# CREATING A DATASET FOR THE DETECTION OF FLYING BIRDS

**Constantinos Constantinopoulos**
nvisionist
Marousi, Greece
cc@nvisionist.com

**Ioannis Misios**
nvisionist
Marousi, Greece
i.misios@nvisionist.com

**Athanasios Sakelliou**
nvisionist
Marousi, Greece
t.sakelliou@nvisionist.com

**ABSTRACT**

We present a pipeline for collecting a dataset for the detection of flying birds in videos. We treat the creation of a dataset as an iterative task, which allows to train an inference system periodically, and to improve gradually its performance. We follow a three stage pipeline, that includes repetitive video recording, video annotation and frame sampling for training a Deep Learning detector. Furthermore, we would like to point out that the end goal of our work is to protect birds flying through wind farms, in order to take protective measures. Therefore it is crucial to find the moment that a bird enters and exits the field of view of the camera. Thus we use our dataset to setup a Temporal Action Detection (TAD) task, and we evaluate our inference pipeline using an appropriate measure.

***K*eywords** wildlife protection · wind farm · machine learning · dataset compilation

## 1 Introduction

Training an efficient Deep Learning model requires a large and comprehensive dataset, but this is not something easy to achieve, because it requires considerable time and resources. Therefore the only viable approach is to grow a dataset in cycles, while in the meantime the model is retrained. This has become common knowledge for industrial applications, and has given rise to the implementation of application specific *data engines*. A data engine has three main functions: data collection, data annotation and dataset sampling for training. Herein we present the data engine we have implemented for tackling the problem of *temporal action detection for small objects* (TAD4SO).

One the one hand, the focus to small objects is necessary for a system that aims to protect birds from collisions with wind turbines. Because an early detection means that most of the time the depicted birds are small, have few salient details, and are affected by camera lens distortion and motion blur. On the other hand, the focus to temporal action detection is equally important, because a bird approaching a wind turbine should trigger protective measures. Therefore detecting the starting and ending frames of the event are more important than individual detections at intermediate frames.

Regarding our contribution, it has two axis. First, we present the design of a data engine that is adequate for a production system. This goes beyond the mere introduction of a new dataset. Second, we argue that the appropriate way to judge the performance of a system that protects wildlife, it is to weight how well it estimates the time interval of an event. On that end, we present preliminary results of temporal action detection using appropriate measures from the literature, deviating from the typical reports of precision and recall.

### 1.1 Prior Art

To our knowledge, there are no published data engines which can be integrated into wildlife protection systems. However, there are several published datasets regarding bird detection, and we focus our review on them. On the contrary, there are published measures for evaluating the performance of temporal action detection, and the brief review that follows aims to make them known to the wider scientific community.

### 1.1.1 Datasets

Early datasets of birds are focused on the task of object classification. Therefore they are quite generic regarding the birds appearance. Namely, they contain birds in various poses, and the relative size of birds is medium or large w.r.t. the image size. The most important datasets of this type are: CUB-200-2011 [1], Birdsnap [2], NABirds [3], VB100 [4], LASBIRD [5, 6] and GBDD1433-2023 [7]. To highlight the scope of the most recent ones, the GBDD1433-2023 consists of 148000 images with 1433 bird species, while the LASBIRD consists of 5000000 images with 11000 species. Naturally, therein exist subsets of images with flying birds, but using them for temporal action detection is not straightforward.

However, latter efforts have produced specialized datasets with flying birds, focused on the task of object detection. The notable differences is that all the depicted birds are flying, their apparent size is small, and the sky is a considerable part of the background. In the following, we give a concise description of those datasets.

The dataset NAE-LAB [8] was created with the specific purpose of recording birds in a wind park. It consists of 32000 images, with a resolution of $5616 \times 3744$ and frame rate 0.5 fps. The images are manually labeled with bounding boxes and class labels. The dataset contains various bird species as well as other types of relevant objects, in total there are 10 classes. Namely: undefined bird, black kite, gray-faces buzzard, eastern marsh harrier, jungle crow, carrion crow, Japanese white-eye, airplane, helicopter, and undefined object. The total number of annotated birds is over 60000, and the number of non-birds is 6000. Regarding the bird size, roughly 60% of the instances are 20 pixels or less.

The dataset Skagen and Klim [9, 10] was recorded at the Skagen Grey Lighthouse and the Klim Fjordeholme wind farm. It consists of grayscale images with 4K resolution ($3840 \times 2160$), recorded at a frame rate of 5 fps. In total, there are 1700 frames annotated with 2400 bounding boxes. Regarding the size of the depicted birds, in the Klim subset the 40% has an area of less than 500 pixels, while in the Skagen subset for the 90% the area is less then 200 pixels.

The dataset AirBirds [11] was recorded in an airport. It consists of 118312 time-series images with a resolution of $1920 \times 1080$ pixels. Those images were manually annotated with 409967 bounding boxes. The original recordings were videos, and the parts which depicted birds were uniformly sampled at 5 fps. Regarding the apparent size of birds, almost 90% of the cases are smaller than 10 pixels.

Finally, the dataset for Safe Drone Flight [12] was recorded with a flying drone. The birds were captured in varying distances, from 10m and to 200m, over various backgrounds. It consist of 21837 images with a resolution of $3840 \times 2160$ pixels, while the number of annotated bounding boxes is 4467. It is worth mentioning that it was included and extended in the context of the *Small Object Detection Challenge for Spotting Birds 2023* [1]. The resulting dataset SOD4SB [13] consists of 47260 images with 60971 annotated bird instances.

Table 1: Comparison of relevant datasets.

| **Dataset** | **Location** | **Resolution** | **Images** | **Birds** |
|---|---|---|---|---|
| NAE-LAB | Wind farm | $5616 \times 3744$ | 32,000 | 60,000 |
| Skagena-Klim | Lighthouse<br>Wind farm | $3840 \times 2160$ | 1,700 | 2,400 |
| AirBirds | Airport | $1920 \times 1080$ | 118,312 | 40,996 |
| SafeDroneFlight | *Various* | $3840 \times 2160$ | 47,260 | 60,971 |

### 1.1.2 Performance Evaluation

The typical evaluation of flying bird detection is focused on spatial localization inside individual frames [8, 10, 14, 15, 16, 17, 18]. Therefore, the performance is measured by the classical mean Average Precision (mAP) and mean Average Recall (mAR), which were introduced for object detection in the COCO challenge [19]. However we would like to shift the focus to temporal localization. The natural choice are measures that were introduced for temporal activity detection.

The challenge THUMOS2014 [20] introduced a dataset for temporal action detection in videos, where the action is limited to a subset of the frames. For the evaluation of competing algorithms, they are based on the temporal Intersection Over Union (IOU) similarity measure. The inferred actions are sorted in decreasing detection confidence, and matched with an annotated action based on temporal IOU. Given a threshold $tIOU$, if the temporal overlapping between one inferred and one annotated action is above the $tIOU$, this case is considered as a True Positive (TP). If more than one detections exceed the threshold with a given annotated action, only the one with maximum $tIOU$ is considered as True Positive, while the rest are counted as False Positives (FP). Naturally, any detection that doesn't exceed the $tIOU$ with

[1] Challenge site: https://www.mva-org.jp/mva2023/challenge

any annotated action is a False Positive case. Moreover, if an annotated action is not sufficiently overlapping with any detection, it is counted as False Negative (FN). Finally, given the number of TP, FP and FN cases, they compute the Average Precision (AP) and Average Recall (AR) for a certain $tIOU$. This procedure is repeated for a number of $tIOU$ thresholds, accounting for loose or strict temporal detections.

In [21] they introduced the 2-Class Segment Error Table (2SET), with the intention to further analyse the FP and FN errors that occur when the inferred and annotated actions are temporally misaligned. The 2SET is an extension of the Confusion Matrix, where the FP cases are further divided into the following:

- *inserted segment* $IS$: the detected action can't be matched with any annotated one
- *merging segment* $MS$: the detected action has overextended duration, covering two successive annotated actions
- *overfilled head segment* $OHS$: the detected action starts earlier than the matched annotation
- *overfilled tail segment* $OTS$: the detected action stops later that the matched annotation.

Respectively, the FN cases are further divided into the following:

- *deleted segment* $DS$: the annotated action can't be matched with any detection.
- *fragmenting segment* $FS$: the annotated action is matched with two successive but not continuous detections
- *underfilled head segment* $UHS$: the detected action starts later than the matched annotation
- *underfilled tail segment* $UTS$: the detected action stops earlier than the matched annotation.

The aforementioned segments are defined after matching detections with annotations. For each pair or set, all the starting and stopping timestamps are sorted by ascending order. Then every pair of consecutive timestamps defines a video segment, which is classified according to the above definitions. Finally, the 2SET is formed by summing the frames of each segment type. Given this table, it is straightforward to calculate percentages for the corresponding type of errors. The details are given in Section 2.5.

## 2 Materials and Methods

The implemented data engine, has three stages. It starts with video recordings in the field, then follows the annotation of the videos on premises, and concludes with selecting an appropriate dataset for training. Before presenting detailed descriptions, we would like to mention that the design which we adopted took into account two constraints.

The first constraint is limitations to the network connection. Because the inference servers are deployed on-the-edge, namely a server at each wind turbine generator, and all generators are connected to the substation of the wind park, in order to transmit data to the internet. Although the generators are connected to the substation with a low latency and high bandwidth link, the connection from the substation to the web could be limited by: bandwidth (e.g. satellite link), data volume (e.g. bellow 5GB), and time window (e.g. after business-hours).

The second design constraint is the sparsity of relevant recordings. Most of the time, the most of the cameras are recording uninformative scenes without birds. Even certain false events that are useful for optimizing the detector, like flying insects or airplanes, do not happen very often. Moreover, the distance of the object to the camera deteriorates the situation, because as the distance increases the observable details are diminished. Therefore we have to block such cases from entering the dataset, making the relevant recordings even more sparse.

### 2.1 Video Sampling Pipeline

The first stage of the adopted data engine is recording events from cameras in the field. We are interested in both TP and FP events, so we have implemented a dedicated procedure for each case, the *periodic-sampling* and the *filtered-sampling* correspondingly.

The periodic-sampling consists of recording short 4K ($3840 \times 2160$ pixels) videos a few times per hour, during the time of maximum bird activity. The duration of each video is 5 min, and the recording starts every 15 minutes. In total, we get 20 min per hour. The optimal interval of the day for recording is provided by experts in ornithology, who take into account the habits of the local bird population. Initially, the videos are stored locally at the edge, in a dedicated hard disk. The older videos are deleted in order to free space for the new ones, so we have always available the most recent recordings. Periodically, all videos from all wind turbine generators are transferred to the Network Attached Storage

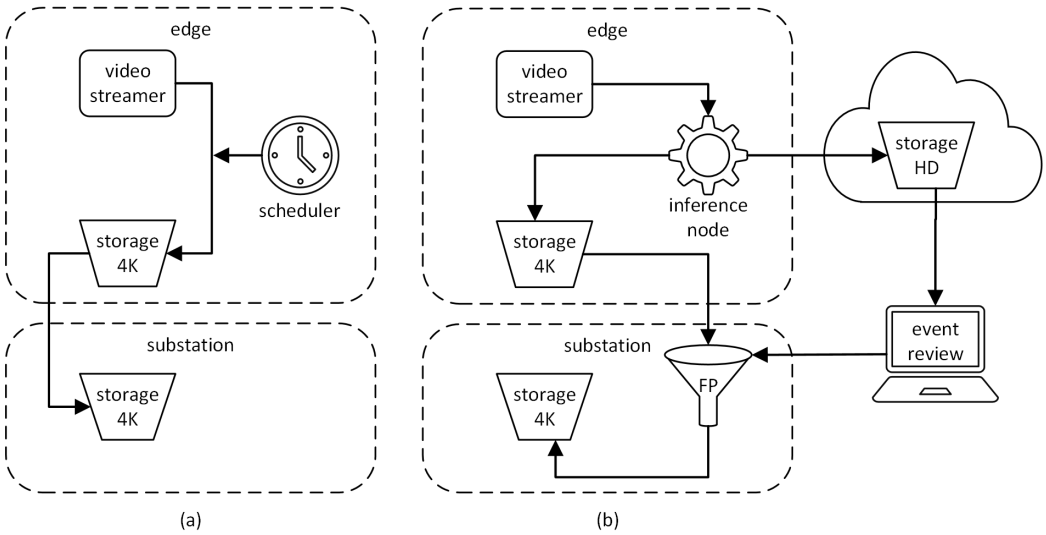


Figure 1: Two pipelines for collecting high-resolution videos from the field. Due to bandwidth limitations it is not feasible to stream straight from the edge to the cloud, so we retreat to sampling and gathering videos in the substation of the park, for manual collection. (**a**) *Periodic-sampling* of videos based on a time schedule. (**b**) *Filtered-sampling* of videos based on events that have been manually assigned as False-Positive ones.

(NAS) that is installed to the substation of the park. Then a technician collects all hard disks of the NAS, and transfers them in the premises for further processing. This procedure is depicted in Figure 1 (a).

The filtered-sampling involves an expert annotator, and has three stages. Initially, when the inference node at the edge detects a bird, uploads the HD ($1920 \times 1080$ pixels) video of the event on the cloud and keeps a 4K copy locally. Periodically, the annotator reviews all the uploaded events, and flags the errors. Finally, when enough videos have been identified as False Positive events, the corresponding high-resolution videos are transferred from the edge to the substation, for collection by the technicians. This procedure is depicted in Figure 1 (b).

### 2.2 Semi-Automated Annotation of Videos

The second stage of the adopted data engine is annotating the collected videos, using a semi-automated pipeline that involves expert annotators. The end result is a dataset increment stored in a database. We use the term *increment* in order to emphasize that we build our dataset iteratively.

The pipeline is built using six components:

- *source volume*: the hard disk drive (4 TB) with the videos
- *swap volume*: the hard disk drive (16 TB) with the frames
- *destination volume*: the hard drive (4 TB) with the datasets
- *inference server*: a server for deploying various CNN models
- *FiftyOne*[2] *server*: a server for storing and viewing datasets
- *CVAT*[3] *server*: a sever for annotating datasets.

For each new hard disk drive that arrives from the wind park we follow a seven steps pipeline, as depicted in Figure 2. First, the 4K videos are converted to full-resolution frames. Only a subset of the frames is kept in the swap volume, namely the $1/8th$ of frames distributed uniformly in time. Second, the frames are fed into the inference server, which detects any objects of interest therein and returns the corresponding bounding boxes. Third, given the frames and the bounding boxes, the annotation is compiled using the FiftyOne format. Fourth, the annotation is transferred to

[2]FiftyOne site: https://voxel51.com/fiftyone/
[3]CVAT site: https://www.cvat.ai/

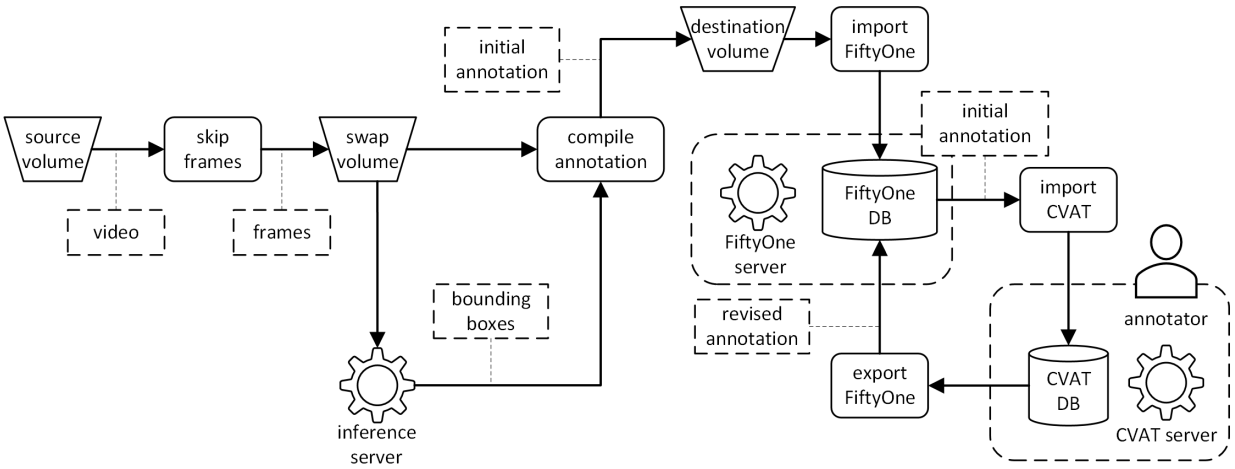


Figure 2: The pipeline for the semi-automated annotation of the videos that where collected in the field. The first stage of the annotation is automatic and based on the detector that we deploy in production. The second stage is manual, and the annotator updates the initial results.

the destination volume and imported into the FiftyOne server. Fifth, the annotation is imported to the CVAT server for manual revision. Sixth, an expert annotator reviews the initial annotation of the dataset and produces the final annotation. Finally, the revised annotation is imported back to the FiftyOne server.

Regarding the inference server, it is worth mentioning that it supports various CNN models that will be deployed in production for detection or even classification of birds. Specifically for the task of detection, the model is deployed with a lower confidence threshold, that allows to collect negative examples.

### 2.3 Data Handling for Training

The third stage of the adopted data engine creates an appropriate dataset for training CNN models, taking care of: the cross-validation subsets, selecting the most informative training samples, and updating the dataset. The end result is a set of files in the destination volume, that can be directly used for training a CNN model. The corresponding pipeline is depicted in Figure 3, and it consists of five steps.

Given a new dataset that is created by the semi-automated annotation, the first step is extracting the metadata of the frames. The relevant metadata include:

- the size of the annotated bounding box
- the date of recording
- the time of recording
- the type of camera
- the ID of camera
- the ID of the wind turbine generator
- the ID of the wind farm.

The next step is optional, and is responsible for merging the datasets that are periodically collected in the field. Moreover, it is also able to merge datasets from alternative sources, for example public and private datasets. The third step is applying various filters in order to select the most valuable cases for training. The meaningful filters could vary a lot depending on the training scenario, because the ultimate goal is to focus the training in challenging cases. However we would like to list a few that we found useful:

- a lower threshold on the size of the bird, for baseline training
- an upper threshold on the size of the bird, for training with distant targets

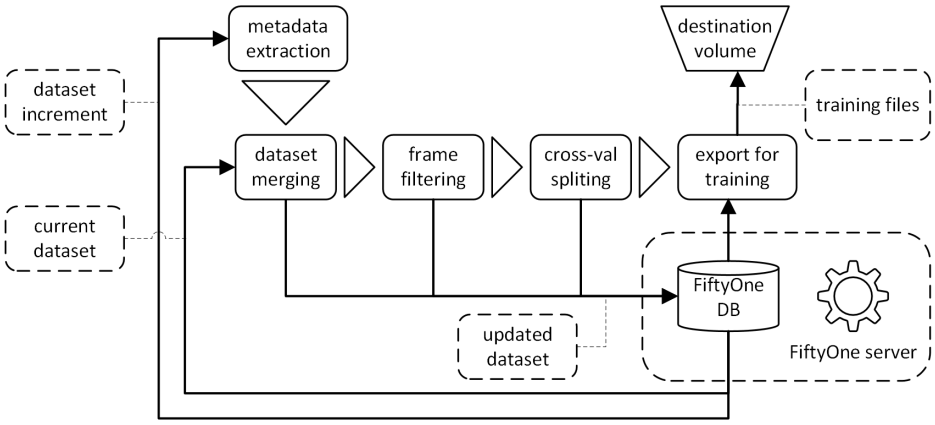


Figure 3: The pipeline for data handling, which creates a set of files ready for loading by a training function.

- selected date and time intervals, for training with specific environmental conditions, like a storm or the sunrise
- selected wind farms and wind turbines, for training with challenging cases of background objects.

The fourth step is to take care of the cross-validation. More specifically, it splits the data in the necessary $k$ folds for the stages of training, validation and testing. Moreover it exploits the metadata in order to avoid target leakage, by making sure that frames from the same camera do not cross folds. In the final step, the dataset is exported on the file system of the destination volume, in a format that is appropriate for the CNN model we wish to train.

### 2.4 Pipeline for Temporal Action Detection

The pipeline for TAD accepts as input a sequence of frames, and returns the interval with flying birds. The pipeline consists of three components, as it is depicted in Figure 4. The first component is a CNN model that detects birds in individual frames. The model has been trained using various types of objects, including airplanes, insects and wind turbine blades. However, only the birds class is considered during inference. The next component is a tracker that ensures time consistency, by dropping isolated detections and also estimating the individual trajectories of concurrent objects. Finally, there is a number of filters based on the estimated trajectories. Those are optimized for rejecting erroneous detections caused by airplanes, insects, etc. If a trajectory passes all the filters, then it is considered as an event of flying bird.

In order to approximate as much as possible the inference during real world deployment, we have adopted for the testing a number of constraints. First, only one-pass forward scanning of the frames is allowed, and only the current frame can be identified as the start or the end of the event. Therefore we are not allowed to optimize the event interval by considering a previous frame as the limit. Second, we are forced to uniformly skip a few frames. Because we would like to match the throughput of the production, where we have achieved an inspection rate of 3 frames per second per camera. Therefore during testing we have implemented a uniform sampling of the videos with the same frame rate. Third, in the case of birds flying together a single event should be reported. Because in production only one protective measure can be triggered each time, the one with the highest priority.

Finally, a consequence of the first constraint is that we have introduced some inertia in the estimation of the starting and stopping frames of the event. Because we would like to avoid reporting an event based on very little evidence in the input. Therefore we wait for 4 detections before identifying the starting frame, and we wait for 3 empty frames before identifying the stopping frame.

### 2.5 Evaluation of Temporal Action Detection

Given the starting and stopping timestamps assigned by the annotator and the ones detected by the inference pipeline, we split each video into segments. All the timestamps are projected onto a common timeline, and every pair of consecutive timestamps (including the first and last frame) defines a new segment. Then the detection for each segment is judged as correct or wrong, specifying eight types of error. In order to take into consideration the length of a segment, the number of frames per segment is used in the calculations of error rates.

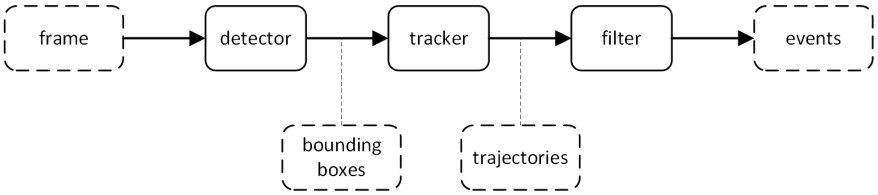


Figure 4: The inference pipeline for the temporal action detection consists of a CNN detector, a tracker and set of trajectory filters.

Assuming that $|PS|$ is the total number of frames in the positive frames according to the annotator, and $|NS|$ the total number of frames in the negative segments, in [21] they define the following 2SET metrics:

- the rate of deletion: $DR = \frac{|DS|}{|PS|}$
- the rate of fragmentation: $FR = \frac{|FS|}{|PS|}$
- the rate of underfilled head: $UHR = \frac{|UHS|}{|PS|}$
- the rate of underfilled tail: $UTR = \frac{|UTS|}{|PS|}$
- the rate of insertion: $IR = \frac{|IS|}{|NS|}$
- the rate of merging: $MR = \frac{|MS|}{|NS|}$
- the rate of overfilled head: $OHR = \frac{|OHS|}{|NS|}$
- the rate of overfilled tail: $OTR = \frac{|OTS|}{|NS|}$

It is straightforward to show that the loss of true positive rate is:

$$(1 - TPR) = DR + FR + UHR + UTR.$$

Similarly, the loss of true negative rate is:

$$(1 - TNR) = IR + MR + OHR + OTR.$$

## 3 Results

### 3.1 The NVbirds Dataset

The deployment of the proposed data engine in our production system has resulted in the creation of a reach dataset for bird detection. The videos that we collected have been recorded by 19 cameras, which were installed in 7 wind turbine generators across Europe. The dataset contains 1118 videos with a total duration of 3.2 hours. The total number of frames is 220000 and the total number of annotated birds is 73000.

In Figure 5 we show a few typical examples contained in our dataset. It is worth to comment on the inclusion of blades. Although most wind turbines have blades with a smooth ending, there are a few manufacturers that offer blades with a bended ending. This variation used to cause a lot of FP errors in production, so we used the proposed data engine to specifically target wind turbines with that shape.

In Figure 6 we present the histogram of apparent bird size, based on the diagonal of the bounding box. The most frequent bird size is between 15 and 20 pixels, while the 65% of the cases are less than 25 pixels.

### 3.2 Performance Evaluation

For the evaluation of our TAD method we used the subset of the NVbirds dataset which has been annotated with bird trajectories and the corresponding time intervals. In Figure 7 we show an example of the annotated trajectory. The evaluation dataset consists of 31 videos with 15000 frames and 15600 bounding boxes. In these videos there are 31


Figure 5: A sample of the objects in the NVbirds dataset, from top to bottom: birds, insects, clouds and wind turbine generators.

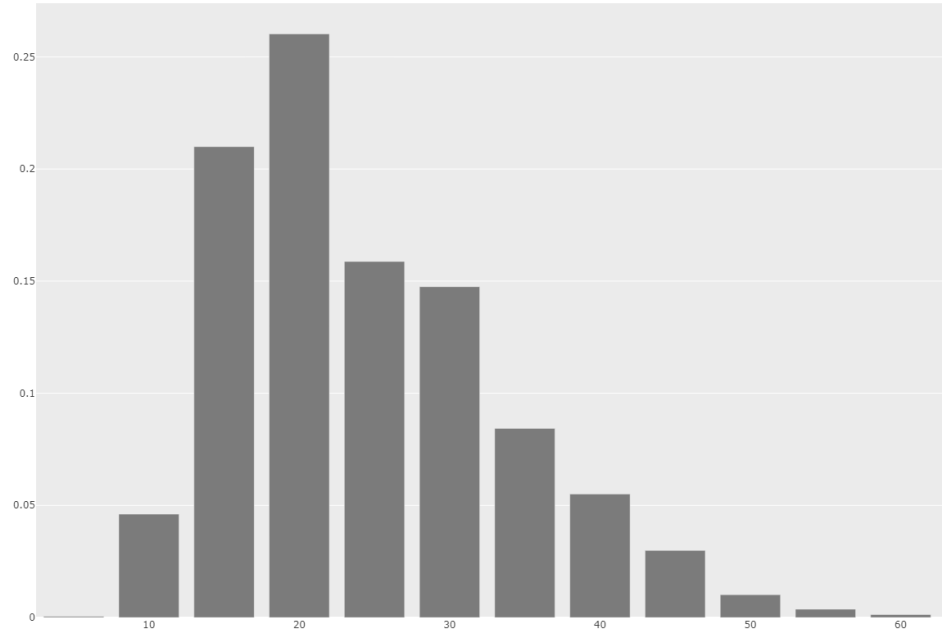


Figure 6: The histogram of apparent bird size in pixels for the NVbirds dataset.

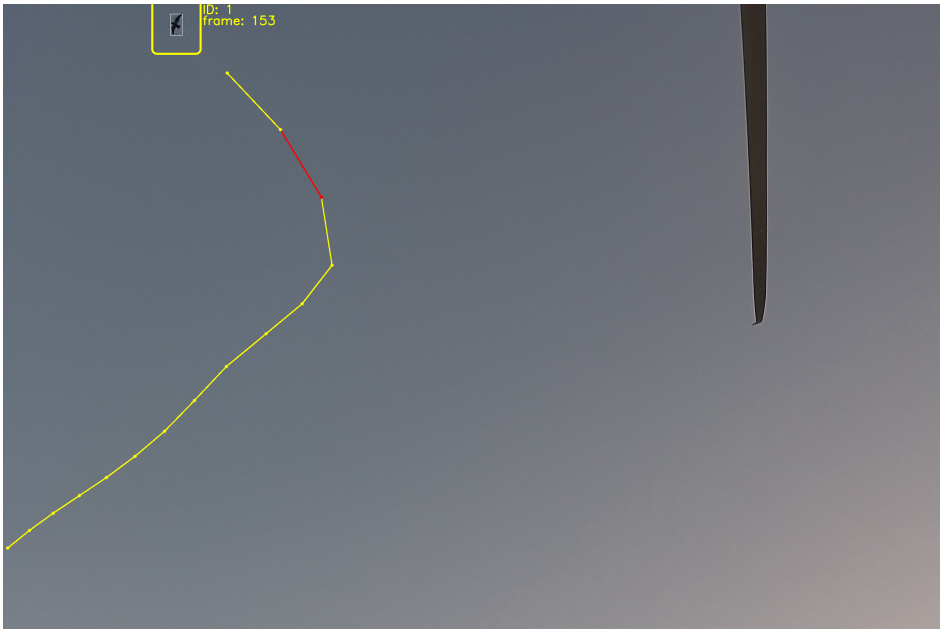


Figure 7: The annotated trajectory of an event (all video frames are superimposed in one image).

events, with a duration that varies between 3.2 and 72.8 seconds, while the median duration is 11.6 seconds. The application of the proposed inference pipeline has achieved the following 2SET metrics: $DR = 13.0\%$, $FR = 1.5\%$, $UHR = 15.8\%$, $UTR = 2.8\%$, $IR = 0.1\%$, $MR = 8.4\%$, $HOR = 0.0\%$ and $TOR = 13.2\%$.

An example of inference is depicted in Figure 8, showing the annotated event and the corresponding inference. In this case the event has an underfilled head segment with a length of 23 frames, it is followed by a true positive segment with the 128 frames, and finally there is an underfilled tail segment with 9 frames.

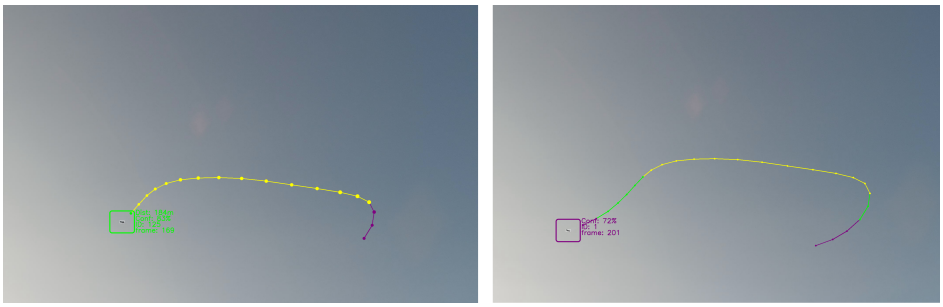

Figure 8: An example of the inferred trajectory (on the left) and the corresponding annotation (of the right).

## 4 Discussion

We have presented the pipeline for compiling the dataset of the NVbirds system. This is data engine which is actively used to this day, and constantly optimized. It has allowed us to create the NVbirds dataset, that has become an extended and rich dataset, on par with the published ones. Moreover the iterative compilation of the dataset gave us the ability to gradually shift the focus of data collection to cases that are challenging for our inference method, as the aforementioned case of blades with bended edge. Therefore we consider the implementation of a data engine as an indispensable part of any wildlife protection system.

Besides the data engine, we have reframed the task of bird detection as temporal action detection. By using appropriate metrics, we gathered refined results, and a better understanding of the errors that our inference system makes. We have seen that the frames with missed detections are almost equally divided between deleted segments and underfilled head segments. However the latter cases are easily explained by the inertia that we have introduced in the detection of the start of the event. Thus in the future we should focus on the former cases, in order to gain a considerable improvement of the CNN model. Similarly the error in overfilled tail segments is explained by the inertia in the detection of the end of the event. So it is not a sign of poor performance, as long as the event is not extended beyond one second. Concluding, we consider that the metrics for temporal action detection offer valuable insights, and they are worth becoming a widely accepted tool.

## References


[1] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[2] Thomas Berg, Jiongxin Liu, Seung Woo Lee, Michelle L. Alexander, David W. Jacobs, and Peter N. Belhumeur. Birdsnap: Large-scale fine-grained visual categorization of birds. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2019–2026, 2014.

[3] Grant Van Horn, Steve Branson, Ryan Farrell, Scott Haber, Jessie Barry, Panos Ipeirotis, Pietro Perona, and Serge Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, 2015.

[4] ZongYuan Ge, Chris McCool, Conrad Sanderson, Peng Wang, Lingqiao Liu, Ian D. Reid, and Peter I. Corke. Exploiting temporal information for dcnn-based fine-grained object classification. In *2016 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2016, Gold Coast, Australia, November 30 - December 2, 2016*, pages 1–6. IEEE, 2016.

[5] Fatima El Jaimi, Wiam Rabhi, Walid Amara, Zakaria Charouh, Houssam Benaboud, and Moudathirou Ben Saindou. Lasbird: Large scale bird recognition dataset, 2023.

[6] Wiam Rabhi, Fatima Eljaimi, Walid Amara, Zakaria Charouh, Amal Ezzouhri, Houssam Benaboud, Moudathirou Ben Saindou, and Fatima Ouardi. An integrated framework for bird recognition using dynamic machine learning-based classification. In *2023 IEEE Symposium on Computers and Communications (ISCC)*, pages 889–892, 2023.

[7] Yang Wang, Jiaogen Zhou, Caiyun Zhang, Zhaopeng Luo, Xuexue Han, Yanzhu Ji, and Jihong Guan. Bird object detection: Dataset construction, model performance evaluation, and model lightweighting. *Animals*, 13(18), 2023.

[8] R. Yoshihashi, R. Kawakami, M. Iida, and T. Naemura. Bird detection and species classification with time-lapse images around a wind farm: Dataset construction and evaluation. *Wind Energy*, 20(12):1983–1995, 2017.

[9] Hiba Alqaysi. Annotated birds datasets for object detection using deep learning, skagen and klim (version 1), 2021.

[10] Hiba Alqaysi, Igor Fedorov, Faisal Z. Qureshi, and Mattias O'Nils. A temporal boosted yolo-based model for birds detection around wind farms. *Journal of Imaging*, 7(11), 2021.

[11] Hongyu Sun, Yongcai Wang, Xudong Cai, Peng Wang, Zhe Huang, Deying Li, Yu Shao, and Shuo Wang. Airbirds: A large-scale challenging dataset for bird strike prevention in real-world airports. In *Asian Conference on Computer Vision (ACCV)*, pages 2440–2456, December 2022.

[12] Sanae Fujii, Kazutoshi Akita, and Norimichi Ukita. Distant bird detection for safe drone flight and its dataset. In *International Conference on Machine Vision Applications (MVA)*, 2021.

[13] Yuki Kondo, Norimichi Ukita, Takayuki Yamaguchi, Hao-Yu Hou, Mu-Yi Shen, Chia-Chi Hsu, En-Ming Huang, Yu-Chen Huang, Yu-Cheng Xia, Chien-Yao Wang, Chun-Yi Lee, Da Huo, Marc A. Kastner, Tingwei Liu, Yasutomo Kawanishi, Takatsugu Hirayama, Takahiro Komamizu, Ichiro Ide, Yosuke Shinya, Xinyao Liu, Guang Liang, and Syusuke Yasui. Mva2023 small object detection challenge for spotting birds: Dataset, methods, and results. In *2023 18th International Conference on Machine Vision and Applications (MVA)*, pages 1–11, 2023.

[14] Hao-Yu Hou, Mu-Yi Shen, Chia-Chi Hsu, En-Ming Huang, Yu-Chen Huang, Yu-Cheng Xia, Chien-Yao Wang, and Chun-Yi Lee. Ensemble fusion for small object detection. *2023 18th International Conference on Machine Vision and Applications (MVA)*, pages 1–6, 2023.

[15] Da Huo, Marc A. Kastner, Tingwei Liu, Yasutomo Kawanishi, Takatsugu Hirayama, Takahiro Komamizu, and Ichiro Ide. Small object detection for birds with swin transformer. In *2023 18th International Conference on Machine Vision and Applications (MVA)*, pages 1–5, 2023.

[16] Yosuke Shinya. Bandre: Rethinking band-pass filters for scale-wise object detection evaluation. In *2023 18th International Conference on Machine Vision and Applications (MVA)*, pages 1–5, 2023.

[17] Dawid Gradolewski, Damian Dziak, Milosz Martynow, Damian Kaniecki, Aleksandra Szurlej-Kielanska, Adam Jaworski, and Wlodek J. Kulesza. Comprehensive bird preservation at wind farms. *Sensors*, 21(1), 2021.

[18] Rafał Tkaczyk, Grzegorz Madejski, Dawid Gradolewski, Damian Dziak, and Wlodek J. Kulesza. Methodological selection of optimal features for object classification based on stereovision system. *Sensors*, 24(12), 2024.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing.

[20] Haroon Idrees, Amir R. Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 155:1–23, 2017.

[21] Jamie A. Ward, Paul Lukowicz, and Hans W. Gellersen. Performance metrics for activity recognition. *ACM Trans. Intell. Syst. Technol.*, 2(1), jan 2011.