The Winning Solution for the Airborne Object Tracking Challenge

Dmytro Poplavskiy[1]

[1]Dmytro.Poplavskiy@gmail.com

1. ABSRACT

The solution to the airborne object tracking is based on the simultaneous detection and tracking algorithm applied to the pairs of aligned video frames and the simple objects association approach. The detection/tracking part of the pipeline is inspired by the CenterTrack tracker [1].

2. METHODS

The prediction pipeline consists of the following steps:

1. Find transformation between the previous and the current frames to compensate for drone movement and align the background.
2. Combine the current frame and aligned previous frames as inputs to the segmentation network, predicting at 1/8th of the original resolution the object's center, size, center offset, tracking vectors, and distance. Ensemble multiple models by simply averaging the object center predictions.
3. Associate the detected objects using the predicted tracking vectors in a similar way to CenterTrack, apply the similar postprocessing to the baseline solution to wait until a few objects are detected before reporting.

Frame alignment model

The initial approach to find transformation between frames was to use the OpenCV implementation of `goodFeaturesToTrack()`, `calcOpticalFlowPyrLK()` and `estimateAffinePartial2D()`. While this approach worked satisfactorily for many images, it was quite slow and failed for some frames. I replaced the OpenCV based approach with the deep learning-based approach, to estimate the optical flow vectors at the 1/32th resolution grid and the confidence/weight of estimated points, as illustrated in Fig. 1. The flow confidence output is inspired by the "Global Weighted Average Pooling" work, where the confidence is normalized with `exp(3sigmoid(x))`. During the training, the central 1024x1024 crop was used with the MSE (error * confidence) loss. I tested two approaches:

1. The "Simple" model, where the current and previous frames are separate input channels of ResNet models
2. The "TSM: Temporal Shift Module for Efficient Video Understanding" inspired model, where the current and previous frames are processed in separate ResNet models with weights shared and part of features exchanged between the convolutions.

The TSM based approach considerably outperformed the simpler combined input channels model, achieving around 2.0 MSE for TSM vs. 100-200 MSE for the simple model on the validation set, as illustrated in Fig. 2. The model runs at around 60fps with 1024x1024 central crops and 2080ti GPU, for the last submissions I switched to bigger crops of 2048x1280 around the lower part of the image.

The model has been trained on 75% of synthetic random transformation and 25% of actual frame pairs with transformation estimated using OpenCV[1], and the cosine annealing with restarts LR scheduler.

[1] For each sample with the probability of 25%, the actual pair of frames was used, with an estimated transformation based on OpenCV, otherwise a simulated random transformation was applied to the current frame to generate a pair.

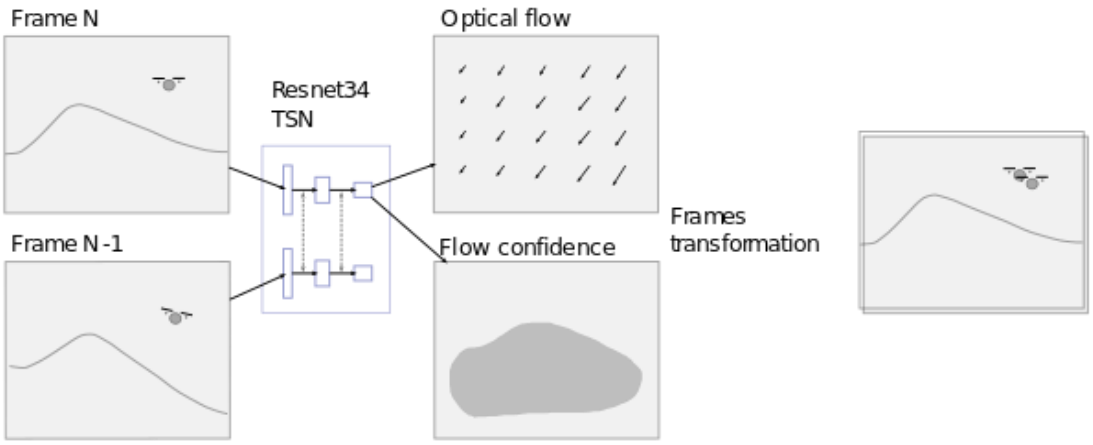
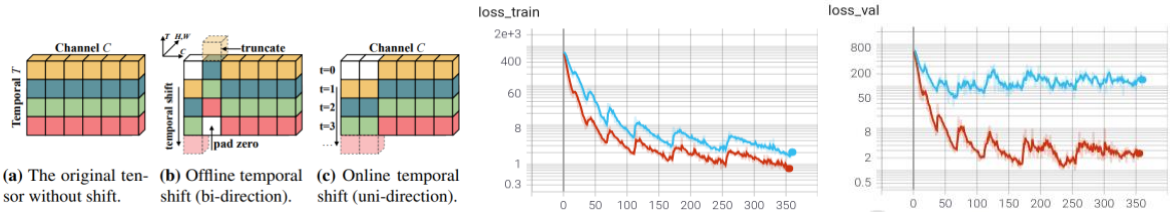


Figure 1: ***Frame alignment overview.***



(a) The original tensor without shift. **(b)** Offline temporal shift (bi-direction). **(c)** Online temporal shift (uni-direction).

Figure 2: ***Frame alignment TSM-inspired model.*** Illustration of the Temporal Shift Module (left) and comparison (right) of the training and validation losses for the simple ResNet based model (blue) and ResNet + TSM model (red)

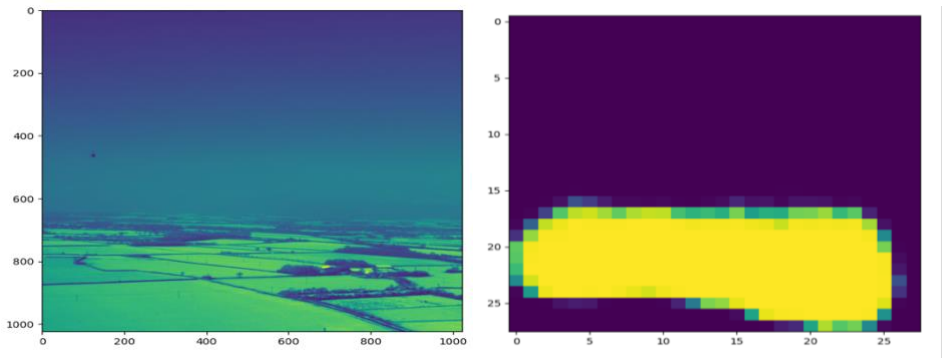


Figure 3: ***Visualization of the flow confidence map***. Interesting observation: unlike OpenCV, the model has learned to ignore the airborne objects.

Potential improvements for the alignment process:

1. With the computing power restrained environment, the frame alignment model can be replaced with IMU/Gyro readings and the camera model.
2. The model can be changed so activations are only copied from the previous frame to the current one and the result of the current frame branch activations is to be re-used for the next step. This allows to 2x improve the inference time and use information from more than one previous frame at zero cost.
3. The estimated affine transformation does not take into account the parallax due to drone movement and camera distortions; this certainly can be improved with information about the drone reference frame, the terrain elevation map, and the camera.

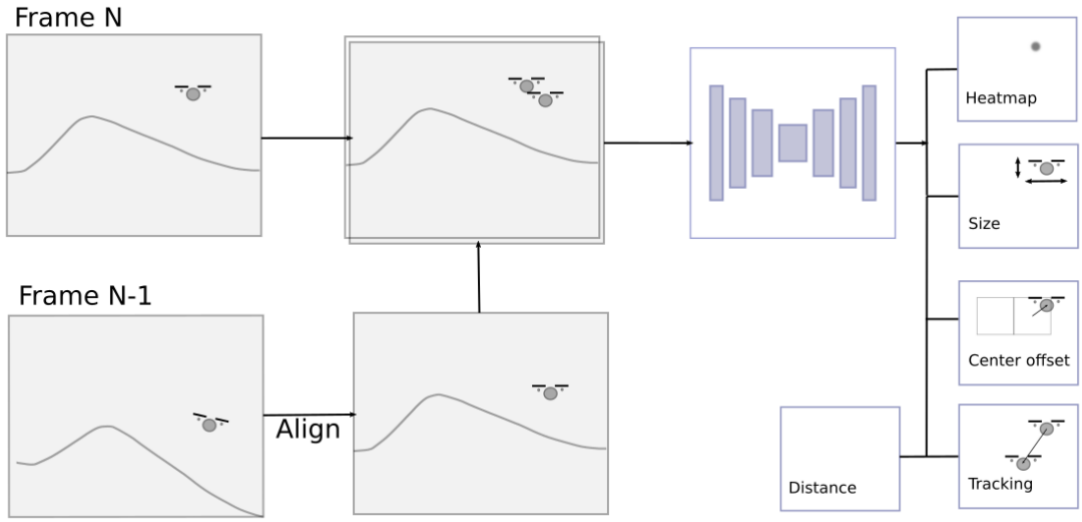


Figure 4: ***Detection and Tracking model***

Detection and Tracking Model

The detection and tracking model is inspired by CenterTrack [1] and has a very simple architecture: the current and aligned previous frames are used as different channels of the segmentation base input, with 8-pixel stride multiple outputs.

A number of models have been tested as the segmentation base, with multiple models performing comparably well, including:

- HRNet32, HRNet48
- EfficientDet D2-D5, only included the EfficientNet backbone and BiFPN
- Custom backbone with BiFPN from EfficientDet (the submission included gernet_m with BiFPN from EfficientDet D2)
- DLA34, DLA60

Unlike the original CenterTrack, the model does not receive the previously predicted heatmap as an input. This is done to avoid the model "getting stuck" to previously detected false positives, make the following post-processing more efficient and simplify the training pipeline.

The segmentation base produced multiple outputs:

- Heatmap of the object centers.
- Size of the object at the log scale.
- Offset from the output grid center to the object center, calculated for the area of a 3x3 pixel around the object center.
- Distance to the object at the log scale.
- Tracking offset between the aircraft positions at the current and previous frames.

Processing the segmentation model output

Converting the segmentation outputs to detection and tracking results is straightforward: the maximum value is selected on the heatmap, with the center offset, size, distance and tracking offsets are read from the matching points on relevant outputs. Values around the heatmap center are suppressed using a similar but slightly bigger mask is used during training and a simple NMS applied.

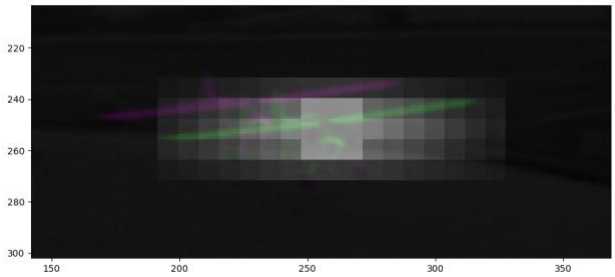


Figure 5: ***Example of the heatmap*** stacked with a pair of input frames.

Model training

The models have been trained using the MADGRAD or SGD and the cosine annealing with restarts LR scheduler. On the few experiments comparing optimizers, SGD performed slightly better or comparable to MADGRAD or Adam.

The loss used is the weighted sum of the heatmap, center offset, size distance, and tracking loss values:

$$L = w_1 L_{heatmap} + w_2 L_{center\ offset} + w_3 L_{size} + w_4 L_{distance} + w_5 L_{tracking}\ , \ [1]$$

where:

$$L_{heatmap} = \min_{pixelwise}\left(FocalLoss(heatmap, all\ airborne), FocalLoss(heatmap, selected\ planned\ airborne)\right)$$

$$L_{center\ offset} = \frac{\Sigma\ (offset\ error)^2\ \times\ offset\ mask}{1 + \Sigma\ offset\ mask}$$

$$L_{size} = \frac{\Sigma\left(\log(size_{pred}) - \log(size_{gt})\right)^2\ \times\ size\ mask}{1 + \Sigma\ size\ mask}$$

$$L_{distance} = \frac{\Sigma\left(\log(distance_{pred}) - \log(distance_{gt})\right)^2\ \times\ distance\ mask}{1 + \Sigma\ distance\ mask}$$

$$L_{tracking} = \frac{\Sigma\ (tracking\ error)^2\ \times\ tracking\ mask}{1 + \Sigma\ tracking\ mask}$$

The offset error is an L2 norm of the error in the prediction of offset from each pixel to the box center. Tracking loss is the L2 norm of the error in the prediction of tracking vectors, where a tracking vector is a difference between the current box center position and the previous center, corrected for frames alignment. The tracking vector is clipped to +/– 256 pixels, which was useful with initial OpenCV based frames alignment where sometimes errors in an alignment produced very large offsets (it might not be necessary anymore given the updated alignment model).

The heatmap is similar to that used in CenterTrack with the Gaussian-shaped peak around the object center and 3x3 box with value of 1. Since the value of the predicted heatmap peak is used for the prediction confidence, using a bigger 3x3 patch even for smaller objects helped to avoid the impact of the center position uncertainty on the prediction confidence, since the central patch size is sufficiently large.

The same mask is used for the distance size and tracking regression, while for the center offset, the values are calculated only for the central 3x3 box.

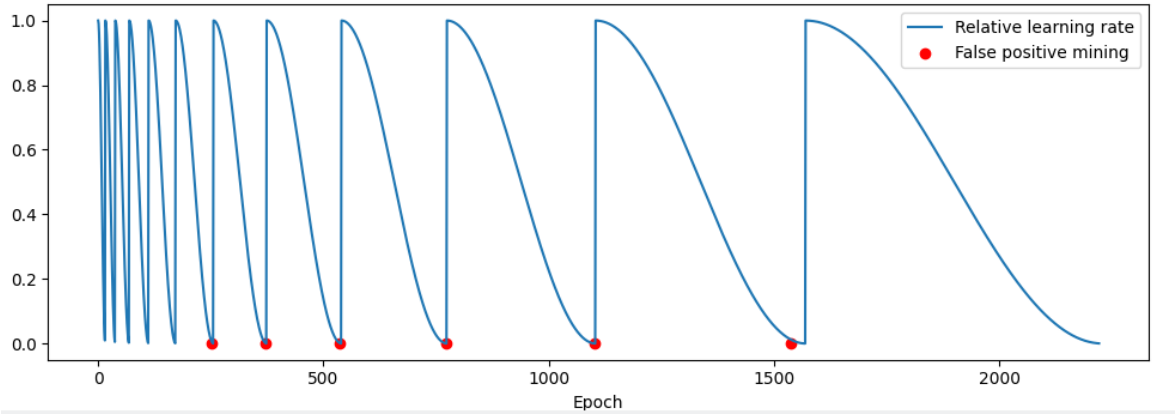


Figure 6: ***Learning rate schedule and the false positive mining epochs***

Unlike CenterTrack, I decided to use the MSE of the log values instead of MAE due to the large range of object sizes[2], plus for IoU metric the relative size is important and the difference of log(size) would better match the metric.
The intuition behind the pixel-wise minimum of heatmap loss for the planned and all airborne: training on all objects would produce a large number of false positives, especially for harder to predict smaller objects, training only on the planned airborne within the short distance may incorrectly mark other airborne at the close distance as negative, while the pixel-wise minimum between all and planned objects loss mimics the task objective and metrics used: for other planes, it's "Don't care" without significant loss penalty regardless on plane predicted or not. However, when I re-tested the impact of the combined heatmap loss, the result of the simple "only planned airborne" approach produced slightly better results, so such an idea with combined loss requires more testing.

The models have been trained on the random 512x512 crops, and sampling of crops had a significant impact on the model performance. The initial experiment when the crops with planes have been heavily oversampled performed worse compared to experiments with less positive samples oversampling and all images used for training. The final strategy was as following: For 50% of samples, I selected the completely random crop, for 25% samples I selected frames with known hard false positives. For the rest 25% of samples, I selected frames with the airborne. but forced to include airborne only with 25% probability. So overall only 1/16 of 512x512 crops would have the airborne object forced to be around the center.

To reduce the number of false positives, hard negative mining was applied: six times during the training, prediction for every n^{th} frame was done and position for all false-positive predictions with confidence over 0.2 are saved. Found negative samples are used for 25% of training samples.
When tested with the single gernet_m model, training with negative mining allowed to improve AFDR by around 4-6% with the score threshold selected to match target FPPI (Table 1 experiment 2 vs. 1). When a model is trained on a pair of alignment frames, but receives the unaligned frames, the performance significantly drops. When instead of the previous frame, the copy of the current frame is used, the performance drops drastically, which indicates the information about the movement between frames helps to improve the prediction performance (Table 1 experiment 3 and 4). Providing the model with more historical frames does not improve the result noticeably.

[2] Evaluating MSE(log) (originally used by CenterTrack) after the challenge showed that the results are similar to MAE

Table 1 – *Ablation study* on Hard Negative Mining (HNM) and frame stacking (Best in **Bold**)

Experiment	Model	Score@ 0.0005	AFDR@ 0.0005	Score@ 0.0002	AFDR@ 0.0002
1	gernet_m with HNM	0.62	**79.7**	0.69	**77.8**
2	gernet_m	0.79	76.1	0.85	71.5
3	gernet_m with HNM, with the current frame used instead of the previous + the current frame	0.66	55.7	0.73	50.4
4	gernet_m with HNM, single (current) frame	0.74	57.2	0.80	49.0

Detection/track association

The detection/track association is quite simple and uses mostly the predicted tracking vector. For larger aircraft (at least 32pix linear size), the tracking error is scaled relative to the aircraft size. The extra penalty is applied for low confidence detection, the difference in size between prediction pairs.

I used the 40 pixels threshold (including the confidence and size difference penalty), but quite a large range of thresholds worked well.

If the found plane could not be associated with any track from the previous frame, a few previous frames, aligned to the current one, are checked for unfinished tracks.

After the detection/track association, the simple post-processing step is applied to delay reporting detection until the track is at least 8 steps long to reduce the number of false positives, in the same way as the baseline solution does.

Ensembling approach

Since the very low number of false positives is important for the task, ensembling the diverse set of models helps to suppress the occasional false positive since when all models in the ensemble already perform well and have a low number of false positives, it's less likely all models would make the same error. Nevertheless, it is still hard to prevent all the errors using this method, with one train/validation split I had all models predicting passing cars as aircrafts, so the diverse training dataset is still very important.

When ensembling, one or two faster models are run on the full resolution and other slower models are applied on 512x512 size crops. This allows ensembles with slower models with much less impact on the prediction time since the crop area is around 20x smaller compared to the full resolution input. I averaged only the heatmap predictions, using other inputs from the single hrnet32 model.

The biggest impact was from ensembling two models, with smaller improvements for bigger ensembles. The final submission ensemble 4 models, hrnet32 + gernet_m predicted at the full resolution and dla60 + hrnet48 predicted on crops.

Table 2 – *Impact of ensembling* on AFDR, estimated on 572 tracks validation set. The ensemble runs at around 2-3 FPS at the full resolution (Best in **Bold**)

Model	AFDR @ 0.0002 FPPI	AFDR @ 0.0005 FPPI
hrnet32	78.2	81.2
hrnet48	81.7	84.7
gernet_m	77.8	79.7
dla60	79.1	81.6
dla34	76.2	78.1
hrnet32 + hrnet48 ensemble	84.3	85.9
hrnet32 + hrnet48 + gernet_m + dla60 ensemble	**85.5**	**87.1**

3. RESULTS

Table 3 – *Results from Public Leaderboard* (red denotes result that does comply with FFPI/HFAR budget of 0.0005 and 0.5 accordingly, **bold** denotes result that gets highest AFDR/ HFAR for the provided budget of FFPI/HFAR)

Submission	Public Leaderboard				
	Score threshold	FPPI	AFDR	HFAR	EDR
5	0.80	0.0008	0.84	9.3	0.99
6	0.75	2.5e-05	0.83	0.61	0.97
7	0.70	4.2e-06	0.80	0.15	0.94
8	0.70	8.4e-06	0.80	0.23	0.96
9	0.70	6.3e-06	0.81	0.23	0.95
10	0.70	6.3e-06	0.82	0.15	0.96
11	0.60	4.0e-05	0.89	0.38	0.99
12	0.60	4.2e-05	0.91	0.38	0.99
13	0.70	2.1e-06	0.89	0.07	0.98
14	0.50	0.0002	0.93	0.76	1.00
15	0.75	0.0	0.87	0.0	0.96

Table 4 – *Results from Both Leaderboards* (red denotes result that does comply with FFPI/HFAR budget of 0.0005 and 0.5 accordingly, **bold** denotes result that gets highest AFDR/ HFAR for the provided budget of FFPI/HFAR)

Submission	Public Leaderboard					Private Leaderboard				
	Score threshold	FPPI	AFDR	HFAR	EDR	Score threshold	FPPI	AFDR	HFAR	EDR
20	0.675	1.7e-05	**0.90**	0.15	**0.99**	0.675	5.3e-05	0.87	0.30	0.95
21	0.75	0.0	0.87	0.0	0.96	0.75	4.0e-05	0.85	0.23	0.91
22	0.6	5.0e-05	**0.91**	0.38	**0.99**	0.60	7.8e-05	**0.89**	0.38	**0.96**

Submission details:

5. The single HRNet32 model trained on parts 2+3 only. No postprocessing, naive tracking assigning track id in the order of confidence.

6. Ensemble of HRNet32, gernet_m (using BiFPN from EfficientDet D2) and EfficientDet D5, trained on the full dataset.

7. The first implementation of the simple predicted offset-based tracking.

8. Replaced EfficientDet D5 to HRNet48.

9. Replaced gernet_m with DLA60

10. Postprocessing parameters changes, based on the validation set tuning: reduced threshold for NMS, exclude a narrow margin around borders from the heatmap prediction since the annotation may be unreliable (is a few parts of the plane is visible, model may predict the different part to the original annotation), increased the distance threshold from 850 to 900m.

11. Use 4 models' ensemble with lower threshold.

12. Estimate the full affine transformation between the current and previous frame instead of only offset and angle. Increased the crop size for transformation estimation between frames: from the centered 1024x1024 crop to 2048x1280, covering more the bottom of the frame.

13. Increased the score threshold to 0.7

14. Allow bigger tracking error for faster moving planes. Reduced the score threshold to 0.5. The model found all tracks but exceeded HFAR.

15. Increased the score threshold to 0.75, zero false positives on the public test set.

Final submissions 20-22 used the same configuration as submissions 14-15 with the different score threshold values.

4. REFERENCES

[1] Tracking Objects as Points

[2] Suo Qiu, Global Weighted Average Pooling Bridges Pixel-level Localization and Image-level Classification

[3] Ji Lin, Chuang Gan, Song Han, TSM: Temporal Shift Module for Efficient Video Understanding

[4] Mingxing Tan, Ruoming Pang, Quoc V. Le, EfficientDet: Scalable and Efficient Object Detection

[5] Ming Lin, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, Rong Jin, Neural Architecture Design for GPU-Efficient Networks

[6] Fisher Yu, Dequan Wang, Evan Shelhamer, Trevor Darrell, Deep Layer Aggregation