A comparative study of joint video tracking and classification for countering Unmanned Aerial Vehicles

Guillaume Vaudaux-Ruth[a], Benjamin Pannetier[a], Vincent Marié[b], Matthieu Planas[c], Thomas Brethomé[a], Lionel Gardenal[b], and Anne-Laure Jousselme[b]

[a]CS GROUP Research Lab, 3 boulevard Thomas Gobert, Playground 91120 Palaiseau, France
[b]CS GROUP, 230 rue Marcellin Berthelot, 83079 Toulon Cedex 9, France
[c]CS GROUP, 22 avenue Galilée, 92350 Le Plessis Robinson, France

ABSTRACT

The rapid adoption of autonomous Unmanned Aerial Vehicles (UAVs) for various real-world applications in both industry and the military is driving the need for efficient UAV surveillance and countering systems, as these vehicles create new threats to the safety of people and assets. These systems typically contain a variety of sensors and effectors, including video sensors that are used for both human confirmation of a potential menacing UAV, and visual servoing of effectors used to counter an aerial threat. In this case, the performance of the system depends on the accuracy of the algorithm chain (classification, localization and threat identification) used for video tracking. In this paper, we study an original approach for temporally stable video tracking of targets. Specifically, we use state-of-the-art algorithms for semantic object detection and then consolidatethem with a pose estimation method to enhance the perception performance. This paper compares different approaches on real data.

Keywords: Video tracking, classification, deep learning, pose estimation

1. INTRODUCTION

Drones, which refers to small and remotely controlled Unmanned Aerial Vehicles (UAVs), are used in various situations such as construction, topographic mapping, inspection of large infrastructures,[1] ... However, while UAVs have benefits in societal roles, they can also be intentionally or unintentionally misused or used for illicit activities,[2] which poses a threat to the safety of others. This rapid development in the use and potential misuse of UAVs has consequently led an increase in research on drone detection[3] to counteract potential risks due to intrusion in restricted areas. The complementary and redundancy of sensors enables detection, tracking and classification of UAVs, enabling threat assessment and planning the proper counter-measure. Despite efforts to develop dedicated systems such as sensors, processing, hard and soft kill systems, counter Unmanned Aerial Systems (CUAS) are often deployed in complex areas, which requires an adequate understanding of the situation including the UAVs behavior.
Existing single-camera detection and tracking methods are mostly unsuitable for drone surveillance due to their limited field of view and the difficulty in accurately estimating the distance of objects far from the camera that occupy very few pixels in the video frame. Therefore, to improve detection, classification, and tracking methods, the existing literature[4] also points out sensor fusion as a solution to achieve more accuracy and robustness compared to a single sensor. However, if the data fusion module receives tracks with poor quality, data fusion algorithms may not improve the tracking process in a multiple-target context. Starting with our system presented in this paper, we studied algorithms to improve video track quality before fusion. In this work, we address the design and evaluation of an automatic drone detection and tracking system based on an optronic system. Proposed developments are built on state-of-the-art machine learning techniques, extending methods from conclusions and related literature recommendations.[5,6] In fact, over the last few years, significant

Further author information: (Send correspondence to B.P.)
G.V.: E-mail: guillaume.vaudaux-ruth@csgroup.eu,
B.P.: E-mail: benjamin.pannetier@csgroup.eu

improvements have been made in addressing a wide variety of long-standing problems in machine learning, artificial intelligence, and computer vision. One of the research topics that has gained attention is the use of computer vision technology to accomplish multi-object detection and tracking tasks. Multiple Object Tracking (MOT) refers to the detection of all targets in each frame of a continuous frame sequence of videos, and obtaining estimates of their position and speed, bounding boxes of each object, and assigning individual ID identifications for every object in each frame. In papers,[7,8] The authors address the Multiple Object Tracking (MOT) problem by exploiting two modules: a detection module to locate the objects of interest by bounding boxes in each frame, and a Re-ID module to associate each object with one of the existing trajectories. Based on this processing, and inspired from,[7] "We investigate a joint detection and multiple-target tracking approach based on deep learning to detect and track several UAVs, while minimizing false alarms. If the simple online and realtime tracking (SORT) approach,[9] which performs Kalman filtering in image space frame-by-frame followed by a global nearest neighbor standard filter, is used, it is not sufficient for our use case and must be extended with a multiple-target tracking approach.

Traditional methods are Multiple Hypothesis Tracking (MHT),[10] the Joint Probabilistic Data Association Filter (JPDAF),[11] and more recently the Gausian Mixture Probility Hypothesis Density (GMPHD).[12] These methods perform data association on a frame-by-frame basis. In the Joint Probability Data Association Filter (JPDAF), a single state hypothesis is generated by weighting individual measurements with their association likelihoods. In the Multi-Hypothesis Tracking (MHT) approach, all possible hypotheses are tracked, but pruning schemes must be applied for computational tractability. In the Gaussian Mixture Probability Hypothesis Density (GMPHD) approach, tracks are obtained by processing the intensity function with the management of group occlusion, where hierarchical data association is utilized to reduce false negatives caused by missed detection. Despite promising results,[10–12] machine learning community admits the better performance of these methods comes at increased computational and implementation complexity. So, research efforts often focus on detection or sequential detection. Despite these works, lessons learnt from our multiple-object tracking (MOT) algorithm show weaknesses with experimental context variation or with the variety of drone types, such as an increase in false detections, lost tracks, and incorrect ID associations. The extension to multiple hypothesis tracking (MHT) appears as a potential solution to improve multiple object tracking algorithms in order to be more resilient to these issues. The scientific question is how to improve MOT performance by increasing time processing with MHT approaches. What are the benefits of using MHT for MOT performance?

To answer these questions, we will briefly present related works and our system. In the second part, we will present both the drone detection approach and the multiple-object tracking algorithm, and explain how we use joint classification results in tracking association processing. Finally, we will provide some results on real data and conclude with future work.

2. RELATED WORKS

Whereas multiple surveys have already been published[13],[14] we firstly provide an overview of popular architectures and literature that addresses generic detection and tracking challenges. Then, we focus on the state of the art on the particular problem of UAV detection and tracking.

2.1 Object detection

Object detection is one of the most fundamental task studied in computer vision. Before the resurgence of neural-network in 2013, object detection was mainly done through classical machine learning techniques such as viola-jones object detection technique[15] or scale-invariant feature transforms (SIFT).[16] In these traditional ML-based approaches, computer vision techniques are used to look at various features of image, such as the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then fed into a regression model that predicts the location of the object along with its label. While these techniques showed good performances, their main disadvantages are small data size, poor portability, high time complexity, window redundancy, no robustness for diversity changes and they show good performance only in specific simple environments. Recent advances in artificial intelligence fostered by the explosion of the computing power (mainly via GPUs) and the abundance of big data benchmark (Caltech, PASCAL VOC, Microsoft COCO, Imagenet, etc) allowed the emergence of new deep neural network algorithms. Consequently, most of (if not all) the recent state of the art results in target detection task have been reported for deep neural network. It can be divided into two categories : the single-stage detection algorithms that treat object detection as a simple regression problem, and the two-stage detection algorithms based on region proposal.

Two-stage detection algorithms have two main steps. Generally, they start with first extracting some region of interest (RoIs) then classify and regress the RoIs. Among the most famous existing algorithms, we can find chronologically Fast-RCNN,[17] Faster-RCNN[18] and Mask-RCNN.[19] All these algorithms are based on RCNN[20] proposed by Girshick in 2014 which is the first real target detection model based on convolutional neural networks which achieves 66% mAP. Up to Mask-RCNN in 2017, the main contributions reside in the detection accuracy improvement and the ease of the learning process. Despite the improvements, these algorithms still cannot achieve real-time detection and are not well suited to applications where high resolution images are processed, where targets are highly maneuverable and where the camera moves between successive frames.

In 2015, a second family of detection algorithm appears. Firstly proposed by Joseph Redmon, YOLOv1 (You Only Look Once)[21] detection model does not require the region-proposal extraction process. During the following years, multiple versions of one-stage detection algorithm appears. Among us, we can cite the multiple version of YOLO (v2, v3, v4, v5, v6, v7, v8), Single Shot Detector[22] (SSD) or CornerNet.[23] For YOLO and SSD, the entire detection model is a simple CNN network structure which take an image as input and output the location and category of the bounding box at the output layer. Concerning CornerNet, the algorithm focuses on detecting pairs of keypoints at the top-left and bottom-right of each object. Despite the progress on object detection field, the process of detecting UAV still remain a difficult task to achieve. UAV have great speeds, can appear at different sizes and more particularly, the early detection goal requires to deals with small objects containing few pixels. Another problem in the aerial target detection field is the birds which leads to false detections. Over the past few years, many researchers have tried to solve these problems.

Saqib et al.[24] proposed flying object detection method using Faster R-CNN and VGG-16. The authors suggested annotating birds as a standalone class to enable the trained model to differentiate birds from drones and improve the performance with regard to reducing false positives. In[25] Aker et al. used a YOLOv2[26] architecture to differentiate UAV from birds. Schumann et al.[27] used both background-subtraction and deep-learning based object proposal methods depending on whether camera is static or moving. The latters are classified into UAV or birds by applying a CNN classifier. Lee et al.[28] proposed a detection method based on Haar Feature-based Cascade Classifier trained on 7000 positive examples (2088 original + 4912 generated by image distortion) and 3019 negative examples. Detections are finally fed into a CNN to identify the drone type. Nalamati et al.[29] experimented various approaches to detect small object and more precisely to differentiate UAVs from birds. They used a combination of two algorithms : Faster-RCNN and SSD with two CNN architectures to determine which combination is better. They divided their datasets consisting in 8771 frames in two parts: when UAVs are near the camera and when they are far away from the camera. In the first case, all the combinations lead to the same good results. In the second case the combination of Faster-RCNN and ResNet-101 better performed but authors didn't provide details concerning the processing times. In,[30] Magoulianitis et al. proposed to incorporate a Super-Resolution (SR) technique in the detection pipeline composed of Faster-RCNN algorithm to increase the overall performance. Craye and Ardjoune proposed in[31] a spatio-temporal semantic segmentation approach based on convolutional neural networks. The authors first identify areas of interest in the image with a U-net[32] architecture and then uses a ResNet CNN to determine whether those areas contain a UAV or not. Seidaliyeva et al.[33] used a background-subtraction technique to detect moving objects and a CNN architecture (mobileNet-v2) to classify the detected objects into three classes: UAV, bird and background. However, the performance of this approach relies on whether the background is static or dynamic. Akyon et al.[34] proposed to fine-tuned a YOLOv5 model with real and synthetically generated data to improve UAV detection confidence. In,[35] Elsayed et al. proposed a method divided in two phases: detection with background-subtraction method followed by a classifier and enhanced detection by tracking phase. The authors suppose a static background which is an obvious limitation.

2.2 Tracking

Regarding the target tracking problem, the goal is to locate objects and generate their trajectories in a video sequence. This task is more challenging than object detection which only aims at localizing objects in an image.

Object tracking can be divided in two categories: the Single-Object Tracking (SOT) introducing the challenge of distinguishing an object from the background and the Multi-Object Tracking (MOT) which goes further and tries to track multiple objects. In this paper, as we deal with multiple objects simultaneously, we focus on the MOT family. As for the detection, the recent interest of deep learning lead to the development of new trackers, focusing on the powerful feature extraction capability of deep neural networks. Generally speaking, the MOT algorithms are based on two main approaches: tracking by detection and joint detection and tracking. In the following, we present an overview of existing algorithms and state of the art based on these approaches.

2.2.1 Tracking by detection

Tracking by detection approach uses an object detector to locate objects in each processed frame. Then it associates the detected objects between frames into tracklets based on similarity scores. Yu et al.[8] use Faster R-CNN algorithm[18] to firslty detect the targets and developed a ReID network based on GoogLeNet[36] to extract discriminative appearance features. In[37] the authors learned images containing different targets simultaneously to predict more discriminative features. They proposed a quadruplet loss to enforce the constraint that makes temporally adjacent detections more closely located than the ones with large temporal gaps. Lee et al.[38] developed a ReID network with a feature pyramid network (FPN) to fuse features at different levels and consequently enrich the resulting information. Features-only trackers showed good performances but are not representative of real tracking scenarios where similar targets are present. Most recent two-stage methods combine the motion and the appearance features of targets for more robust tracking. Wojke et al.[9] proposed an algorithm called SORT composed of a Kalman Filter[39] to predict the next targets location and a data association process based on the Intersection over Union (IoU) score. In 2017, they proposed the DeepSORT algorithm,[7] a evolution of the previous algorithm which enrich the motion model with a ReID network. Chen et al.[40] proposed to remove unreliable detections and redundant trajectories by a scoring mechanism. Zhou et al.[41] used CNN to model the motion patterns and inter-target interactions of targets. In,[42] Li et al. designed an auto-tuning Kalman filter to more accurately predict next target location and evaluate the inter-target similarity by a recurrent neural network. More recently, Zhang et al.[43] proposed the ByteTrack tracking algorithm by associating every detection box instead of only the high score ones. For the low score detection boxes, authors utilize their similarities with tracklets to recover true objects and filter out the background detections. Du et al.[44] upgrade DeepSORT and developped the StrongSORT algorithm by adding two algorithms to further refine the tracking results: AFLink and Gaussian-smoothed interpolation (GSI).

2.2.2 Joint detection and tracking

Joint detection and tracking approach simultaneously solves detection and tracking tasks in a single framework. This allows better detection performance by using the tracking information. Voigtlaender et al.[45] proposed the TrackR-CNN algorithm by integrating a ReID head in Mask R-CNN.[19] Wang et al.[46] proposed to integrate a ReID head in YOLOv3[47] and Joint Detection and Embedding (JDE). The JDE network training is considered as a multi-task learning problem and an automatic balance loss is applied to balance the importance of classification, regression and ReID within the network. In,[48] the authors proposed to exploit the bounding box regression of an object detector to predict the position of an object in the next frame, thereby converting a detector into a tracker. Zhang et al.[49] proposed the FairMOT algorithm based on the anchor-free object detection algorithm CenterNet.[50] Zhou et al.[51] proposed a simultaneous detection and tracking algorithm called CenterTrack by applying detection to a pair of images and predicting targets associations within these images. Subsequently, Tokmakov et al.[52] proposed the PermaTrack algorithm, an extension of CenterTrack to videos of arbitrary length. Recently, Transformer-based methods such as TransTrack[53] or TrackFormer[54] have been proposed.

3. PROPOSED METHOD

3.1 Drone detection using YOLOv5

Drone detection presents a complex problem due to various factors, including the diversity of drone types, sizes, shapes, and flight dynamics. Real-world scenarios introduce additional challenges, such as occlusions, cluttered backgrounds, changing lighting conditions, and other moving objects such as birds. Drone detection systems need to operate in real-time and provide accurate and reliable results to enable timely decision-making and response.

The YOLOv5 algorithm has particularly been proved well-suited for drone detection, thanks to its real-time performance, high accuracy, and adaptability. Given an input image $I \in \mathbb{R}^{H\times W\times 3}$, where H and W represent the height and width of the image respectively, the YOLOv5 model processes it through a series of convolutional and pooling layers to generate an intermediate feature map $F \in \mathbb{R}^{S\times S\times D}$. The model's output tensor $O \in \mathbb{R}^{S\times S\times(B\times 5+C)}$ is obtained from the feature map, where S is the grid size, B denotes the number of bounding boxes per grid cell, D is the depth of the feature map, and C refers to the number of classes.

The tensor elements encode the bounding box coordinates and objectness scores as :

$$(x, y, w, h, o) = (b_x, b_y, b_w, b_h, b_o) \tag{1}$$

with (b_x, b_y) indicating the box's center relative to the grid cell, (b_w, b_h) representing the box's width and height relative to the entire image, and b_o being the objectness score. The class probabilities are denoted as $P(c_l)$, with $l = 1, 2, \ldots, C$.

To train the YOLOv5 model for drone detection, a labeled dataset $D = (I_n, Y_n)_{n=1}^{N}$ containing images I_n with diverse UAVs types, angles, distances, and conditions is required. The ground-truth label Y_n is represented as a tensor with the same dimensions as the model output. The training process aims to minimize the loss function:

$$L_{\text{total}} = \lambda_{\text{loc}} \cdot L_{\text{loc}} + \lambda_{\text{obj}} \cdot L_{\text{obj}} + \lambda_{\text{cls}} \cdot L_{\text{cls}}$$

where the losses L_{loc}, L_{obj}, and L_{cls} correspond to localization, objectness, and classification, respectively, and their respective weights are λ_{loc}, λ_{obj}, and λ_{cls}.

After training, the model can be used for real-time drone detection and localization as it is a one-stage detector. The predicted bounding boxes and class probabilities are processed using a threshold T and non-maximum suppression (NMS) with an Intersection over Union (IoU) threshold τ to generate the final set of detected drones $D = (b_j, c_j, \mathbf{f}_j)_{j=1}^{M}$, where b_j is the j-th predicted bounding box, c_j is its corresponding class label and $\mathbf{f}_j$ the corresponding appearance object descriptor.

3.2 Measurement and target motion model in spherical frame

As it was explained, MOT algorithms on video sequences is traditionally realised in frame reference to estimate pixel location, speed, acceleration of each targets with the same ID. Our system provides frame with meta-data, in the manner that each frame contains camera orientations, zoom, focal at current time. In this way we make the choice to convert each detection in image reference to sensor frame (*i.e.* in spherical coordinates more precisely in the focal plane of the cameras).

Starting from detection module presented in previous part 3.1, the tracking module receive a set $Z(k)$ of m_k measurements corresponding to the detection list in the frame at time t_k.

$$Z(k) = \{\mathbf{z}_k^1, \ldots, \mathbf{z}_k^{m_k}\} \tag{2}$$

where each measurement is expressed in sensor frame ($\forall j = 1, \ldots, m_k$) by :

$$\mathbf{z}_k^j = h_k(\mathbf{x}_k^i) + \mathbf{b}_k^j \tag{3}$$

with $\mathbf{z}_k^j = [\theta_k^j, \phi_k^j]'$, $\mathbf{x}_k^i$ is the target state at current time, $h_k(\cdot)$ is the observation function at current time t_k and $\mathbf{b}_k^j$ is a Gaussian white noise process with a covariance $\mathbf{R}_k^j$.

Here, we don't introduce in the observation vector the aspect ratio r_k^j of the bounding box with height g_k^j as it is done in.[7] Most of popular video tracker introduce bounding box in the MOT and it is right if the tracking is implemented in image frame. Nevertheless, our MOT is implemented in sensor frame (*i.e.* in spherical coordinates) and we have observed decrease performance if we add in the observation state $\mathbf{z}_j(k)$ the bounding box measurement (r_k^j, g_k^j). The reasons seem to be the lack of physical sense in modeling bounding box motion in spherical coordinates and the dynamic state vector's inability to discriminate track associations in cases of occlusion. So we only try to estimate the state $\mathbf{x}_k^i = (\theta_k^i, \dot{\theta}_k^i, \phi_k, \dot{\phi}_k^i)'$ of a track i among n_k tracks in location and velocity by only considering the measurement $\mathbf{z}_k^j$ in spherical space.

For the moment, we consider a constant velocity motion model :

$$\mathbf{x}_k^i = \mathbf{F}_k \mathbf{x}_{k-1}^i + \nu_k \tag{4}$$

where $\mathbf{F}_k$ is the transition matrix of the constant velocity motion model, ν_k is the white Gaussian noise of covariance $\mathbf{Q}_k$.

At each frame k, we consider a track $\mathcal{T}^{k,i}$, $i = 1, \ldots, n_k$, representing the current estimated state $\hat{\mathbf{x}}_{k|k}^i$ and its associated covariance $\mathbf{P}_{k|k}^i$ obtained with standard Kalman. In a MOT approach, we need to evaluate track association hypotheses of a track $\mathcal{T}^{k-1,i}$ with a measurement $\mathbf{z}_k^j$. By using Gaussian assumption, we usually express the cost association with the likelihood $\lambda_i^j(k)$ by :

$$\lambda_i^j(k) = (1 - p_D)^{1-u_j} \times p_D p(\mathbf{z}_k^j | \mathbf{x}_k^i)^{u_j} \tag{5}$$

where p_D is the detection probability, $p(\mathbf{z}_k^j | \mathbf{x}_k^i)$ the normal probability density function and u_j defined by :

$$u_{l,0} = \begin{cases} 1, \text{ for } j = \{1, \ldots, m_k\} \\ 0, \text{ if } j = 0 \end{cases} \tag{6}$$

Considering the deep detection processing presented in 3.1, we define the augmented measurement set $Z(k)^*$ where each augmented measurement $\mathbf{z}_k^{j*}$ are expressed by :

$$\mathbf{z}_k^{j^*} = \{\mathbf{z}_k^j, \mathbf{f}_k^j, r_k^j, g_k^j\}, j = 1, \ldots, m_k \tag{7}$$

For a track $\mathcal{T}^{k,i}$, $i = 1, \ldots, n_k$, estimated state $\hat{\mathbf{x}}_{k|k}^i$ and covariance $\mathbf{P}_{k|k}^i$ are obtained with a Kalman filter, estimated feature vector $\hat{\mathbf{f}}_k^i$ is updated in an exponential moving average (EMA) manner as follows :

$$\hat{\mathbf{f}}_k^i = \alpha \hat{\mathbf{f}}_{k-1}^i + (1 - \alpha) \mathbf{f}_k^j \tag{8}$$

The vector $\hat{\mathbf{f}}_k^i$ is the appearance embedding of the current matched detection $\mathbf{z}_k^{j^*}$. The EMA updating strategy leverages the information of inter-frame feature changes and can depress detection noise. Estimated box size is obtained according a simple averaging process on the last s frames :

$$\hat{r}_k^i = \frac{\sum_{t=k-s}^{k} r_t^{j(t)}}{s}, \hat{g}_k^i = \frac{\sum_{t=k-s}^{k} g_t^{j(t)}}{s} \tag{9}$$

Finally, track estimation processing outputs n_k tracks $\mathcal{T}^{k,i}$, $i = 1, \ldots, n_k$ with updated information in angular location, angular velocity, feature and object size in pixel. The track $\mathcal{T}^{k,i}$, $i = 1, \ldots, n_k$ is sequentially defined by: $\mathcal{T}^{k,i} = \{\hat{\mathbf{x}}_{k|k}^i, \mathbf{P}_{k|k}^i, \hat{\mathbf{f}}_k^i, \hat{r}_k^i, \hat{g}_k^i, \mathcal{T}^{k-1,i}\}$.

Unfortunately, track association ambiguities arises due to a lot of measurements generated by camera movement, contrast, context *etc.*.To palliate track association problem two algorithms are presented in next sub-parts.

3.3 Multiple Object Tracking

3.3.1 Global Nearest Neighbor Standard Filter (GNNSF)

In the subsection, we consider m_k measures issued from the detector, and a list of n_k tracks at time t_k. In order to associate a track to a measurement and update the state, we compute the association matrix $\mathbf{A}_k \in \mathrm{R}^{n_k \times m_k}$ where each components $a_k^{i,j}$ represents association between a track i, $i = 1, \ldots, n_k$, with a measure j, $j = 1, \ldots, m_k$, in the manner that:

$$a_k^{i,j} = \begin{cases} 1 \text{ if association between track } i \text{ with measure } j \text{ is possible} \\ 0 \text{ if not} \end{cases} \tag{10}$$

In the same way, we define $\mathbf{C}_k \in \mathrm{R}^{n_k \times m_k}$ the association cost matrix $c_k^{i,j}$, where each component represents the association cost between a track i, $i = 1, \dots, n_k$ with measure j, $j = 1, \dots, m_k$ at current time t_k. Usually, the association cost is based on the negative likelihood logarithm defined in (5).

$$c_k^{i,j} = -log(\lambda_i^j(k)) \tag{11}$$

The objective in MOT is to obtain the optimal association of the n_k tracks to m_k measures in order to minimize the global association cost. This can be reformulated as the following assignment problem :

$$\min_{\mathbf{X}^*} \sum_{i=1}^{n_k} \sum_{j=0}^{m_k} a_k^{i,j} c_k^{i,j} \tag{12}$$

under constraints :

$$\sum_{j=0}^{m_k} a_k^{i,j} = 1 \tag{13}$$

$$\sum_{i=0}^{n_k} a_k^{i,j} = 1 \tag{14}$$

where $\mathbf{X}^*$ represents the optimal binary association matrix.

To take into account the fact that some measurements may come from false alarms, or from missed detection, a dummy track is added to the set of existing tracks ($i = 0$), In this way, the dimension of matrix $\mathbf{C}_k$ is extended to $(n_k + 1) \times (m_k + 1)$. The inputs in the $(n_k + 1)^{th}$ row represents the association cost of tracks where measurements are considered as false alarms. We make the assumption that tracks cannot be associated to false alarm. Missed detection are defined after the gating step defined in 3.4. In this paper, to compute track association hypothesis, we use a classical Mahalanobis distance $d^1_{(\cdot,\cdot)}$ between predicted track state $\hat{\mathbf{x}}^i_{k|k-1}$, $i = 1, \dots, n_k$, with measurement $\mathbf{z}_k^j$, $j = 1, \dots, m_k$. To obtain the matrix $\mathbf{X}^*$, we use Munkres algorithm. This approach is known as global nearest neighbor standard filter (GNNSF). In situations with a lot of clutter, false alarms, missed detection, and occlusions, the performance of these algorithms will deteriorate significantly. Indeed, it may be advantageous to instead use a sliding window of previous and/or future track states to construct assignment costs that model the relationship between tracks and new sensor measurements more accurately.

3.3.2 S-D assignment

Another way to improve track association is to increase the association gate $\mathcal{S} > 2$ to compute optimal data association integrated on several frames. Let Γ, the measurement partition set of $\mathcal{S}_k$. We want to compute the optimal partition $\gamma^* \in \Gamma$ who minimize the global cost on $\mathcal{S}_k$. All track hypothesis are built for each partition of Γ. Then, we define a partition γ^i as a particular partition of Γ representing a track $\mathcal{T}^{k,i}$. The track is sequentially associated to a particular sequence of measurements $\gamma^i = \{\mathbf{z}_{k-\mathcal{S}}^{j(1)}, \mathbf{z}_{k-\mathcal{S}-1}^{j(2)}, \dots, \mathbf{z}_k^{j(\mathcal{S})}\}$.

For the track $\mathcal{T}^{k,i}$, we define a cost function named $c_k^{i,j(1),j(2),\dots,j(k)} \in \mathbb{R}$ and an association function $a_k^{i,j(1),j(2),\dots,j(k)} \in \{0, 1\}$. The association function represents which measurement the track is associated with.

As in the subpart 3.3.1, we are looking for the optimal association minimizing the global association cost on the $\mathcal{S}$ last frames. On the other hand, it consists to compute the optimal partition γ^* defined by :

$$\min_{\gamma^* \in \Gamma} \sum_{j_1=0}^{m_{k-\mathcal{S}}} \dots \sum_{j_k=0}^{m_k} a_k^{i,j_1,\dots,j_k} c_k^{i,j_1,\dots,j_k} \tag{15}$$

under the constraints :

$$\sum_{j_2=0}^{m_{k-\mathcal{S}-1}} \dots \sum_{j_k=0}^{m_k} a_k^{i,j_1,\dots,j_k} = 1 \qquad \forall j_1 = 0, \dots, m_{k-\mathcal{S}} \tag{16}$$

$$\sum_{j_1=0}^{m_{k-\mathcal{S}}} \sum_{j_3=0}^{m_{k-\mathcal{S}-2}} \dots \sum_{j_k=0}^{m_k} a_k^{i,j_1,\dots,j_k} = 1 \qquad \forall j_2 = 0, \dots, m_{k-\mathcal{S}-1} \tag{17}$$

$$\vdots \qquad \vdots \tag{18}$$

$$\sum_{j_1=0}^{m_{k-\mathcal{S}}} \dots \sum_{j_{k-1}=0}^{m_{k-1}} a_k^{i,j_1,\dots,j_k} = 1 \qquad \forall j_k = 0, \dots, m_k \tag{19}$$

$$\tag{20}$$

We add a fictive measurement 0 in the detection set corresponding to the non-association track assumption due to false alarms hypothesis or non-detection of the tracked object. The process to compute the optimal association is executed on a sliding window of size $\mathcal{S}$. It implies the holding of track association hypotheses during the $\mathcal{S}$ frames. Despite the holding of all hypotheses, only the optimal association is considered in the track evaluation.

Instead of expensive global data association, some have proposed heuristics involving solving a search cascade for local optimum. It's the case of StrongSORT[7] who has gained in popularity due to its real-time speed and effective use of association features to achieve high-quality tracking. In the next part, we adapt GNNSF and SD-Assignment algorithms with similar deep approach to improve track association.

3.4 Joint tracking and classification

In the detection step describes in 3.1 each measurement is enhanced with feature descriptors of the detected object. After extracting the feature descriptor using the neural network, this value is saved for all tracked objects, and is updated continuously (8). In this way, we introduce a similarity criteria between all pairs of objects (predicted and detected) calculated using the cosine similarity measure.[55] The cosine similarity measure is a method to determine how similar two vectors are. From a mathematical perspective a cosine similarity function computes the cosine of the angle between two vectors in a multi-dimensional space, The cosine similarity $d^2_{(\cdot,\cdot)}$ is at its most when the angle is closer to zero. In this manner, when we receive the set $Z(k)^*$, we use both the Mahalanobis distance $d^1_{(\cdot,\cdot)}$ issued from (5) and the cosine similarity $d^2_{(\cdot,\cdot)}$ to validate the compatibility between predicted track $\mathcal{T}^{i,k}$, for $i = 1, \dots, n_k$, with measurement $\mathbf{z}_k^{j*}$, for $j = 1, \dots, m_k$. We speak about deep tracker.

The steps of our proposed deep tracker (figure 1) are the following :

1. gating. Use distances to validate the association between a track $\mathcal{T}^{i,k}$ and a measurement $\mathbf{z}_k^{j*}$:

$$a_k^{i,j} = \mathbb{1}_{(d^1_{(\hat{\mathbf{x}}^i_{k|k-1}, \mathbf{z}^j_k)} \leq \delta)} \mathbb{1}_{(d^2_{(\hat{\mathbf{f}}^i_{k|k-1}, \mathbf{f}^j_k)} \leq \alpha)} \tag{21}$$

   where δ is obtained according Chi2 table and $\alpha \in [0, 1]$ is threshold similarity criterion.

2. clustering. Use all tracks interactions to group tracks in order to subdivide the association computation;

3. track association. In each cluster, for all validated association, compute the deep association cost $c_k^{i,j}$:

$$c_k^{i,j} = -log(\lambda_i^j(k)^*) \tag{22}$$

   where the deep likelihood is defined by:

$$\lambda_i^j(k)^* = (1 - p_D)^{1-u_j} \times p_D(p(\mathbf{z}_k^j | \mathbf{x}_k^i) d^2_{(\hat{\mathbf{f}}^i_{k|k-1}, \mathbf{f}^j_k)})^{u_j} \tag{23}$$

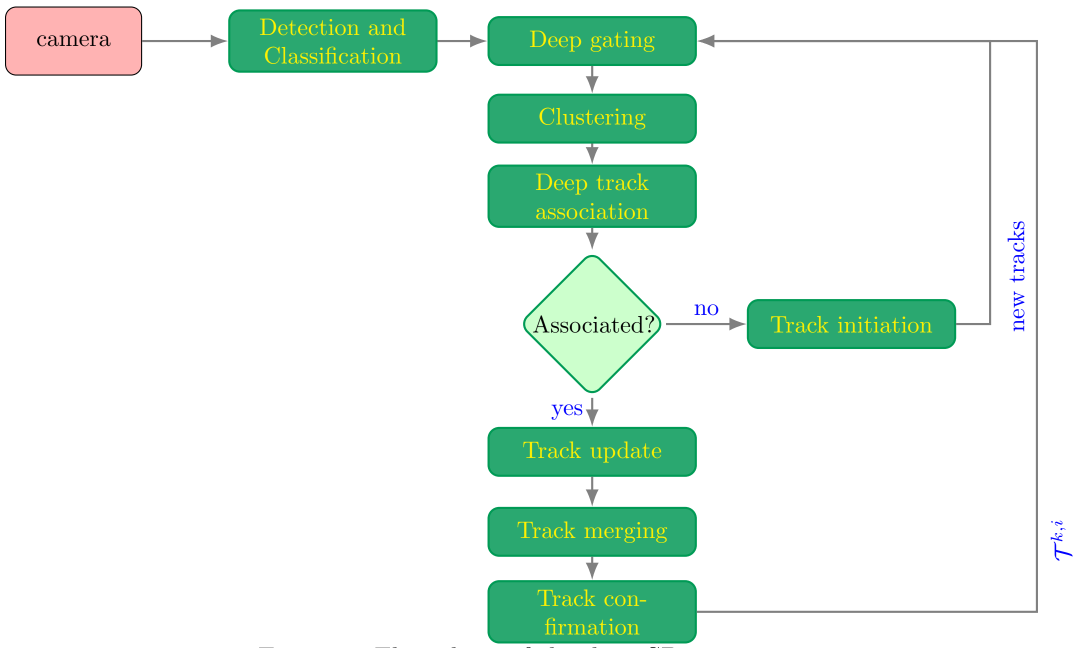


Figure 1: Flow chart of the deep SD-assignment.

4. compute best track association. From the cost association matrix, we compute in each cluster the optimal association (15) on a sliding window $\mathcal{S}$;
5. track estimation. For each track, we estimate the target state, feature and box size of the track $\mathcal{T}^{i,k}$;
6. track initiation. Non-associated measurement are used to initialise new tracks;
7. track merging. Closest tracks are merged according to the following criteria :

$$\frac{\sum_{t=k-\mathcal{S}}^{k} d^1_{(\hat{\mathbf{x}}^i_{t|t-1}, \mathbf{z}^{j(t)}_t)}}{\mathcal{S}} \leq \delta \tag{24}$$

The deep association differs to usual association by considering feature distance in tracking association processing. Both metrics complement each other by serving different aspects of the assignment problem. On the one hand, the Mahalanobis distance provides information about possible object locations based on motion that are particularly useful for short-term predictions. On the other hand, the cosine distance considers appearance information that are particularly useful to recover identities after long term occlusions,or when the camera movement is quick. To smooth effect of cosine distance, we introduce a weighted sum in (22) :

$$c^{i,j}_k = -(1-u_j) \cdot log(1-p_D) - u_j \cdot (w_1 \cdot log(p_D p(\mathbf{z}^j_k|\mathbf{x}^i_k)) + w_2 \cdot log(d^2_{(\hat{\mathbf{f}}^i_{k|k-1}, \mathbf{f}^j_k)}) \tag{25}$$

where $w_1 + w_2 = 1$.

To address the problem of SD-assignment with $\mathcal{S} > 2$ the cost function is concatenated with the previous association cost obtained along the track $\mathcal{T}^{k,i}$:

$$c^{i,j_1,\ldots,j_k}_k = \prod_{l=k-\mathcal{S}}^{k} c^{i,j(l)}_l \tag{26}$$

The flowchart of the algorithm is given in figure 1. We have selected interested sequences in next part to test and compare Deep-GNNSF and Deep-SD-Assignment with StrongSORT.

4. EXPERIMENTS

4.1 System

A large dataset has been collected to accomplish the necessary training of the detector and classifier thanks to a cooled thermal camera (Band II, MWIR, reference CAMIR/Z15-690 HD) and a color daylight camera (reference CAMEZOOM-FHD/Z35-350), embedded on a two-axis turret from Exavision as can be seen in the figure 2. Both cameras and the turret were developed by Exavision, a company specializing in complex optronic solutions for field operations in harsh environments. The thermal infrared videos have a resolution of 1280×720 pixels while the visible videos have a resolution of 1920×1080 pixels with an image refresh rate of 25 frames per second.

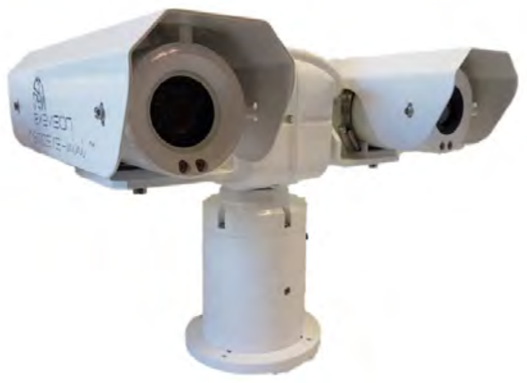


Figure 2: Exavision turret with Visible and Infrared cameras

In addition to being able to record video streams and extract images from them, we can also get some information on the cameras like the azimuth angle, the elevation angle or even the field of view. These information are critical as we can take into account the camera's movement and improve our tracking thanks to them. At the time of the experiments, we could only get such information on the visible camera and we saved those as meta-data in the frames. Hence why we only present our result on the visible stream even if we can extend our results to the infrared stream.

4.2 Dataset

We have tested our system in field in order to stress the performance under different conditions with the flights of several UAVs like the Phantom 4 Pro V2 and the Mavic 2 Zoom. Let us present two collected video sequences on which we evaluated the performance of our solution and compared it to the StrongSORT algorithm. Note that sequences does not contain continuous frames, some are missing. First sequence contains the flight of two simultaneous UAVs, with different sky backgrounds and distances to the camera (figure 3). As the turret had rather smooth movements to follow both UAVs resulting in an interesting but not that hard video sequence. While the second sequence presents more difficulties for the detection and the tracking of the UAV. The UAV sometimes is off-screen and with different sizes, flew at higher speeds in the sky but also behind and in front of vegetation making the tracking much more challenging and more discriminating between different tracking approaches (figure 4).

To perform annotation tasks on the data collected we used the tool CVAT (Computer Vision Annotation Tool). It's a simple tool with a good amount of functionalities in addition to supporting several annotation formats. This step allowed us to prepare our large dataset in order to train our detection models as well as to build the ground truth files. These will serve as reference points in the study of the results obtained with different approaches. A ground truth file contains the history of the positions of a given object through the frames extracted from a video sequence. It is built as a succession of lines such as :

frame_index, track_id, box_x_top_left, box_y_top_left, box_width, box_height, class_id, unused, unused

(a) frame002189.

(b) frame002343.

(c) frame003171.

(d) frame003230.

(e) frame003538.

(f) frame003738.

(g) frame004664.

(h) frame005477.

(i) frame006416.

Figure 3: Sequence 1 multi-UAVs

where $track\_id$ is a unique integer for a specific object which can be seen in the video sequence, $frame\_index$ indicates on which frame we are working, both $box\_x\_top\_left$ and $box\_y\_top\_left$ are the box top-left coordinates and combined with box_width and box_height it precisely identifies the position of the object in the frame. Finally, $class\_id$ helps us to categorize whether the object is a UAV, a human, a bird or something else, and last two columns are unused in this work.

4.3 Parameters and evaluation metrics

Concerning StrongSORT, evaluation on sequence were carried out with the default algorithm parameters : the maximum number of missed detections before a track is deleted is 30, the number of consecutive detections before the track is confirmed is 3, the maximum size of the appearance descriptor gallery is 100, the EMA factor is $\alpha = 0.9$ and the Kalman filter parameters also remain the default ones. Being in scenarios wherein the targets and the camera can move quickly, we have modified the StrongSORT primary data association procedure which is based on IoU metric in order to not bias the results. Instead, we compute the euclidian distance $De_k^{i,j}$ between the track i, $i = 1, \dots, n_k$ and the measure j, $j = 1, \dots, m_k$ at current time t_k in spherical coordinates. The association is then computed in the manner that:

$$\begin{array}{ll} \text{if } De_k^{i,j} \leq T_{D_e} & \text{Track } i \text{ associated to measure } j \\ \text{if } De_k^{i,j} > T_{D_e} & \text{Track } i \text{ not associated to measure } j \end{array}$$

where $T_{D_e} = 1°$. Note that the original StrongSORT threshold for IoU metric is 0.2.

Concerning Deep-GNNSF and Deep-SDA, evaluation on sequence were carried out with following parameters : the noise process in the motion model (4) is fixed at $2 \cdot s^{-2}$. The sample time corresponds to the real sample time between two frames, in (8), the EMA factor is $\alpha = 0.9$ and the size of the sliding window for the box estimation (9) is $s = 10$. The distinction between Deep-GNNSF and Deep-SDA lies on the value $\mathcal{S}$, respectively

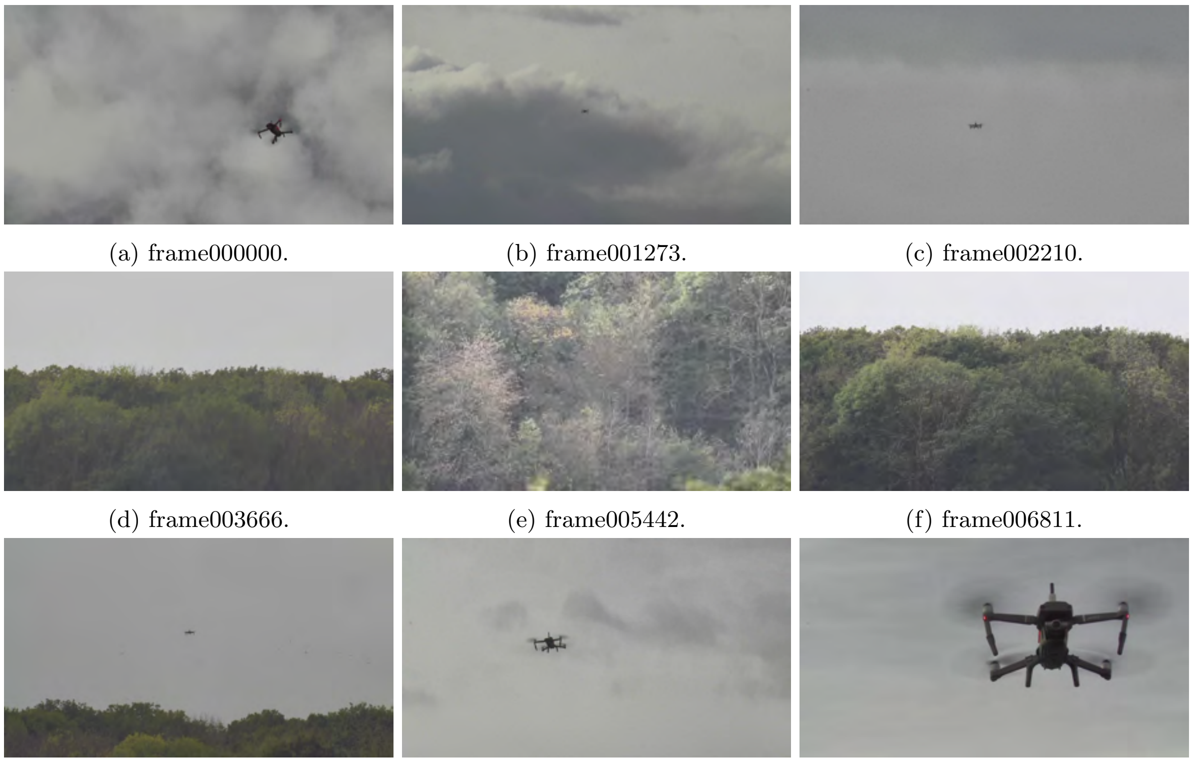

(a) frame000000. (b) frame001273. (c) frame002210.
(d) frame003666. (e) frame005442. (f) frame006811.
(g) frame007652. (h) frame010036. (i) frame011545.

Figure 4: Sequence 2 UAV with vegetation

with $\mathcal{S} = 2$ and with $\mathcal{S} = 3$. Gating parameters for track association in (21) are obtained according to the Chi2 table $\delta = 5.991$ and similarity distance $\alpha = 0.6$.

Evaluation is carried out according to the following metrics [*]:

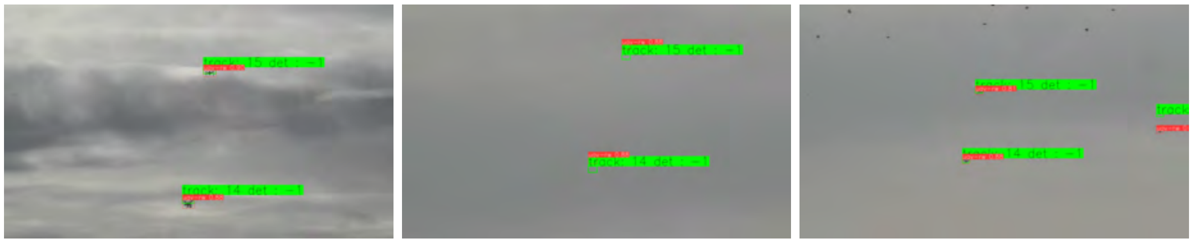
- Identification Metrics (IDF1) : ratio of correctly identified detections over the average number of ground-truth and computed detections;
- Mostly tracked (MT): percentage of ground-truth tracks that have the same label for at least 80% of their life span;
- Mostly lost(ML): percentage of ground-truth tracks that are tracked for at most 20% of their life span;
- False Negative (FN) : number of times when ground truth exists but prediction was missed;
- False Positive (FP) : number of times where tracker prediction exists for no ground truth track;
- Identity switches (IDs): number of times the reported identity of a ground-truth track changes;
- Fragmentation (FM): number of times a track is interrupted by a missing detection;
- Multi-object tracking accuracy (MOTA): summary of overall tracking accuracy in terms of false positives, false negatives and identity switches;
- Multi-object tracking precision (MOTP): summary of overall tracking precision in terms of pixel distance between track and ground truth.

[*] https://github.com/cheind/py-motmetrics

4.4 Results

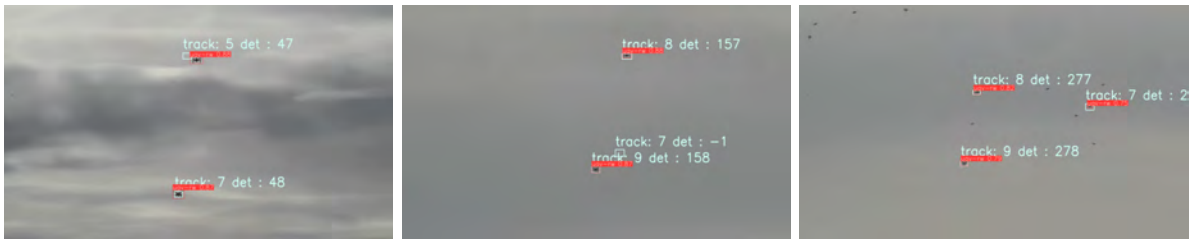
4.4.1 Results on sequence 1

We test and compare performances of our approach on the sequence 1. We evaluate tracking performances of StrongSORT, Deep-GNNSF and Deep-SD Assignment with moving camera. On figure 5, we observe the results of StrongSORT to track several UAVs. The algorithm maintains the track continuity and we do not observe wrong association despite the flock of birds at the end of the sequence. On the contrary for the Deep-GNNSF algorithm (figure 6) we observe a wrong association due to camera movement and the flock of birds. This is reinforced by numerical results presented in table 1, the False Positive number and False Negative Number are higher than for the StronSORT algorithm which explains lower performances on MOTA and MOTP. However, if look into Deep-SD assignment performances (figure 7) we observe the maintain of the tracks and track ID comparing to the Deep-GNNSF. This is due to the increase of the time decision process $\mathcal{S}$ in the track association step. It explains the better numerical performances in table 1. If we compare StrongSORT with Deep-SD Assignment, the IDF1 is better with StrongSORT. A high IDF1 score estimates the total number of unique objects in a scene than giving information about good detection or association. But the tracks with Deep-SDA are never lost, the difference lives in the track precision and the track association with ground-truth for the evaluation process.Overall, the results suggest that StrongSORT and Deep-SD Assignment are better suited for tracking multiple UAVs in the presence of a flock of birds, while Deep-GNNSF is not adapted to maintain track continuity in this context.



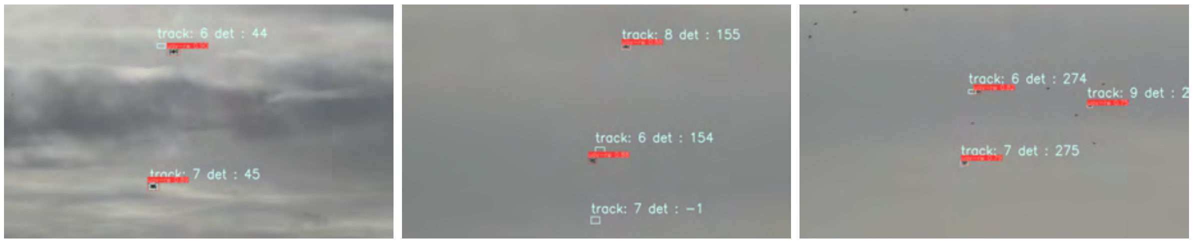
(a) frame002589. (b) frame004251. (c) frame006274.

Figure 5: Results on sequence 1 with StrongSORT algorithm.



(a) frame002589. (b) frame004237. (c) frame006274.

Figure 6: Results on sequence 1 with Deep-GNNSF algorithm.



(a) frame002574. (b) frame004251. (c) frame006274.

Figure 7: Results on sequence 1 with Deep-SDA algorithm.

Table 1: Results on sequence 1.

Algo.	IDF1	GT	MT	ML	FP	FN	IDs	FM	MOTA	MOTP
StrongSORT	**90.0%**	2	2	0	1	1	**0**	0	**80.0%**	25.58 pix
Deep-GNNSF	52.2%	2	2	0	5	2	2	2	10.0%	28.02 pix
Deep-SDA	75.0%	2	2	0	0	0	2	0	75.0%	**24.29 pix**

4.4.2 Results on sequence 2

In this sequence we are following one drone with pop-up movement with weak contrast. As previously, Deep-GNNSF lost its track 7 associated to the UAV. During the first pop-up movement the YOLO detector detects the UAV with few false alarms. The UAV doesn't maneuver in the vegetable background and the track is maintain. But in the second pop-up movement the drone maneuvers in the vegetable background and the Deep-GNNSF associate its track with false alarms bringing about initialisation of new track. For StrongSORT algorithm (figure 8), the track is not lost immediately but after several frames because of the high number of false tracks.

On sequence 2 results are quite different. The pop-up movement of the drone and the weak contrast of the object pose some challenges for the tracking algorithms. While the YOLOv5 detector performs well in detecting the UAV in the first pop-up movement with few false alarms, Deep-GNNSF loses its track 7 (9d) associated with the UAV in the second pop-up movement as it associates its track with false alarms. On the other hand, the StrongSORT algorithm maintains the track initially but loses it after several frames due to the high number of false tracks (8). These observations suggest that the tracking algorithms need to be further improved to handle challenging situations such as pop-up movements and weak contrast objects. However the sliding association window in Deep-SD Assignment (10) allows the algorithm to handle track continuity even in cases of occlusion or temporary loss of detection. This is because the algorithm considers multiple hypotheses for track association over a longer period of time, rather than just the current frame, and thus can make more robust decisions. With the Deep-SDAssignment, our implementation runs at approximately 11 Hz with roughly half of the time spent on feature generation. Therefore, given a modern GPU and C++ implementation, the system will remain computationally efficient and will operate at real time.

Table 2: Results on sequence 2.

Algo.	IDF1	GT	MT	ML	FP	FN	IDs	FM	MOTA	MOTP
StrongSORT	30.4%	2	1	1	103	24	18	10	58.1%	119.69 pix
Deep-GNNSF	34.4%	2	2	0	69	29	5	11	71.5%	84.14 pix
Deep-SDA	**87.2%**	2	1	0	75	25	**1**	13	**69.9%**	**78.12 pix**

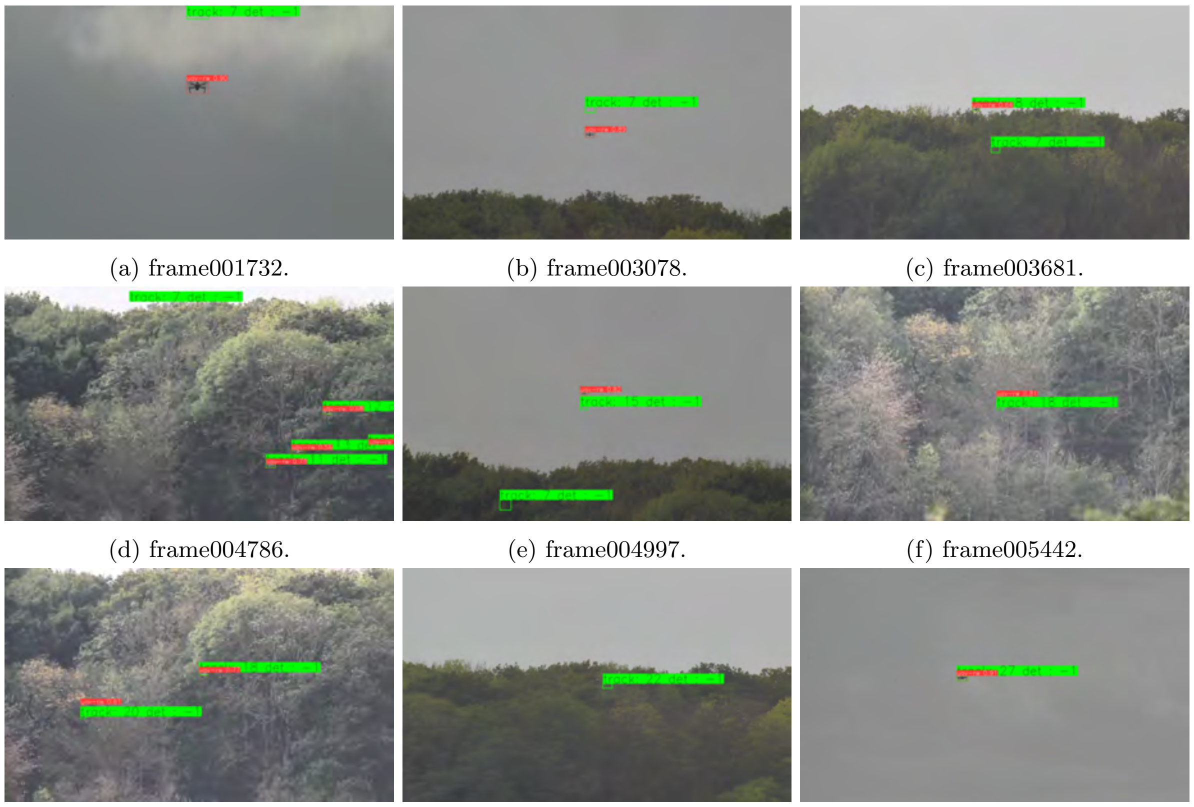
5. CONCLUSION

In this paper, we have presented a comparative study on joint tracking and classification algorithms with online state-of-the-art algorithm StrongSORT. Our particular context lies in the need to track UAVs in the spherical frame and not in image frame. Then usual assumption on state vector or track association are challenged with our approach. Obtained results seem to prove better performances with Deep-SD Assignment algorithm in presence of false alarms or low contrast. The counterpart is the decrease in time processing. Future work should prove this claim and evaluate the impact of the Deep-Assignment in our data fusion system.

6. ACKNOWLEDGMENTS

Authors thank EXAVISION company [†] for their help, contribution and support in order to achieve this research project.

[†]https://exavision.com


(a) frame001732. (b) frame003078. (c) frame003681.

(d) frame004786. (e) frame004997. (f) frame005442.

(g) frame006147. (h) frame007143. (i) frame008359.

Figure 8: Results on sequence 2 with StrongSORT algorithm.

REFERENCES

[1] Mohsan, S., Khan, M., Noor, F., Ullah, I., and Alsharif, M., "Towards the unmanned aerial vehicles (uavs): A comprehensive review.," *journal of Drones* **6**(6) (2022).

[2] Choi, Y., "Security threat scenarios of drones and anti-drone technology," *Academy of Strategic Management Journal* **21**(1), 1–7 (2022).

[3] Castrillo, V. U., Manco, A., Pascarella, D., and Gigante, G., "A review of counter-uas technologies for cooperative defensive teams of drones," *journal of Drones* **6**(3) (2022).

[4] Park, S., Kim, H. T., Lee, S., Joo, H., and Kim, H., "Survey on anti-drone systems: Components, designs, and challenges," *IEEE Access* **9**, 42635–42659 (2021).

[5] Svanstrom, F., Alonso-Fernandez, F., and Englund, C., "Drone detection and tracking in real-time by fusion of different sensing modalities," *journal of Drones* **6**(11) (2022).

[6] Guvenc, I., Koohifar, F., Singh, S., Sichitiu, M. L., and Matolak, D., "Detection, tracking, and interdiction for amateur drones," *IEEE Communications Magazine* **56**(4), 75–81 (2018).

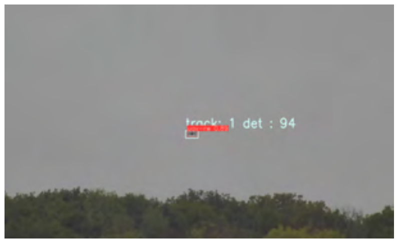
[7] Wojke, N., Bewley, A., and Paulus, D., "Simple online and realtime tracking with a deep association metric," *CoRR* **abs/1703.07402** (2017).

[8] Yu, F., Li, W., Li, Q., Liu, Y., Shi, X., and Yan, J., "POI: multiple object tracking with high performance detection and appearance feature," *CoRR* **abs/1610.06136** (2016).

[9] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B., "Simple online and realtime tracking," *CoRR* **abs/1602.00763** (2016).

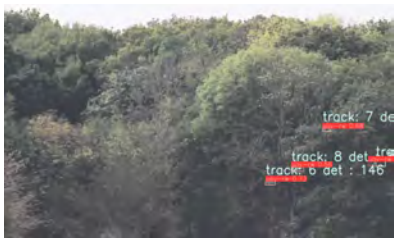
[10] Kim, C., Li, F., Ciptadi, A., and Rehg, J. M., "Multiple hypothesis tracking revisited," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (December 2015).

(a) frame001744.

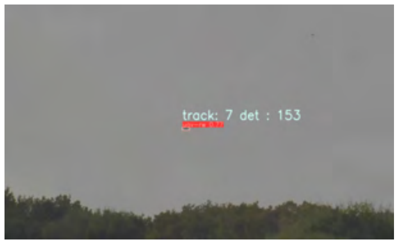


(b) frame003066.

(c) frame003692.



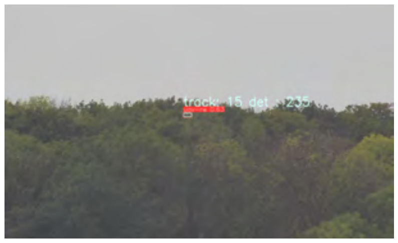
(d) frame004786.



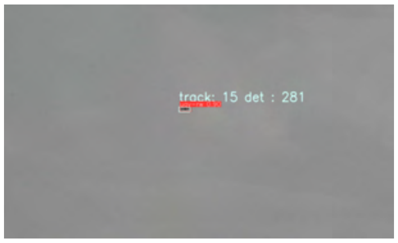
(e) frame005010.

(f) frame005442.

(g) frame006147.



(h) frame007159.



(i) frame008346.

Figure 9: Results on sequence 2 with Deep-GNNSF algorithm.

[11] Ur-Rehman, A., Naqvi, S. M., Mihaylova, L., and Chambers, J. A., "Multi-target tracking and occlusion handling with learned variational bayesian clusters and a social force model," *IEEE Transactions on Signal Processing* **64**(5), 1320–1335 (2016).

[12] Song, Y.-M., Yoon, K., Yoon, Y.-C., Yow, K. C., and Jeon, M., "Online multi-object tracking with gmphd filter and occlusion group management," *IEEE Access* **7**, 165103–165121 (2019).

[13] Kaur, R. and Singh, S., "A comprehensive review of object detection with deep learning," *Digital Signal Processing* **132**, 103812 (2022).

[14] Bashar, M., Islam, S., Hussain, K. K., Hasan, M. B., Rahman, A. B. M. A., and Kabir, M. H., "Multiple object tracking in recent times: A literature review," (2022).

[15] Viola, P. and Jones, M., "Rapid object detection using a boosted cascade of simple features," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001* **1**, I–I (2001).

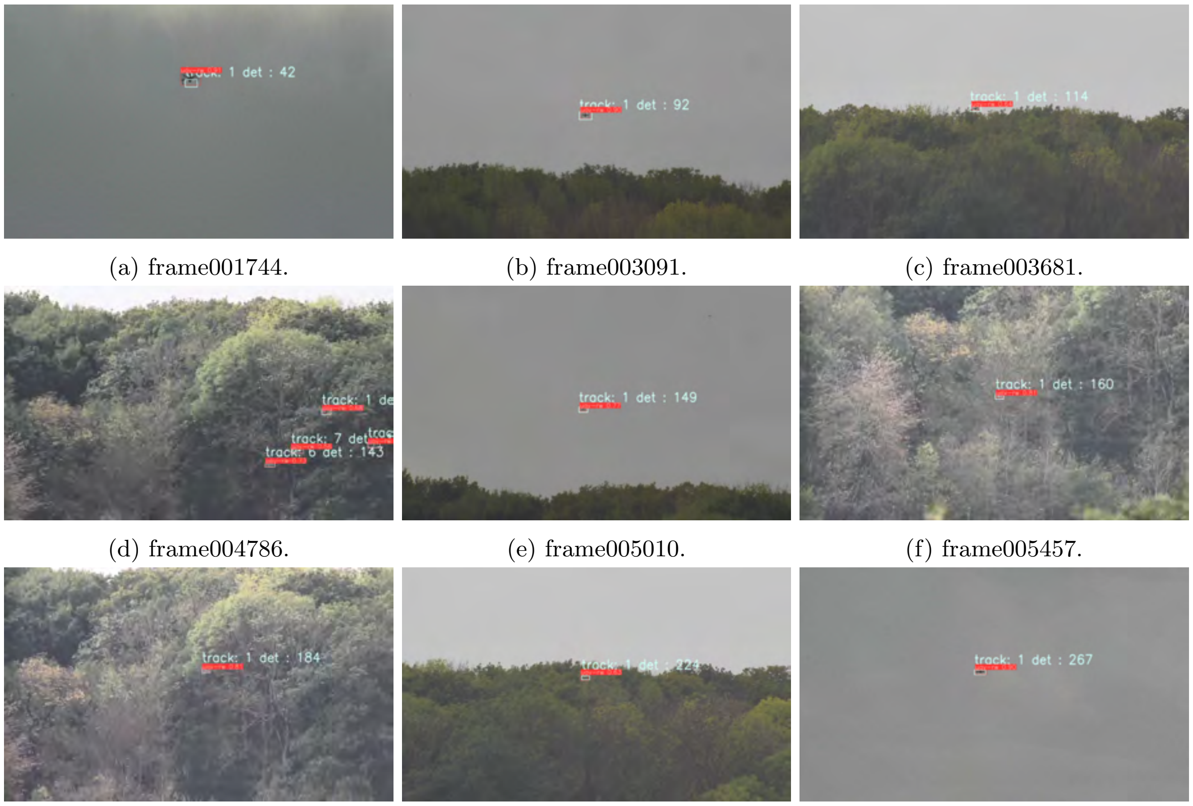
[16] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision* **60**, 91–110 (Nov. 2004).

[17] Girshick, R., "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)* , 1440–1448 (2015).

[18] Ren, S., He, K., Girshick, R., and Sun, J., "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis ; Machine Intelligence* **39**, 1137–1149 (jun 2017).

[19] He, K., Gkioxari, G., Dollár, P., and Girshick, R. B., "Mask R-CNN," *CoRR* **abs/1703.06870** (2017).

[20] Girshick, R., Donahue, J., Darrell, T., and Malik, J., "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* , 580–587 (jun 2014).



(a) frame001744. (b) frame003091. (c) frame003681.

(d) frame004786. (e) frame005010. (f) frame005457.

(g) frame006163. (h) frame007159. (i) frame008346.

Figure 10: Results on sequence 2 with Deep-SDA algorithm.

[21] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A., “You only look once: Unified, real-time object detection,” *CoRR* **abs/1506.02640** (2015).

[22] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S. E., Fu, C., and Berg, A. C., “SSD: single shot multibox detector,” *CoRR* **abs/1512.02325** (2015).

[23] Law, H. and Deng, J., “Cornernet: Detecting objects as paired keypoints,” *CoRR* **abs/1808.01244** (2018).

[24] Saqib, M., Daud Khan, S., Sharma, N., and Blumenstein, M., “A study on detecting drones using deep convolutional neural networks,” *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–5 (2017).

[25] Aker, C. and Kalkan, S., “Using deep networks for drone detection,” *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–6 (2017).

[26] Redmon, J. and Farhadi, A., “YOLO9000: better, faster, stronger,” *CoRR* **abs/1612.08242** (2016).

[27] Schumann, A., Sommer, L. W., Klatte, J., Schuchert, T., and Beyerer, J., “Deep cross-domain flying object classification for robust uav detection,” *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–6 (2017).

[28] Lee, D., Gyu La, W., and Kim, H., “Drone detection and identification system using artificial intelligence,” *2018 International Conference on Information and Communication Technology Convergence (ICTC)* , 1131–1133 (2018).

[29] Nalamati, M., Kapoor, A., Saqib, M., Sharma, N., and Blumenstein, M., “Drone detection in long-range surveillance videos,” *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–6 (2019).

[30] Magoulianitis, V., Ataloglou, D., Dimou, A., Zarpalas, D., and Daras, P., “Does deep super-resolution enhance uav detection?,” *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–6 (2019).

[31] Craye, C. and Ardjoune, S., "Spatio-temporal semantic segmentation for drone detection," *2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–5 (2019).

[32] Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," *CoRR* **abs/1505.04597** (2015).

[33] Seidaliyeva, U., Akhmetov, D., Ilipbayeva, L., and Matson, E. T., "Real-time and accurate drone detection in a video with a static background," *Sensors* **20**(14) (2020).

[34] Akyon, F. C., Eryuksel, O., Ozfuttu, K. A., and Altinuc, S. O., "Track boosting and synthetic data aided drone detection," *2021 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* , 1–5 (2021).

[35] Elsayed, M., Mashaly, A. S., Reda, M., and Amein, A. S., "Visual drone detection in static complex environment," *2022 13th International Conference on Electrical Engineering (ICEENG)* , 154–158 (2022).

[36] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., "Going deeper with convolutions," *CoRR* **abs/1409.4842** (2014).

[37] Son, J., Baek, M., Cho, M., and Han, B., "Multi-object tracking with quadruplet convolutional neural networks," in [*2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 3786–3795 (2017).

[38] Lee, S. and Kim, E., "Multiple object tracking via feature pyramid siamese networks," *IEEE Access* **7**, 8181–8194 (2019).

[39] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering* **82**, 35–45 (03 1960).

[40] Chen, L., Ai, H., Zhuang, Z., and Shang, C., "Real-time multiple people tracking with deeply learned candidate selection and person re-identification," *CoRR* **abs/1809.04427** (2018).

[41] Zhou, H., Ouyang, W., Cheng, J., Wang, X., and Li, H., "Deep continuous conditional random fields with asymmetric inter-object constraints for online multi-object tracking," *CoRR* **abs/1806.01183** (2018).

[42] Lit, Z., Cai, S., Wang, X., Shao, H., Niu, L., and Xue, N., "Multiple object tracking with gru association and kalman prediction," in [*2021 International Joint Conference on Neural Networks (IJCNN)*], 1–8 (2021).

[43] Zhang, Y., Sun, P., Jiang, Y., Yu, D., Yuan, Z., Luo, P., Liu, W., and Wang, X., "Bytetrack: Multi-object tracking by associating every detection box," *CoRR* **abs/2110.06864** (2021).

[44] Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H., "Strongsort: Make deepsort great again," (2023).

[45] Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B. B. G., Geiger, A., and Leibe, B., "MOTS: multi-object tracking and segmentation," *CoRR* **abs/1902.03604** (2019).

[46] Wang, Z., Zheng, L., Liu, Y., and Wang, S., "Towards real-time multi-object tracking," *CoRR* **abs/1909.12605** (2019).

[47] Redmon, J. and Farhadi, A., "Yolov3: An incremental improvement," *CoRR* **abs/1804.02767** (2018).

[48] Bergmann, P., Meinhardt, T., and Leal-Taixé, L., "Tracking without bells and whistles," *CoRR* **abs/1903.05625** (2019).

[49] Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W., "A simple baseline for multi-object tracking," *CoRR* **abs/2004.01888** (2020).

[50] Zhou, X., Wang, D., and Krähenbühl, P., "Objects as points," *CoRR* **abs/1904.07850** (2019).

[51] Zhou, X., Koltun, V., and Krähenbühl, P., "Tracking objects as points," *CoRR* **abs/2004.01177** (2020).

[52] Tokmakov, P., Li, J., Burgard, W., and Gaidon, A., "Learning to track with object permanence," (2021).

[53] Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., and Luo, P., "Transtrack: Multiple object tracking with transformer," (2021).

[54] Meinhardt, T., Kirillov, A., Leal-Taixe, L., and Feichtenhofer, C., "Trackformer: Multi-object tracking with transformers," (2022).

[55] Saada, M., Kouppas, C., Li, B., and Meng, Q., "A multi-object tracker using dynamic bayesian networks and a esidual neural network based similarity estimator," *Computer Vision and Image Understanding* **225** (2022).