Transformer Affinity for Tracking: Efficient reidentification of anchor-based detections in non-constant frame rate conditions

Abdelbadie Belmouhcine, Julien Simon, Sébastien Lefèvre

Transformer Affinity for Tracking: Efficient reidentification of anchor-based detections in non-constant frame rate conditions

Abdelbadie Belmouhcine[1], Julien Simon[2], and Sébastien Lefèvre[3,4]

[1] LabISEN, VISION-AD, ISEN Ouest, 29200 Brest, France
[2] DECOD, L'Institut Agro, IFREMER, INRAE, 56100 Lorient, France
[3] Univ. Bretagne-Sud, IRISA, 56000 Vannes, France
[4] UiT – The Arctic University of Norway, Tromsø, Norway

Abstract. Multiple object tracking (MOT) is an essential task in computer vision. It comprises a motion prediction module and an appearance-based reidentification mechanism. Relying only on motion prediction without incorporating appearance features results in identity switches, whereas using only appearance without motion prediction leads to identity merges. In this work, we are targeting real-time object detection and tracking in RGB images in an energy-restricted environment. The challenge in such a task is the uneven processing time between frames processed by the system, depending on the time required on each frame for detection and matching operations. As a result, some frames may be skipped until the system is ready to process a new one even if the camera streams at a fixed frame rate. Besides, when objects move in the camera's direction, they can change scale, making intra-scale association insufficient. To tackle these issues, we propose Transformer Affinity for Tracking (TransAT), which unifies the motion prediction and appearance reidentification in a single network and performs cross-scale matching. Our experiments validate our contributions and show that TransAT achieves successful reidentification of objects even when they change scales. Moreover, our method is robust to a drastic change in the frame rate, unlike DeepSORT, StrongSORT, and RetinaTrack.

Keywords: MOT · Multiple Object Tracking · SORT · DeepSORT.

1 Introduction

Multiple Object Tracking (MOT) is an extension of object detection. In addition to object localization, MOT must distinguish between the same category's instances and reidentify them across a succession of images. Most existing MOT algorithms tackle detection and tracking separately. First, an independent detector is used to detect objects, then the detections in two consecutive frames are associated to complete tracks. Therefore, they cannot benefit from each other as they treat geometric position prediction and appearance similarity individually and then combine them later. Indeed, tracking based only on appearance suffers from many ID merges, while tracking using only geometry leads to many ID switches. An ID switch (IDsw) occurs when a ground-truth trajectory is split into two predicted tracks, whereas an ID merge (IDmg) happens when

a predicted trajectory covers two ground-truths. In practice, a reidentification network carries appearance similarity computation while the geometric prediction is performed via a motion prediction model like Kalman filter [21], optical flow [1] or a recurrent network [18].

One simple but efficient tracking-by-detection technique is the well-known DeepSORT [41]. It uses a reidentification siamese network [44] to compute the similarity between objects in two frames and a Kalman filter to get the geometry involved in the matching phase by restricting the association area. This matching is performed using the Hungarian algorithm [22]. DeepSORT was upgraded in [13], by renewing its components. Nevertheless, as shown before [3], using the Kalman filter, whose uncertainty is high at initialization and increases as the model is not updated due to occlusions or to the detector's imperfections, leads to many ID switches. Moreover, an appearance-based reidentification network unaware of the location leads to ID merges, especially in datasets with high visual similarity between their objects, such as fish.

In this work, we apply MOT to trawl fisheries, which are known to have a high presence of cohabiting species [33]. The objective is to reduce bycatch in the case of quota-regulated conditions and to count captured species in real time. The intent is to build an intelligent fishing trawl driven by an embedded computer for autonomous machines that captures only desired fish and hence improves the environmental impact of fishing trawls.

Since, in real-time applications, the time taken by object detection depends on the image complexity and its processing time, the frame rate is not invariant. Thus, we need a tracking pipeline robust to frame rate change. Also, since tracked objects can move in the camera's direction, they can change scales between two successive images. Thus our tracking system must be robust to scale change, especially when the frame rate is low. Hence this paper presents a method called TransAT (Transformer Affinity for Tracking) that applies a transformer [40] between two images to predict 1) object displacement between them and 2) an affinity matrix between objects of those two images. It improves upon Fully DeepSORT [3] by enabling associations even when objects undergo scale changes between frames.

To sum up, TransAT applies a transformer between EfficientDet's [38] feature maps to calculate an affinity matrix using appearance and positions and to predict the posterior position using appearance. Nevertheless, the vanilla Transformer is unsuitable for interaction between two different images [24] since it uses self-attention [46] in the encoder. Thus, as we want to learn a matching between two images, we used two encoders, one for each image, and a decoder that performs cross-attention between the two encodings. Similar to [24], we apply an encoding-decoding step in each layer instead of stacking L encodings and then L decodings.

The contributions of this paper are summarized as follows:

- We propose a tracking method easily integrated on top of a pre-trained anchor-based detector. This method outputs an affinity matrix along with estimated posterior object positions. It does not need to learn any explicit vector representation for similarity computation and does not have to know any information about the object trajectory pattern.

– We introduce a tracking technique that uses a transformer to establish a matching between objects of two images rather than to aggregate features of a single one.
– Our tracking approach is robust to significant frame rate changes and keeps stable counting performance.

Our paper is organized as follows; We review related work in Sec. 2. We then expose our proposed technique and highlight our contributions in Sec. 3. We present the experimental setup in Sec. 4. We finally conclude the paper in Sec. 5.

2 Related Work

2.1 Multi Object Tracking

Most tracking by detection approaches [2,6,9,14,27,41] are composed of two branches: one performs object detection, and the other extracts features for reidentification. They strongly rely on the detector's performance. The most simple but effective tracking approach is Simple Online Realtime Tracking [6] (SORT). Having bounding boxes from a detector as input, it uses the Kalman filter [21] motion model to estimate the movement of each object between two consecutive frames and the Hungarian algorithm [22] for bipartite data assignment. The association cost is the intersection over union (IoU) between every detection in the current frame and the estimated positions of all living tracks in that frame. SORT suffers from a high number of identity switches due to the high uncertainty of the Kalman filter at the beginning of tracks or when the motion model is not updated. DeepSORT [41] extends SORT by using both motion and appearance in the affinity computation process. The goal is to reduce the number of IDsw. However, DeepSORT uses the appearance affinity only after a certain number of frames, which does not solve the problem of ID switches at the beginning of trajectories. Moreover, DeepSORT uses crops of images corresponding to bounding boxes to build the vector representation for similarity computation and does not leverage any information about the object location, leading to ID merges for similar entities. Accordingly, DeepSORT still uses the Kalman motion model to restrict the association area. StrongSORT [13] revisited and upgraded DeepSORT. It employs two offline postprocessing techniques: Appearance-Free Link (AFLink) and Gaussian Smooth Interpolation (GSI). AFLink

(a) Image $t - 10$ (b) Image t

Fig. 1: Two identical but distinct instances appear in the same zone of two images separated by ten frames. We can see two different langoustines; if we only consider their appearance, they are identical. Moreover, they appear in the same area, so if the tracker does not get images between the two frames, it will make an identity merge.

eliminates some kinds of ID switches, and GSI reduces fragmentations by filling the gaps in tracks. However, they do not affect most ID switch and merge cases. In addition, they are not suitable for real-time applications. Fully DeepSORT [3] showed that integrating the motion estimation into the deep detection network helps reduce identity switches further. However, Fully DeepSORT fails when objects change scales between two frames since it makes fusion using only features of the same scale.

Many approaches combined detection and tracking [4,8,26,47]. Tracktor [4] adapts Faster RCNN [30] to perform tracking. However, it stays limited to high frame-rate videos. DEFT [8] is similar to DeepSORT. Still, it does not use the Kalman filter for motion prediction but relies on an LSTM [18] model instead and applies a Deep Affinity Network [36] to compute appearance-based affinity between objects. However, it suffers from IDsw like DeepSORT. RetinaTrack [26] extracts an embedding vector for each anchor and uses greedy bipartite matching at inference time to associate tracks. The association is based on an embedding learned by a reidentification network that shares a part with the detection network. RetinaTrack can be seen as a simplified DeepSORT without the Kalman filter. However, as Tracktor, it is not useful if frames are not consecutive and the frame rate is not sufficiently high. Hence, after not updating a track for a certain number of frames, the matching is accomplished exclusively based on the appearance because the position is no longer used. Thus, we can get ID merges and switches when objects have almost the same visual appearance (Fig. 1).

2.2 Transformers

Transformers [40] are autoregressive models. A stack of L same layers forms the encoder. Each layer comprises a multi-head self-attention (MHSA) and a fully connected feed-forward head (FFH). After each layer's component, a residual connection and a layer normalization are applied. A stack of L same layers also forms the decoder. In addition to the two main components (MHSA and FFH) forming the encoder layers, the decoder adds a third sublayer that performs multi-head cross-attention (MHCA) using the encoder's output. Likewise, each layer component is followed by a residual connection and a layer normalization. The main module of the vanilla Transformer is the scaled dot product attention, which maps a query and a pair of key-value to an output, as follows:

$$\begin{cases} Attention(Q,K,V) = Softmax_{col}(\frac{QK^T}{\sqrt{d_k}})V \\ \text{Softmax}_{col}(A) = exp(A) \oslash ((exp(A)\mathbf{1})\mathbf{1}^T) \end{cases} \tag{1}$$

where $\oslash$ is Hadamard division of two matrices, $Q \in \mathbb{R}^{T\times d_k}$, $K \in \mathbb{R}^{M\times d_k}$, $V \in \mathbb{R}^{M\times d_v}$, T and M are respectively lengths of query and key sequences, d_k is the dimension of the query and the key, and d_v is the dimension of the value.

The multi-head (MH) attention allows the model to use information from different representations at different positions at the same time and is defined as:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{2}$$

$$MH(Q,K,V) = Concat(head_1, head_2, \ldots, head_H)W^O \tag{3}$$

where $W_i^Q \in \mathbb{R}^{d\times d_k}$, $W_i^K \in \mathbb{R}^{d\times d_k}$, $W_i^V \in \mathbb{R}^{d\times d_v}$, $W^O \in \mathbb{R}^{d\times d}$, $d_k = d_v = \frac{d}{H}$ and H is the number of heads. In multi-head self attention, $Q = K = V$ while in multi-head cross-attention $Q \neq K$ and $Q \neq V$.

Shifted Windows (SWin) Transformer [25] is a multi-stage hierarchical model that partitions feature maps into non-overlapping windows and computes the self-attention locally within those windows, reducing the high computational cost of self-attention. Then to allow interactions between different windows, window partitioning is gradually shifted along the network hierarchy.

2.3 Transformers in MOT

Transformers were first introduced to computer vision by Dosovitskiy et al. [12] through the Vision Transformer (ViT) for image classification. Subsequently, Carion et al. [7] proposed DETR, which reformulates object detection as a set prediction problem using a Transformer encoder–decoder architecture. This formulation has inspired a wide range of Transformer-based multi-object tracking approaches.

Early DETR-based trackers include TransTrack [34], which uses object queries for detection and track queries propagated across frames for association. TrackFormer [28] further unifies detection and tracking as a set prediction problem, enabling joint prediction of object states across frames. MOTR [45] extends this paradigm by introducing fixed-length detection queries and dynamic tracking queries updated over time. Several subsequent works enhanced the MOTR framework. MOTRv2 [48] improves detection quality by initializing object queries using proposals from a pre-trained YOLOX detector, alleviating conflicts between detection and association. MeMOTR [16] introduces trajectory-aware track queries with a tracklet-aware label assignment strategy to ensure one-to-one correspondence between queries and targets. MOTRv3 [43] and CO-MOT [42] further refine query supervision and interaction, adopting release–fetch supervision and cooperative label assignment with shadow queries, respectively. DN-MOT [15] addresses severe occlusions via denoising training and cascaded mask coordination, while MOTIP [17] formulates tracking as direct identity classification using a learnable ID dictionary. Recently, Shao et al. [32] proposed FDTA (From Detection to Association), which refines object embeddings through three specialized adapters: a Spatial Adapter (SA) that integrates 3D geometric information, a Temporal Adapter (TA) that captures temporal dependencies across frames, and an Identity Adapter (IA) that enhances instance-level discrimination.

Despite their elegant end-to-end formulation, most Transformer-based trackers rely on DETR-like architectures that jointly handle detection and tracking, often resulting in high training and inference costs. FastTrackTr [23] mitigates this issue by reducing query redundancy through efficient inter-frame information transfer. Moreover, unified detection–tracking frameworks often lack explicit discriminative constraints, leading to high inter-object similarity and degraded tracking performance [32], particularly in scenarios with visually similar instances such as animals. In contrast to DETR-based trackers, our approach builds upon an existing anchor-based detector and employs a Transformer specifically for reidentification and association. This design preserves the efficiency and robustness of anchor-based detection while leveraging attention mechanisms to improve identity consistency over time.

2.4 EfficientDet

For the detection step, we used EfficientDet [38], which is an anchor-based detector, to which we added an association component. To extract features from an image I_t, EfficientDet uses EfficientNet [37] as a backbone together with Bidirectional FPN [38]. Outcomes are five feature maps $\{f_t^i\}_{i=1...5}$ of different spatial resolutions $S_i \times S_i$ where $S_i = 5 \times 2^{5-i}$. Those feature maps are passed in parallel through classification and regression heads. For each anchor and pixel of each feature map, the regression head outputs an offset. r_t is the matrix of all offsets corresponding to an image I_t. Likewise, for an image I_t, the classification head outputs classes (c_t) with their corresponding scores (s_t). The offsets are applied to anchors K in order to produce bounding boxes b_t of objects in the image I_t.

We adopt EfficientDet as our detector since its anchor-based multi-scale architecture leads to a detection cost that depends on scene complexity and object density. On embedded hardware, this variability is amplified by limited computational resources, resulting in irregular processing times.

3 Proposed method

In this work, we target real-time fish counting to enable the cessation of fishing for a given species once its quota is reached, thereby reducing bycatch. Missing a detection does not significantly affect counting, as accurate counts can be obtained if each object is detected at least once and successfully reidentified. Nevertheless, many IDsw and IDmg compromise the counting because IDsw increases the counting's false positive, and IDmg increases the counting's false negative. To reduce IDsw and IDmg under non-constant frame rates, we cannot rely on an object's motion pattern to predict its next position, and we cannot rely only on its appearance when dealing with objects of almost the same visual appearance. Therefore, we propose a method called TransAT (Transformer Affinity for Tracking) reidentifying objects between frames regardless of their trajectories. The reidentification becomes independent of the frame rate and exploits position and appearance to associate objects. It is also resistant to scale change which is noticeable when the frame rate is low, and objects are moving toward the camera.

3.1 TransAT

To obtain a representation for each feature map with respect to an anchor, we apply m separable convolutions [10] followed by an instance normalization [39] and an ELU [11] to each post-BiFPN feature f_t^i of an image I_t. This operation is performed for all anchors. The results are anchor embeddings $f_t^{k,i}$ associated with each feature map of EfficientDet and each anchor. They are expressed as follows:

$$f_t^{k,i} = \text{Embedding}_k(f_t^i) = \text{SeparableConvbloc}_k^{(\times m)}(f_t^i) \tag{4}$$

where K is the set of anchors and $k \in \{0, \ldots, |K| - 1\}$ corresponds to indices of anchors. The flattened concatenation of anchor embeddings of each feature map f_t^i

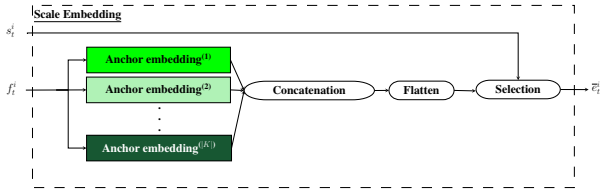


Fig. 2: The computation of all anchor embeddings of the i$^{\text{th}}$ scale. f_t^i is the i$^{\text{th}}$ feature map of the image I_t, and s_t^i represents their corresponding ranked classification scores taken from the classification head of EfficientDet. The output $\overline{e}_t^i$ corresponds to anchor embeddings of objects of the i$^{\text{th}}$ scale, having N top classification scores.

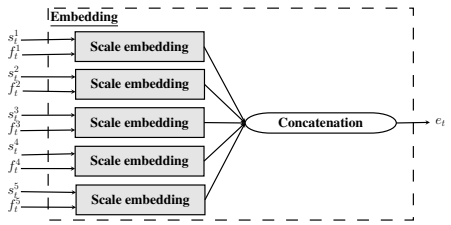


Fig. 3: The computation of the final object embeddings. The Scale Embedding block is described in Fig. 2. The output corresponds to the concatenation of all the five scale embeddings. All Scale embedding modules share weights (are the same).

results in its embedding e_t^i, which is a matrix of size $N_i \times d$ where $N_i = |K| \times S_i \times S_i$ is the number of detections of the i^{th} scale and d is the dimension of the latent space representation. For each embedding e_t^i we select top N vectors based on the scores corresponding to the classification of its corresponding objects to produce $\overline{e}_t^i$ of size $N \times d$. Fig. 2 shows the process of the computation of $\overline{e}_t^i$ for a particular feature map f_t^i. We concatenate the resulting matrices of the five feature maps to produce a final object embedding matrix e_t of size $5N \times d$. Fig. 3 illustrates the process of building the final object embeddings.

To compute the affinity between objects of two images I_t and $I_{t-\delta}$, we compute their object embeddings e_t and $e_{t-\delta}$. Then we pass them through a transformer. Fig. 4 displays the architecture of our transformer. It encodes features of two images by leveraging their Multi-Head Self Attention [40] using two separate stacks of encoders. A stack of decoders mixes the outputs of the encoders using a cross-attention to produce two cross-attention feature maps $d_{t-\delta \to t}^l$ and $d_{t \to t-\delta}^l$. As illustrated in Fig. 5, at each layer l, two encoders [40] output a self attention feature map per image: $x_{t-\delta}^l$ and x_t^l, then a decoder computes two cross-attention feature maps $d_{t-\delta \to t}^l$ and $d_{t \to t-\delta}^l$ for $x_{t-\delta}^l$ and x_t^l. Since for each of the five scales of EfficientDet we selected N detections with top classification confidence scores, we can split object embedding into five windows of size N and then apply the shifted windows strategy of SWin Transformer to compute attention only within windows. Then, to enable cross-attention computation between objects of different scales (an object can change scale between two images), we shift

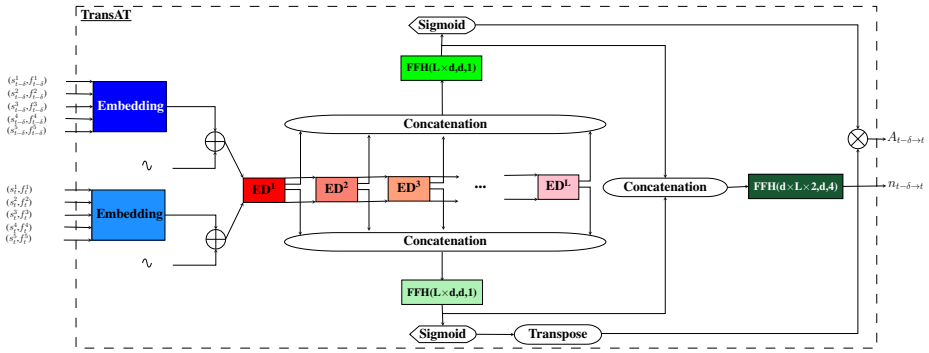


Fig. 4: The whole process of TransAT. Embedding block is displayed in Fig. 3. The symbol ∿ represents the positional embedding. Different colors mean different weights. The inputs are (from top to bottom): object embeddings of the image $I_{t-\delta}$, positional embeddings of its objects $\text{pe}_{t-\delta}$, object embeddings of the image I_t and positional embeddings of its objects pe_t. The outputs are (from top to bottom): the affinity matrix and the prediction of the displacement of objects in image $I_{t-\delta}$ from that image to image I_t. ED blocks are detailed in Fig. 5 whereas FFH blocks are detailed in Fig. 6. Different colors mean different weights.

the windows of the first image by a multiple of N while keeping those of the second image fixed. Let L be the number of layers, all $\{d^l_{t-\delta\to t}\}_{l=1\dots L}$ and all $\{d^l_{t\to t-\delta}\}_{l=1\dots L}$ are concatenated and then passed through a Feed Forward Head (FFH) to produce two probability vectors $Y_{t\to t-\delta}$ representing the probability that an object of I_t was present in $I_{t-\delta}$, and $Y_{t-\delta\to t}$ representing the probability that an object of $I_{t-\delta}$ appears in I_t. The final affinity matrix $A_{t-\delta\to t}$ of size $5N \times 5N$ is computed by multiplying the two probability vectors, assuming that the two distributions are independent. The affinity matrix stops tracks when an object is not likely to exist in a particular frame. Fig. 6 shows the Feed Forward Head architecture, which takes an input of size $N \times d_i$ and produces an output of size $N \times d_o$. Unlike the vanilla transformer [40], which stacks L encoders and then L decoders, we perform an encoding-decoding step in each of the L layers, as shown in Fig. 4. In order to include the position of objects in the learning process, we applied sine and cosine functions as in [40] to compute a 2D positional embedding pe^i_t for objects of the i^{th} scale of the image I_t. Equation 5 defines the output of the encoders, whereas Equation 6 defines the output of the decoder. The affinity matrix is computed as follows: $A_{t-\delta\to t} = Y_{t-\delta\to t} Y^T_{t\to t-\delta}$. We called the whole process Transformer Affinity for Tracking (TransAT), illustrated in Fig. 4.

$$\begin{cases} x^l_{t-\delta} = \text{Encoder}^l_1(x^{l-1}_{t-\delta}) \\ x^l_t = \text{Encoder}^l_2(x^{l-1}_t) \\ x^0_{t-\delta} = e_{t-\delta} + \text{pe}_{t-\delta} \\ x^0_t = e_t + \text{pe}_t \end{cases} \tag{5}$$

$$d^l_{t-\delta\to t}, d^l_{t\to t-\delta} = \text{Decoder}^l(x^l_{t-\delta}, x^l_t) \tag{6}$$

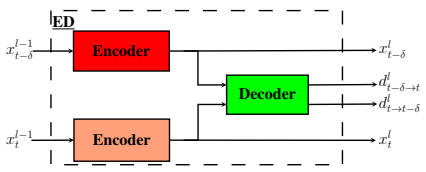


Fig. 5: Encoding-Decoding Layer. We bypass the decoder to input the encoded features to the next encoder as specified by Equation 5. Different colors mean different weights.

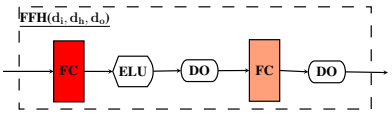


Fig. 6: The Feed Forward Head (FFH). It takes an $N \times d$ input, then passes it through a Fully Connected (FC) layer followed by an ELU and a Drop Out (DO), to produce an intermediate result of size $N \times d_h$, which is then passed through another Fully Connected layer followed by a Drop Out to produce a final result of size $N \times d_o$. Note that d_i, d_h and d_o are hyper-parameters.

3.2 ID assignment

To assign IDs to objects of an image I_t, we use images $\{I_{t-\delta}\}_{\delta=1...T_{Lost}}$ and compute the affinity matrices $\{A_{t-\delta\rightarrow t}\}_{\delta=1..T_{Lost}}$. After Non-Maximum Suppression, we keep only rows and columns corresponding to the remaining bounding boxes. For each image $I_{t-\delta}$, we compute the standardized pixel distance [19] between $n_{t-\delta\rightarrow t}$, the predicted position of the bounding boxes of $I_{t-\delta}$ in I_t, and the bounding boxes b_t of image I_t. Indeed, to reduce the difference induced by the depth of field between the pixel distance and the actual distance, standardized distance normalizes the pixel distance between two bounding box centers by the minimum width and height of their bounding boxes. The following formula gives the standardized distance:

$$D_{sp} = \sqrt{\left(\frac{x_1 - x_2}{min(w_1, w_2)}\right)^2 + \left(\frac{y_1 - y_2}{min(h_1, h_2)}\right)^2} \tag{7}$$

where (x_1,y_1) and (x_2,y_2) are coordinates of two bounding box centers, w_1 and w_2 are widths of the bounding boxes, and h_1 and h_2 their heights. The result is a set of distance matrices $\{D_{t-\delta\rightarrow t}\}_{\delta=1...T_{Lost}}$. All costs corresponding to distances greater than a threshold α (fixed empirically) are replaced by -1. We then weight those costs using the products of scores of their corresponding pairs to encourage matching between objects of high confidence. The score of an ID is the average of scores of detections of its corresponding track. Furthermore, to get the age of images involved, so that old images weigh less than recent ones, we used exponential moving average. Let $\text{EMA}_{\delta=1...T_{Lost}}$ be the exponential moving average, the association matrix C_t used for ID assignment

to objects of image I_t based on the last T_{Lost} images is computed as follows:

$$\begin{cases} C_t = A_t \oslash D_t \\ A_t = \text{EMA}_{\delta=1\ldots T_{Lost}}(A_{t-\delta\to t}) \\ D_t = \text{EMA}_{\delta=1\ldots T_{Lost}}(D_{t-\delta\to t}) \end{cases} \tag{8}$$

Finally, the Hungarian algorithm is used for bipartite matching. We discard all associations having a negative value and all correspondences between objects of different categories.

4 Experiments and Results

4.1 Experimental setup

We evaluated our model using a seabed species detection and tracking dataset collected by the French Research Institute for Exploitation of the Sea (IFREMER)*. This dataset contains images with ground-truth object IDs and a strong visual resemblance between entities of the same category, as shown in Fig. 1. While standard MOT datasets such as MOT17 [29] or DanceTrack [35] are widely used, they do not capture the specific conditions of our application. Our dataset reflects the challenges of real-world seabed imagery, including visually similar species, overlapping objects, and the natural variability of the underwater environment. Consequently, using this dataset allows us to directly evaluate our method for real-time fish counting, which is the main focus of this work.

To build our dataset, a camera was placed on a sledge and recorded 2048×1152px videos of the seabed at 12 fps. The dataset is composed of 48 videos. We used 43 videos for training, one video for validation and parameter tuning, and four videos for testing. Objects in the dataset are assigned to five categories [3]. All data were annotated manually by specialists in IFREMER. We used this dataset since our target application is the creation of an autonomous fishing trawl.

To evaluate the tracking performance, we used the most famous evaluation metrics, which are: i) CLEAR MOT metrics, i.e. Multi-object tracking accuracy (MOTA), Multi-object tracking precision (MOTP), Identity switches (IDsw), Identity merges (IDmg) and Fragmentation (FM) [5]. ii) ID metrics, i.e. ID precision (IDP), ID recall (IDR), and ID F_1 score (IDF1) [31].

We grouped each image with its following one, allowing the model to learn direct associations. However, it is essential to learn long-range associations to make temporal learning and be able to reidentify objects after absences. Thus, in addition to direct associations, we produce up to five pairs for each image by taking up to five of its subsequent images containing at least one common identity.

To learn the affinity matrix, we used the dual focal loss (DFL) [20] which is suitable for cases of class imbalance. Moreover, to make the network focus more on the associations between objects with high classification confidence scores, we weight this loss by the product of the confidence scores of its matched pair of objects.

*https://wwz.ifremer.fr/

For the network hyperparameters, we utilized a latent space dimension $d = 88$, $L = 10$ for the number of layers in the transformer, two heads in the MHSA block, and $N = 20$ for selection in the embedding phase. The threshold for the standardized pixel distance is $\alpha = 5$, and the number of frames after which a track dies when no association is done is $T_{Lost} = 20$. All parameters were fixed empirically using the validation video.

4.2 Results and discussion

From Table 1, we can see that TransAT provides slightly better results than Fully DeepSORT. It further eliminates the few remaining IDsw of Fully DeepSORT caused by scale changes since Fully DeepSORT only makes the fusion of features of the same scales. Therefore, we obtain an excellent reidentification with just one IDsw and one IDmg. Fig. 7 shows an example of IDsw.

When we apply only IoU between detections as an affinity (with a threshold of 0.9), we get too many IDsw and IDmg, because our dataset has a low frame rate and exhibits varying object motion.

Although the quality of RetinaTrack's detections is better than those of EfficientDet (as reflected by the MOTP in Table 1), the tracking of RetinaTrack suffers from many IDsw and IDmg. Indeed, in most cases, RetinaTrack only uses an appearance-based similarity to make the association because the IoU is not helpful in our dataset, as we can see with the IoU baseline.

Approach	**MOTA** ↑	**MOTP** ↑	**IDsw** ↓	**IDmg** ↓	**FM** ↓	**IDTP** ↑	**IDFP** ↓	**IDFN** ↓	**IDP** ↑	**IDR** ↑	**IDF1** ↑
IoU	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	0.950	0.478	0.636
DeepSORT++	**0.492**	0.776	20	**0**	**22**	5305	387	4885	0.932	0.522	0.668
StrongSORT	0.461	0.781	18	**0**	593	4883	246	5307	0.952	0.479	0.638
StrongSORT++	**0.492**	0.776	17	**0**	**22**	5322	378	4868	0.934	0.522	0.670
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	0.640
Fully DeepSORT++	0.489	0.775	4	**0**	47	5340	401	4847	0.930	0.524	0.671
TransAT	0.462	0.781	**1**	1	665	4910	**216**	5277	**0.958**	0.482	0.641
TransAT++	0.489	0.775	**1**	1	47	**5358**	386	**4829**	0.933	**0.526**	**0.673**

Table 1: Tracking performance.

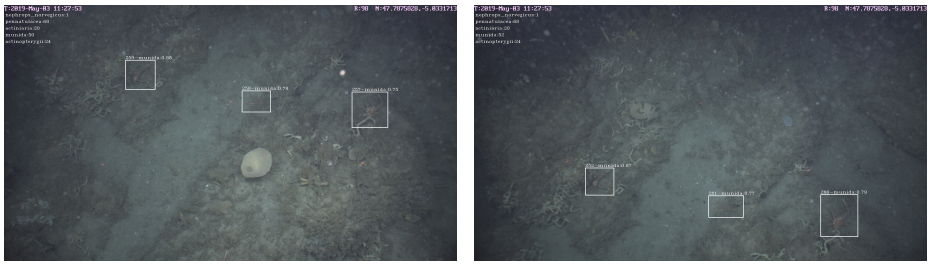
Our method differs from DETR-based tracking approaches, which rely on joint end-to-end detection and tracking trained on large-scale datasets. In contrast, we build on a pre-trained anchor-based detector, a design better suited for real-time deployment in resource-constrained and application-specific settings. We focus our evaluation on a domain-specific seabed dataset rather than on standard MOT benchmarks such as MOT17 [29] or DanceTrack [35]. Although these benchmarks involve human tracking, our target scenario presents distinct challenges, including strong intra-class visual similarity, significant scale variations due to depth changes, and frequent object appearances and disappearances.

In our case, our goal is to do real-time detection and tracking with an autonomous machine that uses an embedded GPU and runs on a restricted-energy budget. Our demersal trawl generally operates on the seabed deep underwater. Consequently, the processing time for each frame may vary. It includes the time taken by the object detector, the time needed for the tracker, and the time consumed by the autonomous actions and network communication. For example, using an NVIDIA Jetson AGX Xavier[†] on 15W power mode, EfficientDet D1 took in the best case 0.02 second and in the worst case, 1 second. That means the detector can process between 50 fps and 1 fps, depending on the image complexity and the number of detections having sufficient confidence to enter the NMS step. Therefore, to show that our method can still work better in case of a high variation of frame rates going from 12 fps (the frame rate of our videos) to 0.5 fps, we conducted several experiments by changing the aggressiveness of random frame rate variation. The results are reported in Fig. 8 and show that the frame rate variation deteriorates the performance of DeepSORT and StrongSORT. Fully DeepSORT gets less impact, but its number of IDsw + IDmg rises as the frame rate variation increases because the impact of scale changing becomes noticeable. However, TransAT gets impacted less and keeps similar performance even with a substantial variation in frame rates because it better treats cases of scale changes.

5 Conclusion

In this paper, we proposed a Transformer based tracking method that, unlike Transformer based trackers in the literature, can be used with an anchor-based detector like EfficientDet. This is particularly advantageous because anchor-based detectors are often pre-trained on smaller datasets, lightweight, and capable of running in real-time on embedded or restricted-energy devices, unlike many end-to-end DETR-based approaches that require large datasets and high computational resources. Our approach, called TransAT, uses a single network for detection and tracking and makes a motion

[†]https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-agx-xavier/



(a) Image t (b) Image $t + 1$

Fig. 7: Example of 3 IDsw caused by DeepSORT due to a variation in frame rate and a low temporal resolution. On the same two images, TransAT does not make any IDsw.

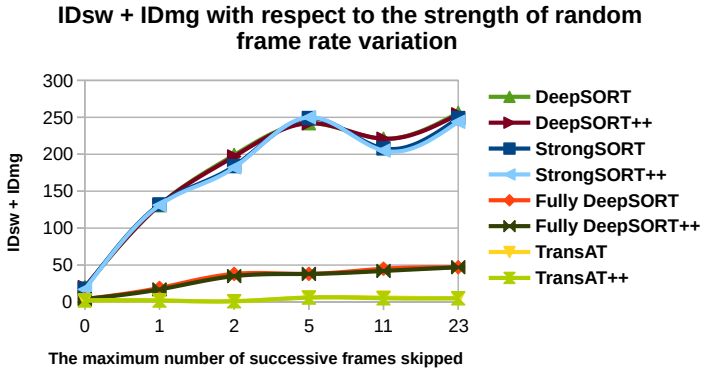


Fig. 8: IDsw + IDmg with respect to the strength of random frame variation

prediction that leverages appearance. In addition, it also outputs an affinity matrix representing the probability of an association. TransAT computes cross-attention in the transformer's decoder and allows cross-scale fusion. Indeed, it is an improvement over Fully DeepSORT.

Our proposed method reduces the ID switches and merges, which are the main problems of respectively DeepSORT and RetinaTrack, allowing a successful reidentification. Furthermore, it uses a Shifted Windows as in SWin Transformer but performs shifting based on scales and only on one image to allow the correspondence between features of different scales. Finally, it is more robust to aggressive frame rate variation since it is less impacted by object scale changes between images, which is the main weakness of Fully DeepSORT. For those reasons, TransAT is suitable for tracking objects moving toward a camera in a restricted-energy scenario.

In the future, we will use stereo videos to track objects in 3D space, which will be beneficial for tracking similar but different objects in crowded scenes. We will also study the impact of integrating acoustic data in the tracking process.

References

1. Beauchemin, S.S., Barron, J.L.: The computation of optical flow. ACM computing surveys (CSUR) **27**(3), 433–466 (1995)
2. Belmouhcine, A., et al.: Robust deep simple online real-time tracking. In: International Symposium on Image and Signal Processing and Analysis (ISPA). pp. 138–144. IEEE (2021)
3. Belmouhcine, A., et al.: Fully Deep Simple Online Real-time Tracking: Efficient Re-Identification by Attention without Explicit Similarity Learning. In: International Conference on Pattern Recognition (ICPR) (2022)
4. Bergmann, P., et al.: Tracking without bells and whistles. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 941–951. IEEE (2019)
5. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. EURASIP Journal on Image and Video Processing **2008**, 1–10 (2008)

6. Bewley, A., et al.: Simple online and realtime tracking. In: 2016 IEEE international conference on image processing (ICIP). pp. 3464–3468. IEEE (2016)
7. Carion, N., et al.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
8. Chaabane, M., et al.: Deft: Detection embeddings for tracking. arXiv preprint arXiv:2102.02267 (2021)
9. Chen, L., et al.: Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: 2018 IEEE international conference on multimedia and expo (ICME). pp. 1–6. IEEE (2018)
10. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
11. Clevert, D.A., et al.: Fast and accurate deep network learning by exponential linear units (elus). arXiv preprint arXiv:1511.07289 (2015)
12. Dosovitskiy, Alexey and others: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: International Conference on Learning Representations (2020)
13. Du, Y., et al.: Strongsort: Make deepsort great again. IEEE Transactions on Multimedia **25**, 8725–8737 (2023)
14. Fang, K., et al.: Recurrent autoregressive networks for online multi-object tracking. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 466–475. IEEE (2018)
15. Fu, T., et al.: Denoising-mot: Towards multiple object tracking with severe occlusions. In: Proceedings of the 31st ACM International Conference on Multimedia. pp. 2734–2743 (2023)
16. Gao, R., Wang, L.: MeMOTR: Long-term memory-augmented transformer for multi-object tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9901–9910 (2023)
17. Gao, R., et al.: Multiple object tracking as id prediction. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 27883–27893 (2025)
18. Graves, A.: Long short-term memory. In: Supervised sequence labelling with recurrent neural networks, pp. 37–45. Springer (2012)
19. He, M., et al.: Pedestrian Flow Tracking and Statistics of Monocular Camera Based on Convolutional Neural Network and Kalman Filter. Applied Sciences **9**(8) (2019). https://doi.org/{10.3390/app9081624}, https://www.mdpi.com/2076-3417/9/8/1624
20. Hossain, M.S., et al.: Dual Focal Loss to address class imbalance in semantic segmentation. Neurocomputing **462**, 69–87 (2021)
21. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
22. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955)
23. Liao, P., et al.: FastTrackTr: Towards Fast Multi-Object Tracking with Transformers (2024)
24. Liao, S., Shao, L.: TransMatcher: Deep Image Matching Through Transformers for Generalizable Person Re-identification. Advances in Neural Information Processing Systems **34** (2021)
25. Liu, Z., et al.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10012–10022 (2021)
26. Lu, Z., et al.: Retinatrack: Online single stage joint detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 14668–14678. IEEE (2020)
27. Mahmoudi, N., et al.: Multi-target tracking using CNN-based features: CNNMTT. Multimedia Tools and Applications **78**(6), 7077–7096 (2019)

28. Meinhardt, T., et al.: Trackformer: Multi-object tracking with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8844–8854 (2022)
29. Milan, A., et al.: MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016)
30. Ren, S., et al.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015)
31. Ristani, E., et al.: Performance measures and a data set for multi-target, multi-camera tracking. In: European conference on computer vision. pp. 17–35. Springer (2016)
32. Shao, Y., et al.: From Detection to Association: Learning Discriminative Object Embeddings for Multi-Object Tracking. arXiv preprint arXiv:2512.02392 (2025)
33. Sokolova, M., et al.: A Deep Learning Approach to Assist Sustainability of Demersal Trawling Operations. Sustainability **13**(22), 12362 (2021)
34. Sun, P., et al.: Transtrack: Multiple object tracking with transformer. arXiv preprint arXiv:2012.15460 (2020)
35. Sun, P., et al.: Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 20993–21002 (2022)
36. Sun, S., et al.: Deep affinity network for multiple object tracking. IEEE transactions on pattern analysis and machine intelligence **43**(1), 104–119 (2019)
37. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019)
38. Tan, M., et al.: Efficientdet: Scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10781–10790. IEEE (2020)
39. Ulyanov, D., et al.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6924–6932 (2017)
40. Vaswani, A., et al.: Attention is all you need. Advances in neural information processing systems **30** (2017)
41. Wojke, N., et al.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE international conference on image processing (ICIP). pp. 3645–3649. IEEE (2017)
42. Yan, F., et al.: Bridging the gap between end-to-end and non-end-to-end multi-object tracking. arXiv preprint arXiv:2305.12724 (2023)
43. Yu, E., et al.: Motrv3: Release-fetch supervision for end-to-end multi-object tracking. arXiv preprint arXiv:2305.14298 (2023)
44. Zagoruyko, S., Komodakis, N.: Wide Residual Networks. In: British Machine Vision Conference 2016. British Machine Vision Association (2016)
45. Zeng, F., et al.: Motr: End-to-end multiple-object tracking with transformer. arXiv preprint arXiv:2105.03247 (2021)
46. Zhang, H., et al.: Self-attention generative adversarial networks. In: International conference on machine learning. pp. 7354–7363. PMLR (2019)
47. Zhang, Y., et al.: Fairmot: On the fairness of detection and re-identification in multiple object tracking. International Journal of Computer Vision **129**(11), 3069–3087 (2021)
48. Zhang, Y., et al.: Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 22056–22065 (2023)