26th International Conference on Pattern Recognition (ICPR), 2022
Pages 3522-3528
https://doi.org/10.1109/ICPR56361.2022.9956283
https://archimer.ifremer.fr/doc/00813/92503/

Fully Deep Simple Online Real-time Tracking: Efficient Re-Identification by Attention without Explicit Similarity Learning

Belmouhcine Abdelbadie [1, 2], Simon Julien [2], Courtrai Luc [1], Lefevre Sebastien [1]

[1] IRISA, Universite de Bretagne Sud, Vannes, France
[2] LTBH, Ifremer, Lorient, France

Abstract :

Most existing Multi-Object Tracking methods consider detection and re-identification as two distinct steps. As a result, the re-identification cannot leverage object location and is only based on appearance, thus leading to ID merges when dealing with highly similar objects. The few works that combine detection and re-identification still generate an appearance descriptor for similarity computation. However, since the detection task conflicts with the tracking task, the network privileges the former and generates similar descriptors for objects of the same class, especially when class instances have a strong visual similarity. Besides, when using a motion model or a motion prediction recurrent neural network to delimit the search area and overcome the problem of ID merges, the rise of uncertainty occurring when those models are not updated often leads to ID switches. In this paper, we tackle these issues and propose to use the same model for detection and re-identification by leveraging attention between features of two frames. By doing so, the network can make motion predictions without providing any appearance descriptor and without computing any learned similarity, thus eliminating the need for any motion prediction model and making the tracking trainable end-to-end. Our experimental results support our main contributions and show that our fully DeepSORT significantly reduces the number of ID switches and merges, even when using non-class-agnostic non-maximum suppression. Besides, our model is more resistant to variations in time lapses between two images, leading to improved tracking results.

Keywords : Resistance, Visualization, Uncertainty, Recurrent neural networks, Tracking, Computational modeling, Predictive models

1 Introduction

Le suivi d'objets multiples (SOM) est une tâche essentielle de la vision par ordinateur. Elle consiste à détecter les objets figurant sur chaque image d'une vidéo pour les associer par la suite à travers les différentes trames. Ainsi, on peut faire une distinction parmi les instances d'une même classe et conserver les identités des objets dans le temps. Dans le cas du SOM, on ne connaît pas au préalable le nombre d'éléments présents sur une vidéo et il n'y a aucun a priori sur leurs apparences. Le paradigme largement employé pour le SOM est le suivi par la détection. Tout d'abord, un détecteur est utilisé pour générer la géométrie des boîtes englobantes qui sont, par la suite, associées entre elles, à travers les différentes images de la vidéo. Cette association permet de créer/compléter des traces d'objets. La majorité des algorithmes de SOM par détection comprennent quatre étapes [1] : la détection, l'extraction de caractéristiques/la prédiction de mouvement, l'affinité et l'association. Notons que certaines de ces étapes peuvent être combinées dans la littérature.

Dans cet article, nous visons le suivi en temps réel des poissons de fonds marins. Le but est de créer des engins de pêche autonomes. Ainsi, à chaque instant t, nous ne connaissons que la trame courante I_t et les trames précédentes. Nous avons utilisé EfficientDet D1 comme détecteur, considérant qu'il fournit de bons résultats de détection dans la littérature. D'autre part, le temps de traitement diffère d'une image à l'autre en fonction de leur complexité intrinsèque. Ainsi, dans les applications temps réel, le temps écoulé entre deux trames successives peut varier en fonction du temps de traitement de chaque image, conduisant à de mauvais résultats de suivi lors de l'utilisation des méthodes classiques.

Nos contributions principales sont : i) la proposition d'un sous-réseau greffé à un EfficientDet déjà entraîné ; il utilise

les caractéristiques de la même échelle de deux images aux instants t et $t-\delta$ pour prédire le déplacement des objets de l'image $I_{t-\delta}$ à l'image I_t ; ii) l'utilisation de l'attention [2] entre les caractéristiques de deux images afin de les fusionner et prédire le déplacement des objets entre ces deux images sans utiliser un réseau tiers pour apprendre une similarité ; iii) une plus grande résistance aux variations de fréquences d'images.

L'article est organisé comme suit. Nous passons en revue les travaux antérieurs et soulignons nos contributions dans la Sec. 2. Nous expliquons ensuite la méthode proposée en Sec. 3, avant de présenter le protocole expérimental en Sec. 4. Nous concluons enfin l'article en Sec. 5.

2 État de l'art

La plupart des méthodes de suivi par détection utilisent deux modèles distincts, l'un pour la détection d'objets et l'autre pour l'extraction de caractéristiques en vue de leur réidentification. L'approche de suivi la plus simple mais la plus efficace est SORT [3], néanmoins elle souffre d'un nombre élevé de changements d'ID (ChID) provoqués par l'incertitude du filtre de Kalman, soit au début des trajectoires, soit lorsque le modèle de mouvement n'a pas été mis à jour depuis une longue période.

Afin de réduire le nombre de ChID, DeepSORT [4] étend SORT en introduisant à la fois le mouvement et l'apparence dans le processus de calcul d'affinité. Néanmoins, puisque la réidentification utilise seulement des extraits d'images correspondant aux objets pour produire la représentation utilisée dans le calcul de similarité, aucune information sur l'emplacement des objets n'est exploitée, conduisant à des fusions d'ID (FID) pour les objets similaires. Par conséquent, DeepSORT utilise toujours le filtre de Kalman pour limiter la zone d'association.

Dans les méthodes SORT et DeepSORT, le filtre de Kalman est utilisé pour la prédiction de mouvement. Ce modèle linéaire à vitesse constante estime le déplacement inter-images de chaque objet mais ne tient pas compte du mouvement de la caméra, de la variation de la fréquence d'images et de l'apparence des objets. En outre, il suppose que la vitesse est constante et que la distribution des positions des objets dans une même trajectoire est gaussienne. Par conséquent, le filtre de Kalman devient limité lorsque la caméra et les objets bougent en même temps ou lorsque la vitesse n'est pas constante et change soudainement. Ce filtre fait aussi généralement des prédictions inexactes au début d'une trajectoire lorsque son incertitude est encore très élevée, ce qui cause plusieurs ChID. Ainsi, l'intégration de l'estimation de mouvement dans le réseau de détection contribuerait à réduire davantage les ChID.

De nombreuses approches combinent détection et suivi. Par exemple, Tracktor [5] adapte Faster RCNN pour estimer l'emplacement des boîtes englobantes dans la nouvelle image à partir de la précédente, tout en restant limité aux cas avec un faible mouvement inter-images. RetinaTrack [6] extrait un vecteur de caractéristiques pour chaque ancre. Il utilise une correspondance bipartite gloutonne au moment de l'inférence pour associer les trajectoires. Cette association est basée sur une représentation apprise par un réseau de réidentification partageant une partie avec le réseau de détection. Cependant, la mesure d'intersection sur union (IsU) utilisée par RetinaTrack n'est pas utile si les trames ne sont pas consécutives et si le mouvement inter-trames n'est pas suffisamment faible. Ainsi, après ne pas avoir mis à jour une trajectoire pendant un certain nombre d'images, la position n'est pas exploitée et l'association se fait exclusivement en fonction de l'apparence, ce qui peut entraîner des fusions et des changements d'ID lorsque les objets ont une apparence similaire comme dans le cas des poissons. De plus, RetinaTrack est inadapté lorsque la fréquence d'images est faible, conduisant à une FID.

Afin de résoudre tous ces problèmes, nous avons combiné la prédiction de mouvement et l'apprentissage de la similarité d'apparence en une seule étape. Ainsi, le réseau est en mesure de réidentifier un objet entre deux images en n'apprenant aucune similarité d'apparence explicitement. En effet, la similarité d'apparence est utilisée implicitement pour prédire le déplacement de l'objet.

3 Méthode

Nous proposons une méthode nommée Fully DeepSORT où l'on intègre un sous-réseau de prédiction de mouvement à un réseau de détection mono-étape à base d'ancres. À partir des caractéristiques extraites de deux images I_t et $I_{t-\delta}$ par EfficientDet on prédit pour chaque objet détecté dans $I_{t-\delta}$ sa position dans I_t.

3.1 Détection

Nous utilisons EfficientDet D1 [7] comme détecteur car il est largement utilisé et montre de bonnes performances

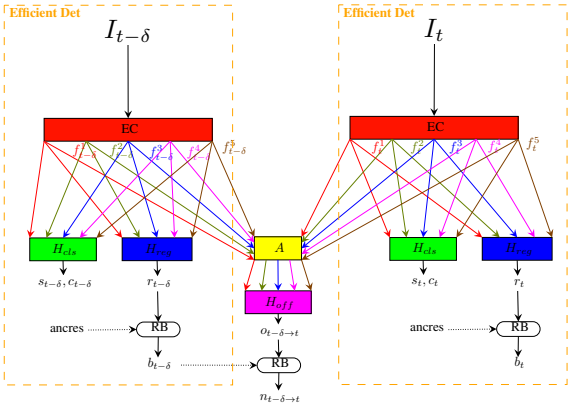

Figure 1 – *L'architecture du modèle proposé. La même couleur signifie des poids partagés. Les cadres en pointillés représentent le même EfficientDet appliqué à deux images différentes. Les détails du bloc A sont mis en évidence dans la Figure 2.*

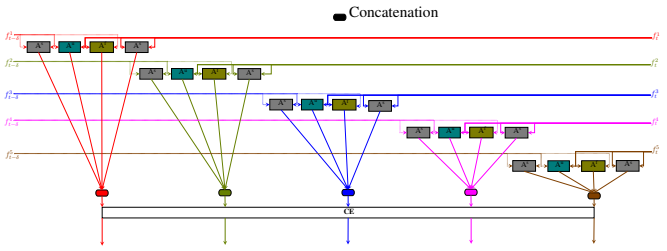

Figure 2 – *L'architecture du bloc A de la Figure 1. Une même couleur signifie des poids partagés.*

dans la littérature. Tout d'abord, il extrait cinq cartes de caractéristiques à résolutions spatiales différentes. Soit $f_t = EC(I_t) = \{f_t^k\}_{k=1...5}$ l'ensemble des cartes de caractéristiques extraites par EfficientDet. Ces cartes passent par deux sous-réseaux: la tête de classification et la tête de régression. A chaque position sur les cartes de caractéristiques, la tête de régression produit un décalage r_t à partir de chaque ancre, et la tête de classification prédit la classe associée à chaque ancre. Soit K l'ensemble des ancres, H_{reg} la fonction de régression, H_{cls} la fonction de classification et RB une fonction qui utilise les décalages et les ancres pour produire des boîtes englobantes. La sortie du réseau à partir d'une image I_t est un ensemble de boîtes englobantes $b_t = RB(H_{reg}(f_t), K)$ et un ensemble de classes correspondant à ces b_t avec leurs scores $c_t, s_t = H_{cls}(f_t)$. La Figure 1 montre le processus de détection dans EfficientDet D1 (dans le cadre orange en pointillés) appliqué à deux images I_t et $I_{t-\delta}$.

3.2 Prédiction de mouvement et attribution des ID

Au lieu d'utiliser un filtre de Kalman ou un réseau récurent pour la prédiction de mouvement, nous greffons un sous-réseau de déplacement au réseau de détection. Ce sous-réseau prend en entrée les cartes $f_{t-\delta}$ et f_t correspondant respectivement à deux images $I_{t-\delta}$ et I_t. Ensuite, pour chaque couple de cartes f_t^k et $f_{t-\delta}^k$ de même résolution spatiale, on calcule une attention [2] vers l'avant, une attention vers l'arrière et deux auto-attentions [8] (une pour chaque image) entre ces cartes. Le résultat consiste en quatre cartes de caractéristiques d'attention. Soit A^f (resp. A^b, A^s) la fonction d'attention vers l'avant (resp. vers l'arrière, vers soi-même). Nous avons une fonction d'attention distincte pour chaque sens (arrière, avant et soi-même) et ces fonctions sont partagées à travers les résolutions spatiales. La $k^{ème}$ carte de caractéristiques d'attention finale $A(f_{t-\delta}^k, f_t^k)$ est le résultat de la concaténation des $k^{èmes}$ attentions vers l'arrière et vers l'avant et les deux cartes de caractéristiques d'auto-attention, passées à travers un réseau de compression et d'excitation (CE) [9] : $A(f_{t-\delta}^k, f_t^k) = CE([A^s(f_{t-\delta}^k), A^b(f_{t-\delta}^k, f_t^k), A^f(f_t^k, f_{t-\delta}^k), A^s(f_t^k)])$. Après avoir calculé les cinq cartes de caractéristiques d'attention correspondant aux cinq résolutions d'EfficientDet, on applique le sous-réseau de déplacement sur ces cartes afin d'obtenir un déplacement des objets détectés sur l'image $I_{t-\delta}$, à partir de $I_{t-\delta}$ vers I_t. Soit $o_{t-\delta\to t} = H_{off}(\{A(f_{t-\delta}^k, f_t^k)\}_{k=1...5})$, la position sur I_t des objets appartenant à $I_{t-\delta}$ est estimée par $n_{t-\delta\to t} = RB(o_{t-\delta\to t}, b_{t-\delta})$. La Figure 1 illustre le processus de prédiction de mouvement à partir de l'image $I_{t-\delta}$ vers l'image I_t, tandis que la Figure 2 met en évidence la fonction A.

Disposant de boîtes b_t d'une image I_t, des boîtes $b_{t-\delta}$ de $I_{t-\delta}$ et des positions estimées $n_{t-\delta\to t}$ de $b_{t-\delta}$ dans I_t, on calcule la distance d'IsU d_{IsU} entre $n_{t-\delta\to t}$ et b_t et on considère l'association entre les boîtes de $I_{t-\delta}$ et celles de I_t si la distance IsU est inférieure à α. Ainsi chaque élément dans b_t peut être associé à plusieurs éléments dans $b_{t-\delta}$.

Pour attribuer des identifiants aux $b_t \in I_t$, on définit des associations entre b_t et $\{b_{t-\delta}\}_{\delta=1...T_{Lost}}$. T_{Lost} est le nombre d'absences consécutives pour considérer qu'un objet est perdu. Ensuite, on calcule un coût d'association entre chaque boîte englobante et les IDs par le biais d'un vote pondéré. Pour chaque objet, chaque image $I_{t-\delta}$ vote positivement avec un poids de $2^{1-\delta}$ pour les ID ayant $d_{IsU} \leq \alpha$ avec l'objet cible, et négativement avec un poids de $-2^{1-\delta}$ pour les ID ayant $d_{IsU} > \alpha$ avec l'objet cible. L'association finale est faite en utilisant l'algorithme hongrois. Toutes les associations entre objets de classes différentes sont ignorées. Tous les éléments de b_t qui n'ont pas eu un ID en reçoivent un nouveau.

4 Expériences

4.1 Protocole expérimental

Nous avons évalué notre approche à l'aide de données de détection et de suivi des espèces du fond marin. Des experts de l'IFREMER ont constitué et annoté un jeu de données, contenant 5 catégories; *Nephrops norvegicus*, *Pennatulacea*, *Actiniaria*, *Munida* et *Actinopterygii*. En effet, une caméra a été placée à l'entrée d'un traîneau de pêche de fond et a enregistré des vidéos du fond marin. Nous avons 48 vidéos; 43 utilisées pour l'entraînement (13 074 images, 16 264 objets et 960 identités), une pour la validation (7 597 images, 586 objets, 32 identités) et quatre pour le test (31 674 images, 8 930 objets, 508 identités).

Afin d'évaluer les performances de suivi, nous avons utilisé les métriques d'évaluation les plus populaires, à savoir : les métriques CLEAR MOT [10] et les métriques d'ID [11].

4.2 Résultats et discussions

D'après la Table 1, un suivi simple basé sur l'IsU fait beaucoup de ChID car il rate très souvent des associations. En effet, dans certains cas, les objets de notre jeu de données peuvent se déplacer rapidement et la fréquence de capture n'est pas très élevée. Notre approche améliore toutes les métriques de suivi à l'exception du MOTP et FM. Le MOTP est lié davantage à la détection qu'au suivi. Le score FM s'explique par des ChID réduits et des ID conservés lors des absences des objets sur certaines images. Notre méthode réduit énormément les ChID,

Table 1 – Performances du suivi.

Approche	MOTA ↑	MOTP ↑	ChID ↓	FID ↓	FM ↓	IDVP ↑	IDFP ↓	IDFN ↓	IDP ↑	IDR ↑	IDF1 ↑
IsU	0,419	0,781	446	27	662	4333	793	5854	0,845	0,425	0,566
SORT	0,436	0,781	271	4	633	4633	496	5557	0,903	0,455	0,605
DeepSORT	0,461	0,781	20	0	**585**	4870	259	5320	0,95	0,478	0,636
RetinaTrack	0,355	**0,803**	47	117	2442	3274	732	6916	0,817	0,321	0,461
Fully DeepSORT	**0,462**	0,781	**4**	**0**	662	**4901**	**225**	**5286**	**0,956**	**0,481**	**0,64**

Table 2 – Performances du suivi en ignorant l'une de deux images consécutives avec une probabilité de 0,5.

Approche	MOTA ↑	MOTP ↑	ChID ↓	FID ↓	FM ↓	IDVP ↑	IDFP ↓	IDFN ↓	IDP ↑	IDR ↑	IDF1 ↑
IsU	0,381	**0,781**	574	36	436	2842	955	4752	0,748	0,374	0,499
DeepSORT	0,444	**0,781**	124	7	**384**	3475	366	4177	0,905	0,454	0,605
Fully DeepSORT	**0,455**	**0,781**	**17**	**2**	468	**3601**	**196**	**3993**	**0,948**	**0,474**	**0,632**

ce qui montre une réidentification réussie. De plus, nous pouvons voir que les performances de suivi de RetinaTrack sont médiocres car notre jeu de données contient des instances très similaires.

Lors du suivi en temps réel, la fréquence d'images peut varier car le temps d'inférence nécessaire pour le détecteur dépend de la complexité intrinsèque de l'image. Ceci rend le suivi basé sur le filtre de Kalman ou l'IsU limité, puisqu'il nécessite dans le premier cas des déplacements constants et dans le deuxième cas des déplacements faibles. Cependant, puisque notre méthode ne repose pas sur la géométrie des boîtes englobantes et utilise implicitement l'apparence, elle conduit à de meilleurs résultats dans ces circonstances. Nous l'avons vérifié expérimentalement en sautant une des deux images consécutives avec une probabilité de $0,5$ pour simuler la variation de la fréquence d'images. La Table 2 montrent les résultats obtenus par DeepSORT, l'IsU et Fully DeepSORT dans ce scénario. Nous pouvons observer que la variation des laps de temps entre les images déstabilise DeepSORT et l'IsU mais pas notre méthode, qui conserve des performances similaires. Par contre, puisque notre approche repose sur l'attention entre les caractéristiques de la même échelle, elle peut devenir défectueuse et entraîner des ChID si un objet change d'échelle entre deux images (notamment lorsqu'il se rapproche ou s'éloigne de la caméra).

5 Conclusion

Dans cet article, nous avons proposé Fully Deep SORT. Cette approche ne fait pas d'apprentissage de similarité et n'applique pas un filtre de Kalman pour la prédiction de mouvement. Elle exploite l'attention entre les caractéristiques de deux images pour prédire le déplacement des objets de l'une à l'autre. Nous avons montré que notre méthode améliore les performances du suivi par rapport à DeepSORT basé sur EfficientDet D1 et RetinaTrack. Elle réduit les ChID et les FID, qui sont les principaux inconvénients rencontrés dans l'état de l'art. Fully Deep SORT est aussi plus résistante aux variations de laps de temps entre les trames. Mais, comme nous n'utilisons que des caractéristiques de la même échelle pour calculer l'attention entre deux images, la réidentification devient déficiente lorsque les objets changent d'échelle entre les images. Corriger ce défaut est la principale piste pour poursuivre ce travail.

References

[1] G. Ciaparrone, F. L. Sánchez, S. Tabik, L. Troiano, R. Tagliaferri, and F. Herrera, "Deep learning in video multi-object tracking: A survey," *Neurocomputing*, vol. 381, pp. 61–88, 2020.

[2] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, 2018, pp. 7794–7803.

[3] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *ICIP*, 2016, pp. 3464–3468.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *ICIP*, 2017, pp. 3645–3649.

[5] P. Bergmann, T. Meinhardt, and L. Leal-Taixe, "Tracking without bells and whistles," in *ICCV*, 2019, pp. 941–951.

[6] Z. Lu, V. Rathod, R. Votel, and J. Huang, "Retinatrack: Online single stage joint detection and tracking," in *CVPR*, 2020, pp. 14 668–14 678.

[7] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *CVPR*, 2020, pp. 10 781–10 790.

[8] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," in *International conference on machine learning*. PMLR, 2019, pp. 7354–7363.

[9] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *CVPR*, 2018, pp. 7132–7141.

[10] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[11] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *ECCV*, 2016, pp. 17–35.