Proceedings Of The 12th International Symposium On Image And Signal Processing And Analysis (ispa 2021)
2021, Pages 138-144
https://doi.org/10.1109/ISPA52656.2021.9552062
https://archimer.ifremer.fr/doc/00728/83972/

Robust Deep Simple Online Real-Time Tracking

Belmouhcine Abdelbadie [1,*], Simon Julien [2], Courtrai Luc [3], Lefevre Sebastien [3]

[1] LTBH - IRISA IFREMER - Université de Bretagne Sud, Vannes, France
[2] LTBH IFREMER, Lorient, France
[3] IRISA Université de Bretagne Sud, Vannes, France

* Corresponding author : Abdelbadie Belmouhcine, email address : abdelbadie.belmouhcine@ifremer.fr

Abstract :

Simple Online and Real-time Tracking (SORT) and its deep extension (DeepSORT) are simple, fast, and effective multi-object tracking by detection frameworks. Their main strengths are simplicity and speed. However, they still suffer from some problems, such as identity switch, instance merge, and many false positives, which prevent the tracking results from being used for subsequent tasks such as counting. In this paper, we strengthen and improve the tracking using EfficientDet and DeepSORT. In our approach, the motion prediction uses appearance, and the appearance embedding uses location. First, we modify the deep detection network to predict the objects' motion in the next frame by leveraging the attention between the current image and the next image. Second, an appearance-based metric is used to associate detection to tracks after false negatives and occlusion. This metric is a learned Mahalanobis distance between two feature descriptors constructed using EfficientDet and attention given to regions of interest from their images. Finally, we count only high confidence tracks having a minimum frequency of apparition. Our approach has been applied to a challenging real-life problem, namely seabed species tracking and counting. Our experimental results show that Robust DeepSORT reduces identity switches and merges. Thus, it improves tracking and counting evaluation measures while keeping the simplicity of the origlnal DeepSORT.

Keywords : Counting, DeepSORT, EfficientDet, MultiObject Tracking, SORT

I. Introduction

Multiple object tracking (MOT) is a central problem in computer vision. It identifies and distinguishes every object in a frame and tracks them until they leave the scene. In addition to object detection, which draws rectangular bounding boxes around objects and indicates their classes, MOT algorithms assign an ID to each box to distinguish the same class instances. MOT can be treated as a data association problem since the goal is to associate detections across multiple frames. Indeed, MOT is typically composed of two steps: detection, i.e., locating objects in the image, and data association, i.e., linking detections across frames to constitute complete trajectories. There are two types of tracking paradigms: online and batched. In this work, our interest is in online tracking, which looks only at the current frame and previous ones, thus targetting real-time tracking. Note that in real-time tracking, we need to make a trade-off between running time and performance.

This work was done as a part of the Game of Trawls project. We thank the European Maritime and Fisheries Fund (contract number 18/2216442) and France Filière Pêche (contract number 19/1000544) for funding.

Simple Online and Real-time Tracking (SORT) [1] is a simple tracking approach that applies Kalman filtering [2] for object motion estimation and the Hungarian algorithm [3], with an association metric that measures bounding box overlap, i.e., Intersection over Union (IoU), for object association. While simple, SORT, with a robust detector, provided good performances at high frame rates [4] on the MOT challenge dataset [5]. Although SORT achieves an overall good tracking precision and accuracy, it suffers from a relatively high number of identity switches [4] due to the low accuracy of the Kalman filter when the uncertainty is high. Therefore, when detection is missing due to occlusion or a false negative, the algorithm becomes defective. In addition, it can also be defective at the beginning of tracks. DeepSORT [4] reduced identity switches by applying an appearance-based metric computed from bounding box embeddings obtained using a third-party Siamese network trained for similarity learning. Even though this extension allows SORT to continue associations through occlusion and false negatives, it does not improve associations at the beginning of trajectories. Improving associations at the beginning can further reduce identity switches.

In this paper, we propose a novel tracking and counting method inspired by the SORT workflow: 1) We use a convolutional neural network for object detection and modify the network to predict the object's motion in the next frame by leveraging the attention mechanism [6], [7]. 2) We use an appearance-based Mahalanobis distance [8] to support the Kalman filter in detection to track connection after missing detections and occlusion. 3) The Hungarian algorithm performs data association to complete the tracks. 4) We use high confidence tracks with a minimum occurrence frequency to count the targets.

Our main contributions are: 1) The use of appearance implicitly in motion prediction by 2) the incorporation of motion prediction in EfficientDet [9]. We use attention between two images to make a more robust detection and motion prediction. We also 3) extract feature descriptors for bounding boxes explicitly and directly from EfficientDet to make associations through occlusion and missing detections by leveraging attention given to regions of interest from their corresponding images. This way, the location is implicitly incorporated when computing object descriptors.

We aim to reduce the use of the Kalman filter to make

SORT stronger. DeepSORT integrates a deep network to learn a similarity metric. However, it relies only on Kalman filtering for motion prediction and does not use deep learning in that part. The Kalman filter is known to be inaccurate when the uncertainty is high, either at the beginning of a track or when detections are missing due to occlusion or false negatives. So, we propose to integrate frame to (following) frame motion prediction in the EfficientDet [9] detector to have a more accurate position in the next frame.

MOT has many real-world applications. In this work, we identify and count seabed species in real-time, using a camera placed in front of a sledge to catch only desired varieties.

Our paper is organized as follows; We review related work and emphasize our contributions in Sec. II. We then detail our proposal in Sec. III, before providing the experimental validation in Sec. IV. We finally conclude the paper in Sec. V.

II. Related Work

Multi-object tracking approaches can be divided into detection-based-tracking (DBT) and detection-free-tracking (DFT) based on initialization methods [10]. DFT methods require targets to be manually identified and then track them in subsequent frames [11], [12]. However, in most MOT real-time applications, we do not have prior knowledge of targets. DBT methods build tracks by detecting objects in each frame and completing the tracks as the video is playing [1], [4]. Hence, DBT methods are more suitable for MOT. According to the processing mode, MOT methods can be divided into online and offline approaches. Online methods only use the current and previous frames [1], [4] and are thus, more suitable for real-time applications.

SORT [1] relies on two classical but efficient methods: Kalman filtering [2] for motion prediction and the Hungarian algorithm [3] for data association. First, a detector is used to provide bounding boxes for each object present in the frame. Then, the inter-frame movement of each object is approximated by a linear constant velocity model. Whenever a detection is associated with a track, the detection is used to update the track state by optimally solving the velocity using the Kalman filter. The association between detections and tracks is done by estimating the target bounding box geometry in the current frame using the motion model. A cost matrix is constructed by computing the intersection over union (IoU) between every new detection and estimated bounding boxes of all existing tracks. Based on this cost matrix, the association is done via the Hungarian algorithm. Associations having an overlap less than IoU_{min} are rejected. Tracks live during a particular time and are deleted if they are not associated with any detection for T_{Lost} frames. Removing tracks after a specific time is helpful because as a track stays without any update, the Kalman filter's uncertainty rises.

SORT suffers from the problem of a high number of identity switches. An identity switch happens when a ground truth trajectory is covered by multiple predicted tracks, which means that the ground truth changes IDs during the tracking process. For this reason, an extension of SORT, called DeepSORT [4], integrates both motion and appearance information into the association problem. Object appearance has a vital role in track-detection association. It helps to reduce ID switches. DeepSORT extracts crops of images corresponding to detections and trains a Siamese network to compute an appearance descriptor r. Furthermore, for each track k, DeepSORT keeps the last L_k associated appearance descriptors to compute the cosine similarity between detections and tracks in the appearance space. Besides, the motion model is also essential since it reduces the search area. In DeepSORT, the authors used Mahalanobis distance between the predicted Kalman filter state and detections; this metric is thresholded at 95% confidence interval computed from the inverse χ^2 distribution. However, DeepSORT cannot eliminate ID switches at the beginning of tracks when the association is still done exclusively based on the Kalman filter's estimation.

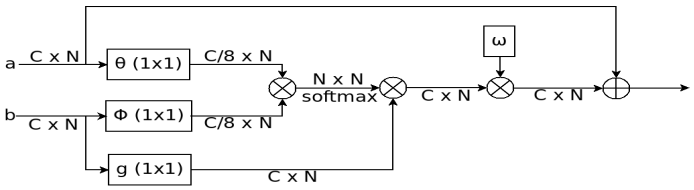

Fig. 1: The architecture of a Non-Local Block.

In SORT and DeepSORT methods, the Kalman filter is used for motion prediction and propagates a target identity into the next frame. Indeed, a linear constant velocity model approximates the inter-frame displacement of each object. This model does not consider the camera motion and object appearance and supposes that the velocity is constant and the distribution of object positions in the same trajectory is Gaussian. Thus, the Kalman filter becomes weak when both the camera and objects move or when the velocity is not constant and changes suddenly. Kalman filter can also make wrong predictions when the object makes an unusual movement. Moreover, the Kalman filter usually makes inaccurate predictions at the beginning of a track when its uncertainty is still very high, leading to many identity switches. For this reason, SORT and DeepSORT give a high number of identity switches. Hence, reducing uncertainty at the beginning of a track can further reduce the number of identity switches.

III. Proposed method

When applying only the Kalman filter, objects' appearance is not considered, making association inaccurate in occlusion cases. DeepSORT solves this by incorporating an appearance-based similarity learned using a Siamese network [13]. However, the appearance metric is not accurate at the beginning of the track since it does not contain enough appearance information, and the initialization of a track can be wrong. Thus, we propose extending DeepSORT to an end-to-end deep learning solution, including appearance description and motion prediction, and replacing the Kalman filter with the network prediction when possible. Furthermore, we make location prediction use appearance implicitly since it is generated

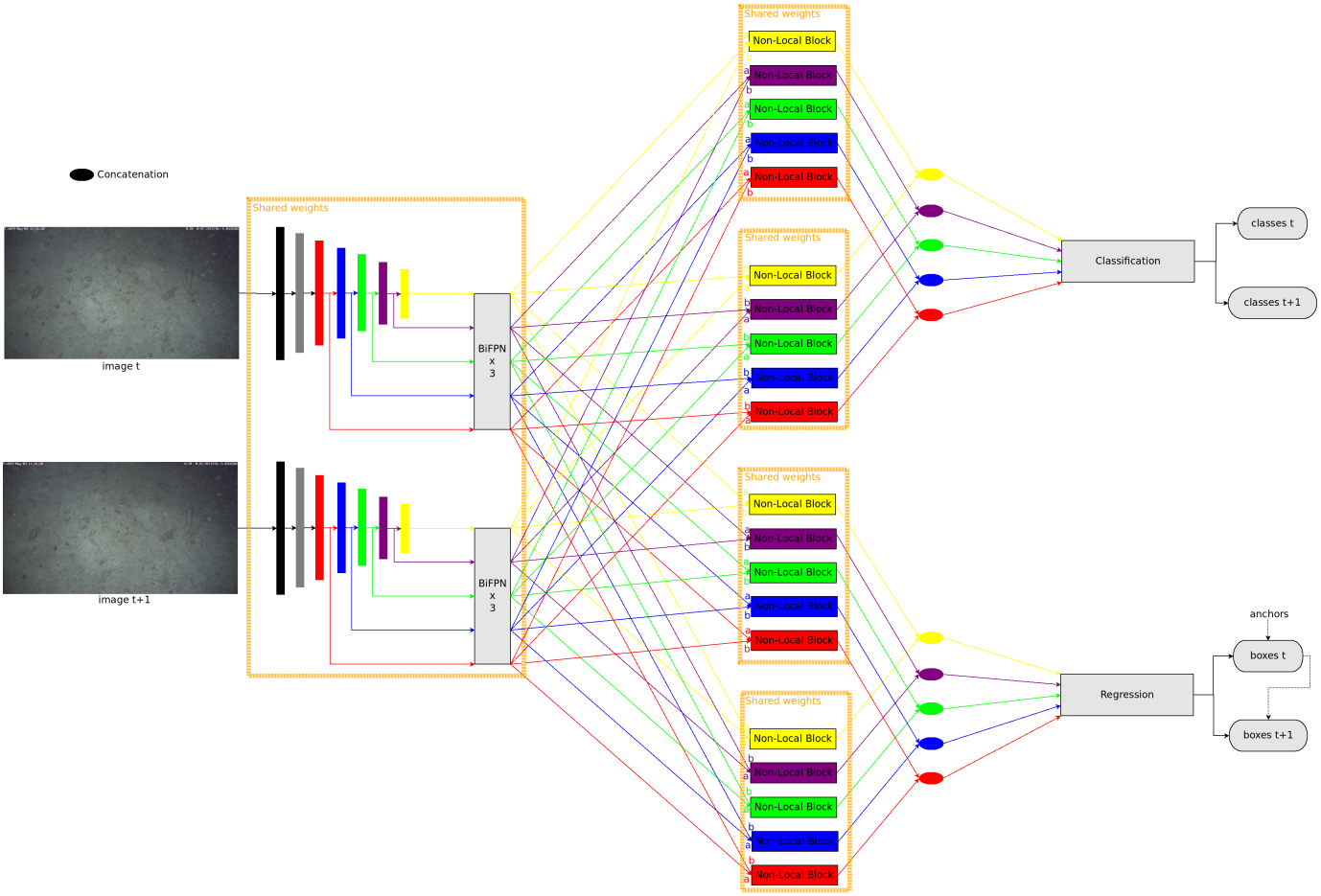

Fig. 2: The architecture of the modified EfficientDet (EfficientTrack).

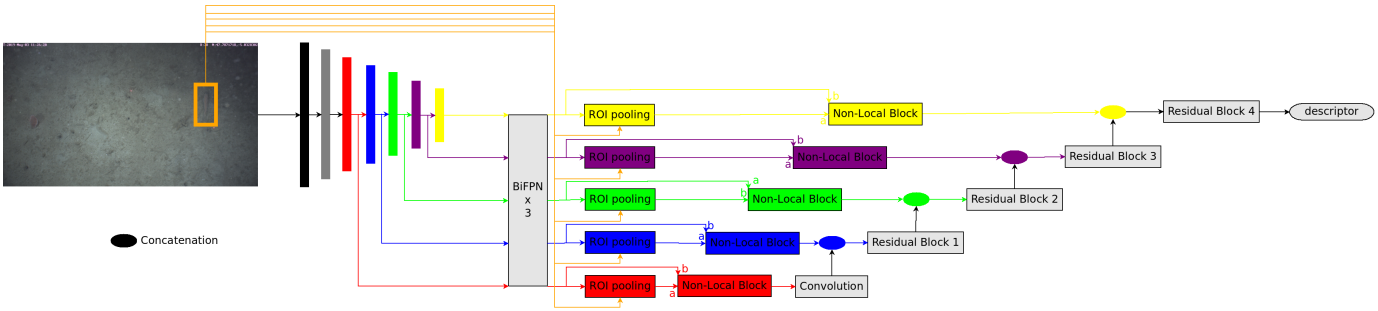

Fig. 3: EfficientDet based feature representation of bounding boxes.

using the detector's feature space. Also, we make appearance description use location implicitly by incorporating long-range dependencies between pixels of two successive frames.

A. Detection

For detection, we used EfficientDet [9], which is a strong high performing object detection network. It is an anchor-based detector that is similar to RetinaNet [14], except that it uses EfficientNet [15] as a backbone instead of ResNet [16] and Bidirectional Feature Pyramid Network (BiFPN) rather than Feature Pyramid Network (FPN) [17]. From the pyramid of features, five feature maps with different spatial resolutions are extracted, as shown in Fig. 2. Each feature map is passed through two task-specific networks: a classification head and a regression head. The classification head predicts each anchor's class, and the regression head outputs an offset from each anchor.

EfficientDet uses RetinaNet's focal loss to overcome the problem of imbalance between foreground and background. This loss uses a parameter γ to exponentially weight easy examples, which have high classification probability, with a smaller weight, to allow the optimization process to focus on

more challenging examples having low classification probabilities.

We modified EfficientDet to make object detection on the target image, leverage features, and, more specifically, attention received from the next frame. We call our modified model: EfficientTrack. We explain it in the following subsection (III-B).

B. Motion prediction

To make a more accurate motion prediction, we modified EfficientDet to take, as input, two successive images. Those two images pass through the same backbone and BiFPN, and each gives five feature maps of different spatial resolution. To fuse each pair of feature maps of the same size, belonging to adjacent frames t and $t+1$, we follow the idea of Non-Local Blocks [6]. Let x_t be a feature map of image t and x_{t+1} be a feature map of image $t+1$. Both have a size of $C \times N$, where N is the resolution and C is the depth. We compute two feature representations $y_{t|t+1} = attn_1(x_t, x_{t+1})$ and $y_{t+1|t} = attn_2(x_{t+1}, x_t)$, that model the relationship between the two feature maps x_t and x_{t+1}, called attention feature maps [7]. $y_{t|t+1}$ is the attention feature map of x_t based on x_{t+1} and $y_{t+1|t}$ is the opposite. Indeed, as shown in Fig. 1, to compute the attention feature map of an input a based on an input b, we first compute an affinity between each position i in a and each position j in b using a function $f(a_i, b_j) =< \theta(a_i), \phi(b_j) >$, where $\theta : \mathbb{R}^C \rightarrow \mathbb{R}^{\frac{C}{8}}$ and $\phi : \mathbb{R}^C \rightarrow \mathbb{R}^{\frac{C}{8}}$ perform a 1×1 convolution. Then, we use an unary function $g : \mathbb{R}^C \rightarrow \mathbb{R}^C$ that computes a representation for the input at position j using a 1×1 convolution. The attention feature map of a based on b ($attn(a, b)$) computes each position i of a using all positions j of b such as:

$$attn(a_i, b) = \frac{1}{C(a_i)} \sum_{\forall j} f(a_i, b_j) g(b_j) \tag{1}$$

with $C(a_i)$ a normalization constant.

The Non-Local Block for an input a based on attention given by an input b is defined as:

$$Z(a_i) = a_i + \omega \times attn(a_i, b) \tag{2}$$

As stated in [6], [7], we initialize the learned parameter ω to 0 to not break the initial behavior of the network, which initially does not contain Non-Local Blocks. By training, the network will figure out how much importance should be given to attention feature maps.

For each head, we use two Non-Local Blocks, one for each image. Then outputs of the same resolution are concatenated depth-wise and passed to the classification head, which outputs a class vector for each anchor according to the first image, and another class vector for each anchor according to the second image. Concatenated feature maps are also passed to the regression head, which outputs now boxes coordinates in the first image (w.r.t. anchors) and box positions in the second image (w.r.t. boxes of the first image). Fig. 2 shows the modified EfficientDet architecture, which we call EfficientTrack. Indeed, the network uses the object's position in the current frame to predict its position in the next frame and implicitly uses the object's appearance. It allows the motion prediction to be more accurate at the beginning of tracks than when using only the Kalman filter, whose uncertainty is very high at the beginning of trajectories. However, when detections are missing due to occlusion or false negatives, the Kalman filtering will still be needed. If the object does not reappear for a long time, the uncertainty will grow, leading to many ID switches. Indeed, as the absence last, the uncertainty grows with it. The association between predicted location and detection is done using Intersection over Union (IoU) [18] and is only accepted if the overlap is at least t_1. Also, the connection is not made when a candidate object for an association to a track has a different category than the track's one.

C. Appearance explicit description

Since the network only predicts frame to following frame motion, we use the Kalman filter to carry on motion prediction when the network misses a detection or when occlusion occurs. However, as evoked before, the Kalman filter's uncertainty grows as the track is not updated. To this end, we employ a CNN similar to the one applied in DeepSORT, which was used for person re-identification [19]. We use it for deep Mahalanobis distance [8] learning. We chose Mahalanobis distance because it does not have negative values as cosine distance and considers the correlations between vectors components. This distance is used for association only when no detection is associated with the track in the previous frame. In that case, the motion prediction is carried on by the Kalman filter, and we use a radius r to limit the association area. The difference with the original DeepSORT model is that a) we reused features extracted from EfficientDet, rather than a resized crop of an image, and b) we employed Mahalanobis distance rather than cosine similarity. Mahalanobis distance between two descriptors d_i and d_j is given as follows:

$$D(d_i, d_j) = \sqrt{(d_i - d_j)^T S (d_i - d_j)} \tag{3}$$

The matrix S is learned while training the network and initialized by the identity matrix so that the training begins by using the Euclidean distance. The network then learns the correlation between elements of the descriptors.

Siamese networks trained with cropped images of objects provide a location-independent appearance description of those objects. So, when we crop an object, we lose location information. Thus, to get the object location involved in feature extraction, we use a Non-Local Block to compute attention maps of pooled regions of interest, using the whole image characteristics. For example, let us take the two objects in Fig. 4. The Siamese network taking cropped images provides similar representations for two similar objects even though they are distinct (cosine similarity is 0.993526). However, if we use the whole image to compute a feature representation using our proposed approach, the representation will implicitly use information about the object's location. As a result, the

Mahalanobis distance between the two similar but different instances is 1.1403774.

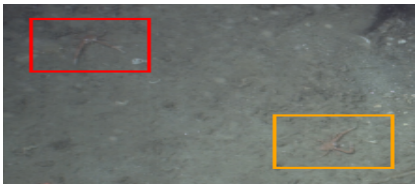

Fig. 4: Two distinct instances, having similar appearance.

When we want to extract features representing an object in an image, we first perform a Region of Interest (ROI) [20] pooling between each of the five feature maps and the object's bounding box. Then we compute attention ROIs, using attention given by the whole image to those ROIs. The model will use long-range dependencies and leverage information about object locations. Resulted attention ROIs are fed to a description network in a cascade way. This network is a wide residual network [16] with two convolutional layers followed by four residual blocks. First, we feed the first feature map to the convolutional layers. Then the output is concatenated channel-wise with the second feature map and fed to the first residual block. We continue the same way with the three remaining feature maps and residual blocks. Fig. 3 illustrates the box feature extraction part. The association between tracks and new detections is done by calculating the mean of Mahalanobis distance between descriptors of detections covered by the track in the past s frames and the new detection. Note that the association is permissible only if the Mahalanobis distance value is at least t_2 and the object to be associated with a track has the same category as the track.

D. Object counting

We propose a counting procedure based on tracking. First, we keep only detections having a score greater than c_1. Then, an object is counted when it has an associated track and appears in the track at least m times with a detection score greater than c_2. Thus, objects are only counted once in case of false positives.

IV. Experimental evaluation

A. Experimental setup

We used a dataset of seabed species. The challenge with that dataset is that individuals may have similar appearances in the same video and the same frame. Moreover, objects appear and leave at any frame, not only on the first and last frame. Therefore, to evaluate our model, we need a dataset with a sufficient number of images containing ground truth object ids, multiple categories, and high visual similarity between objects of the same class.

We used images captured by a front-down camera placed at the entrance of a sledge to evaluate the detection, tracking, and counting performance of the proposed algorithm. As a detection and tracking dataset, our data is composed of 46 videos issued from a French campaign conducted by IFREMER[1] in 2019, and two other different videos issued from an Irish campaign. The frame rate of video recordings is 12 fps, and image resolution is 2048×1152. First, we used 43/46 videos for training, which gives 13,074 annotated images, including 16,264 bounding boxes with 960 individual identities. Second, we validated the approach with one video containing 7,597 images with 586 objects and 32 individuals. Finally, we utilized the two remaining videos and the Irish videos for testing, which gives us 31,674 images containing 8,930 bounding boxes and 458 instances. We used Irish data only in tests to objectively evaluate the tracking and counting system and assess its generalization ability since videos were recorded in a different place, with different lighting conditions, and by different persons. All the data were annotated manually by experts in IFREMER to allow the evaluation of our contributions. The dataset contains 5 classes with enough data; Nephrops norvegicus, Pennatulacea, Actiniaria, Munida, and Actinopterygii.

To improve the algorithm's generalization ability, we applied several data augmentations during the training, i.e., random gamma correction, random HSV adjustment, random JPG compression rate, the use of random brightness and contrast, random Gaussian noise addition, random cut off, random vertical and horizontal flips, and random affine transformation. Moreover, we used random mosaic image combinations for training the detector.

To evaluate the tracking performance, we used the most popular evaluation metrics : Multi-object tracking accuracy (MOTA), Multi-object tracking precision (MOTP), Identity switches (IDSW), Merge, Fragmentation (FM) and ID metrics [21].

The counting algorithm is evaluated using precision (P), recall (R), F_1-Score, and mAP@0.5. Note that, as we have multiple categories, we used two types of F_1 averaging; micro-F_1 computed using global class agnostic precision and recall, and macro-F_1 calculated by averaging F_1 of different categories.

An EfficientDet [9] D1, pre-trained on COCO [22], was fine-tuned on our dataset in two phases for 250 epochs (200 epochs with a frozen efficientNet backbone, then 50 epochs with an all trainable model starting from the best model obtained using a frozen backbone) by applying stochastic gradient descent (SGD). For all experiments, the thresholds for detection's non-maximal suppression (NMS) are $c_1 = 0.3$ and $IoU_{min} = 0.3$. The tracking is performed on detections, then when the track to detection association is completed, the detections having a score less than $c_3 = 0.7$ are discarded. An individual is counted if it appears on at least $m = 2$ frames with a confidence score greater than $c_2 = 0.7$. Note that ($c_1 \leq c_2$ and $c_1 \leq c_3$). We must mention that although the counting mAP is independent of c_2 and c_3 scores, it is calculated using $c_1 = 0.3$. The threshold for appearance metrics is $t_2 = 25$, the threshold for box overlaps is $t_1 = 1$, the association radius is $r = 25$, and the number of frames after

[1] https://wwz.ifremer.fr/

which a track dies when no association is done is $T_{Lost} = 20$. All those parameters were fixed using the validation video.

B. Results and discussion

As we see in Table I, in comparison to DeepSORT, our method enhances MOTA even though it raises the number of false-positive detection. Besides, robust DeepSORT further reduces the number of identity switches by **96.67%** w.r.t DeepSORT without raising the number of merged ground truths, which stays null. However, the algorithm looks at two images simultaneously, propagating false-positive detections and increasing ID's false positives. Nevertheless, the improvement in ID's true positives is more significant, providing a **4.89%** progress in IDF1. As DeepSORT w.r.t. SORT, our adjustment increases the number of fragmentations; this is due to the reduction of identity switches at the beginning of tracks, which encourages maintaining identities through occlusion and missing detections.

From Table II, we see that the use of counting thresholds leads to fewer counting false positives w.r.t. ID false positives while keeping an increase in counting true positives. This is because the counting threshold helps discard all duplicated tracks caused by identity switch and tracking uncertainty at the beginning of trajectories.

Our method performs well in the case of crossovers. For example, in Fig. 5, it keeps track of identities even though the Actinopterygii occludes the Munida in the middle frame, as opposed to DeepSORT, which results in an identity switch and merge.

After training, the algorithm learns the parameter ω of the Non-Local Blocks. It gives more importance to attention feature maps in classification $(-0.1253,\ 0.79)$ than in regression $(-0.0002,\ -0.0042)$. Fig. 6 shows the contribution of the second frame features to the point marked in red on the first frame, according to the classifier's first attention block. We see that the most stronger attention is received from the object (Munida) area within the second frame.

With our adjustment, we kept the ability of the algorithm to run in real-time. The robust DeepSORT, implemented using Pytorch[2], processes, on average, nine frames per second on Nvidia GeForce RTX 2060.

V. Conclusion

This paper proposes an improvement of tracking based on EfficientDet by using attentions between two successive images and extend DeepSORT to a method that does not rely only on the Kalman filter for motion prediction but also uses the deep detection network for that part. The robust DeepSORT uses the detection network, i.e., EfficientDet, to predict motion and compute appearance similarity by leveraging long-range dependencies modeled by Non-Local Blocks. This adaptation improved tracking performances and reduced the number of ID switches and merges. It has also improved the counting performances. Furthermore, the tracking framework remains simple and runs in real-time.

[2]https://pytorch.org/

We will integrate information from different modalities in future work to enhance detection and tracking, even if objects in the image are occluded or hard to identify. We will also try to make the tracking problem more based on learning rather than treating it as an association problem.

References

[1] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 3464–3468, IEEE, 2016.

[2] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[3] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[4] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649, IEEE, 2017.

[5] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *arXiv preprint arXiv:1504.01942*, 2015.

[6] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.

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

[8] P. C. Mahalanobis, "On the generalized distance in statistics," National Institute of Science of India, 1936.

[9] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10781–10790, 2020.

[10] M. He, H. Luo, B. Hui, and Z. Chang, "Pedestrian flow tracking and statistics of monocular camera based on convolutional neural network and kalman filter," *Applied Sciences*, vol. 9, no. 8, p. 1624, 2019.

[11] W. Hu, X. Li, W. Luo, X. Zhang, S. Maybank, and Z. Zhang, "Single and multiple object tracking using log-euclidean riemannian subspace and block-division appearance model," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 12, pp. 2420–2440, 2012.

[12] L. Zhang and L. van der Maaten, "Structure preserving object tracking," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1838–1845, 2013.

[13] G. Koch, "Siamese neural networks for one-shot image recognition," 2015.

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[15] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*, pp. 6105–6114, 2019.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[17] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, IEEE Computer Society, 2017.

[18] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, "Generalized intersection over union: A metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 658–666, 2019.

[19] L. Zheng, Z. Bie, Y. Sun, J. Wang, C. Su, S. Wang, and Q. Tian, "Mars: A video benchmark for large-scale person re-identification," in *European Conference on Computer Vision*, pp. 868–884, Springer, 2016.

[20] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[21] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, no. 1, pp. 1–10, 2008.

[22] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

Approach	MOTA	MOTP	IDSW	Merge	FM	IDTP	IDFP	IDFN	IDP	IDR	IDF1
SORT	0.475	**0.772**	89	**0**	**1378**	4867	775	4060	0.863	0.545	0.668
DeepSORT	0.481	**0.772**	30	2	1388	4916	**726**	4011	**0.871**	0.551	0.675
Robust DeepSORT (ours)	**0.498**	0.767	**1**	**0**	1786	**5439**	993	**3488**	0.846	**0.609**	**0.708**

TABLE I: Tracking performance.

Approach	TP	FP	FN	P	R	mAP@0.5	Micro-F1	Macro-F1
SORT	321	**64**	137	**0.834**	0.701	0.732	0.762	0.729
DeepSORT	336	78	122	0.812	0.734	0.752	**0.771**	0.737
Robust DeepSORT (ours)	**357**	111	**101**	0.763	**0.779**	**0.794**	**0.771**	**0.743**

TABLE II: Counting performance.

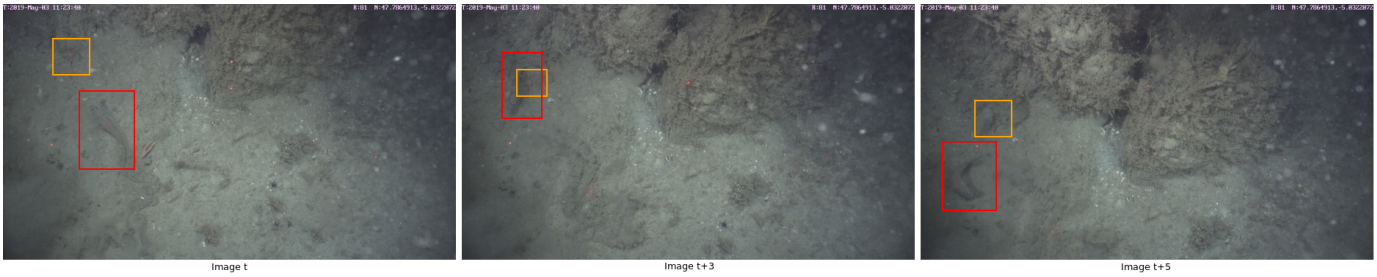

Fig. 5: Problem of identity switch and merge due to occlusion/crossover.

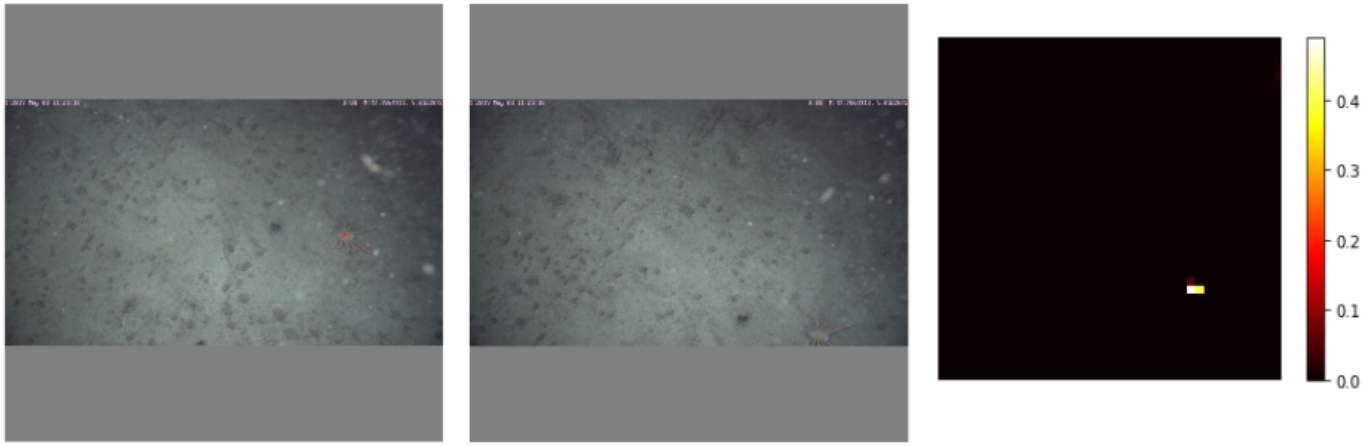

Fig. 6: Attention received from the second frame to the point marked by the red cross in the first frame.