tab=readme-ov-file)) même si son implémentation utilise YOLOv8. Nous avons tout simplement utilisé le Google Collab du site afin de réaliser notre expérience. Afin de pouvoir facilement utiliser ce code, nous avons dû mettre notre entrée sous forme de fichier. Nous avons ainsi utilisé la librairie OpenCV et la fonction VideoWriter afin de convertir nos images en une seule vidéo de format AVI. Une fois cela fait, nous avions simplement besoin d'ajuster les paramètres du code existant afin de lancer l'exécution de notre méthode avec le bon modèle. Nous avons ainsi utilisé le modèle YOLOv8 pour la détection (avec le jeu de donnée YOLOv8l), osnet\_x0\_25\_msmt17 comme modèle de réidentification et ByteTrack comme modèle de suivi. Nous avons filtré nos résultats avec la classe 41 qui représente les “cups” et enregistré la sortie sous le format désiré avec l'option “--save-txt”. Le choix de ses paramètres a été fait selon les paramètres déjà présents dans le code source ainsi que du classement au MOT17 et MOT20. De ce fait, nous avons laissé le Osnet\_x0\_x25\_msmt17, mais avons modifié le modèle pour qu'il soit un peu plus rapide et utilisant le meilleur tracking possible des classements MOT. Pour ce qui est des données préentraînées, nous avons utilisé des données assez performantes sans nécessairement être les meilleures et étant quand même rapides afin de pouvoir facilement corriger le code si nécessaire. En ce qui concerne le seuil de confiance pour la détection des objets, nous avons utilisé un seuil de 0.20, car il était déjà présent dans le code source et nous donnait des résultats excellents.

En ce qui concerne l'évaluation de notre modèle, nous avons utilisé le même code source que le modèle, qui se base sur le GitHub de TrackEval afin d'évaluer un modèle sous plusieurs types de classement. Nous avons utilisé les mêmes paramètres que pour notre modèle afin de pouvoir bien l'évaluer. Pour le benchmark, nous avons utilisé MOT17.

## 5. Présentation des résultats de validation

Tableau 1: Résultat de performance sur MOT17

MOT17-pedestrian	HOTA	DetA	AssA	LocA	HOTA(0)
MOT17-02-FRCNN	33.206	24.564	45.1	86.686	38.14
MOT17-04-FRCNN	41.731	32.692	53.308	86.814	50.444
MOT17-05-FRCNN	42.656	41.307	44.419	78.993	59.422
MOT17-09-FRCNN	55.857	57.727	54.145	84.264	71.027
MOT17-10-FRCNN	45.976	43.936	48.289	80.124	61.839
MOT17-11-FRCNN	59.008	52.593	66.413	85.503	74.143
MOT17-13-FRCNN	39.175	32.501	47.443	82.474	49.312
<b>Combiné</b>	<b>43.582</b>	<b>36.14</b>	<b>52.761</b>	<b>84.453</b>	<b>54.209</b>

Comme nous pouvons le constater dans le tableau, le score moyen pour HOTA est de 43.582, de 36.14 pour DetA, de 52.761 pour AssA, de 84.453 pour LocA et de 54.209 pour HOTA(0).

## 6. Discussion des résultats

En ce qui concerne la différence entre nos résultats de HOTA et HOTA(0), il faut d'abord comprendre le fonctionnement de ces mesures. La mesure de HOTA(0) ne prend pas en compte la localisation dans son calcul, car elle utilise une localisation fixe, tandis que HOTA fait une intégration sur l'ensemble des valeurs de HOTA(0) pour chaque seuil de localisation par bon de 0.05 (5%). Comme notre modèle prend en compte des seuils très bas pour le suivi d'objet, il performe ainsi nécessairement mieux dans un seuil plus bas. Également, HOTA prend en compte la précision temporelle du suivi des objets et la précision des identifiants des objets suivis.

Notre modèle contient tant des forces que des faiblesses. Avec les résultats que nous avons obtenus, on peut remarquer un score très faible pour la précision de détection (DetA) et un score moyen pour celui de la précision de l'association (AssA). De ce fait, bien que notre modèle ait une précision énorme pour la précision de la localisation des objets (LocA), celui-ci a beaucoup de difficulté à obtenir une confiance sur la détection de l'objet. De plus, l'association des objets entre les trames n'est pas la meilleure. On peut aussi le voir dans nos résultats avec les données sur les tasses. Plusieurs des tasses sont correctement identifiées, mais notre modèle arrive difficilement à faire une association entre des tasses déjà vues et des tasses qu'il vient de détecter, surtout quand celles-ci sortent du champ de vision de la caméra. Cependant, comme la précision de la localisation des objets est très élevée, cela nous permet d'avoir un modèle plutôt performant, sans être le meilleur.

En ce qui concerne les problèmes que nous avions identifiés, plusieurs ont pu être résolus par notre solution.

Le problème du changement de grandeur a été assez bien résolu. Bien que parfois notre modèle se trompait d'objet étant donné la forme changeante de l'objet avec sa grandeur, il arrivait assez bien à détecter l'objet et à effectuer son suivi. On peut justement le voir dans les images suivantes où il détecte bien le même objet tant rapproché qu'éloigné.

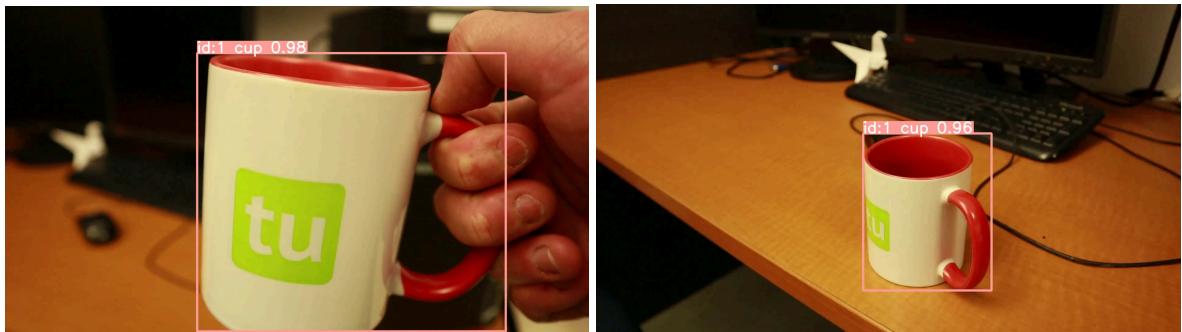


Figure 7. Exemple de suivi d'objet ayant un agrandissement/rétrécissement

Pour ce qui est du problème d'occlusion des tasses, notre modèle a aussi très bien performé. Bien qu'il a échoué parfois à identifier certaines tasses, il s'en est vraiment bien sorti dans la majorité des situations. On peut d'ailleurs le voir en action à un moment assez difficile étant donné qu'il y a plusieurs occlusions en même temps. Bien qu'il ait mal identifié la boîte de Tetley en arrière plan, l'ensemble des tasses a été identifié correctement. On peut ainsi considérer que le problème du scale est bel et bien résolu.

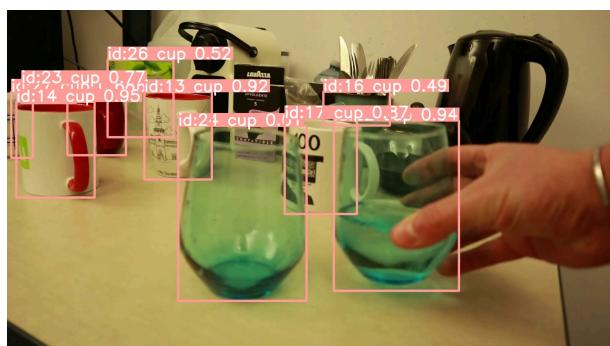


Figure 8. Exemple de suivi d'objet ayant beaucoup d'occlusion

D'un côté moins positif, notre modèle a complètement échoué sur la rotation des objets dans le plan. La majorité des trames où il y avait un objet avec une faible rotation, celui-ci n'était aucunement identifié par le modèle. Même une petite rotation rendait l'objet totalement invisible pour notre modèle. Ainsi, le problème de la rotation n'est vraiment pas résolu.



Figure 9. Exemple de suivi d'objet ayant une rotation

En ce qui concerne l'avant-dernière difficulté, qui était la présence de tasses floues, le modèle que nous avons choisi arrive bien à la surmonter, comme on peut le voir sur la figure ci-dessous.

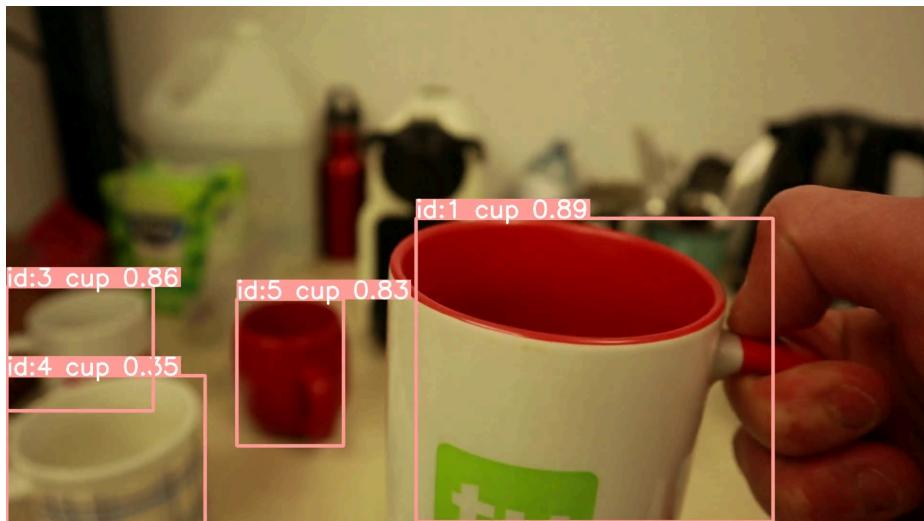


Figure 10. Exemple de suivi d'objet ayant du flou (blur)

Sur la figure 10, les tasses qui portent les identifiants 3, 4 et 5 sont floues, mais elles sont tout de même très bien détectées par le modèle.

Finalement, le dernier problème reposant sur les tasses qui sortent et entrent dans le cadre de la caméra n'est pas correctement pris en charge par notre modèle, comme le démontre la figure 11.



Figure 11. Exemple de suivi d'objet ayant sorti du champ de vision de la caméra

Dans la figure 11, sur l'image de gauche, la tasse a un id de 5. Quelques trames plus tard, elle n'est plus présente dans la vidéo et quelques trames après, elle revient dans la vidéo. On peut voir que lorsqu'elle revient, son id est de 12, ce qui est différent de son id initial.

En ce qui concerne nos attentes par rapport à la performance de notre modèle sur la séquence de Moodle, nous sommes plutôt confiants sur nos performances. Bien que l'identification entre les trames d'un même objet est très difficile pour notre modèle, la localisation des objets est très performante. Ainsi, nous nous attendons à un meilleur résultat que celui obtenu pour le jeu de données MOT17. Après l'étude de la majorité des trames, nous sommes très confiants de dire que nous pourrions facilement atteindre 60% de détection des objets.

## Références

- [1] OpenCV: <https://docs.opencv.org/4.x/index.html>
- [2] NumPy: <https://docs.scipy.org/doc/numpy/user/quickstart.html>
- [3] GitHub yolov9-tracking [https://github.com/xjl-le/yolov9\\_tracking?tab=readme-ov-file](https://github.com/xjl-le/yolov9_tracking?tab=readme-ov-file)
- [4] Résultat MOT17 <https://zenodo.org/records/7629840>
- [5] TrackEval <https://github.com/JonathonLuiten/TrackEval/tree/master/scripts>
- [6] ByteTrack : Multi-Object Tracking by Associating Every Detection Box  
<https://doi.org/10.48550/arXiv.2110.06864>
- [7] HOTA : A Higher Order Metric for Evaluating Multi-Object Tracking <https://arxiv.org/pdf/2009.07736.pdf>