Article

Advanced Computer Vision Methods for Tracking Wild Birds from Drone Footage

Dimitris Mpouziotas [1], Petros Karvelis [1,*] and Chrysostomos Stylios [2]

[1] Department of Informatics and Telecommunications, University of Ioannina, 451 10 Ioannina, Greece; mpouziotasd@gmail.com
[2] Industrial Systems Insitute, Athena RC, 263 31 Patra, Greece; stylios@isi.gr
* Correspondence: pkarvelis@uoi.gr

Featured Application: Our methodologies feature an advanced model capable of processing footage recorded in remote and difficult-to-access environments. It aims to extract valuable analytics about wildlife from the footage, such as estimating counts of wildlife, visualizing activities of detections, and rigorously tracking detections. The model takes advantage of state-of-the-art computer vision models, and its main objective is to leverage long-term protection for wildlife or any endangered species.

Abstract: Wildlife conservationists have historically depended on manual methods for the identification and tracking of avian species, to monitor population dynamics and discern potential threats. Nonetheless, many of these techniques present inherent challenges and time constraints. With the advancement in computer vision techniques, automated bird detection and recognition have become possible. This study aimed to further advance the task of detecting wild birds using computer vision methods with drone footage, as well as entirely automating the process of detection and tracking. However, detecting objects from drone footage presents a significant challenge, due to the elevated altitudes, as well as the dynamic movement of both the drone and the birds. In this study, we developed and introduce a state-of-the-art model titled ORACLE (optimized rigorous advanced cutting-edge model for leveraging protection to ecosystems). ORACLE aims to facilitate robust communication across multiple models, with the goal of data retrieval, rigorously using various computer vision techniques such as object detection and multi-object tracking (MOT). The results of ORACLE's vision models were evaluated at 91.89% mAP at 50% IoU.

Keywords: computer vision; yolo; object tracking; oracle model; small object detection

check for updates

Citation: Mpouziotas, D.; Karvelis, P.; Stylios, C. Advanced Computer Vision Methods for Tracking Wild Birds from Drone Footage. *Drones* **2024**, *8*, 259. https://doi.org/10.3390/drones8060259

Academic Editors: Humberto L. Perotto-Baldivieso and Aaron M. Foley

Received: 29 April 2024
Revised: 10 June 2024
Accepted: 11 June 2024
Published: 12 June 2024

1. Introduction

The study and conservation of wildlife have entered a new era with the integration of advanced technologies, providing researchers with unprecedented tools to monitor and understand animal behavior in their natural habitats [1]. Among these technologies, unmanned aerial vehicles (UAVs) have emerged as versatile platforms for ecological research, offering the ability to capture high-resolution imagery and video from vantage points that were once inaccessible [2].

In the past, wildlife avian surveillance of the Amvrakikos Gulf [3] was carried out through physical monitoring or by utilizing telescopes.However, both methods inappropriate for surveying wildlife. Physically approaching islets to monitor birds caused stress, potentially leading them to break their eggs and suffer long-term detrimental effects on their population. Meanwhile, using a telescope may keep animals unharmed, due to the distance from the islets to the available land, this can also pose challenges due to the angle and visibility, making it even more difficult to survey the behavior of the wildlife.

In the realm of ornithological research, the utilization of UAVs holds immense potential for tracking and studying bird species in their natural environments [1,4]. When UAVs are coupled with automated computer vision methods, particularly those based on deep learning architectures, researchers can automate the identification and tracking of bird species in drone-captured footage, thereby overcoming the limitations of traditional manual tracking methods and eliminating the dangers of physical monitoring or the limitations of using a telescope. Figure 1 depicts an image retrieved above an islet at a safe altitude and containing wild birds.

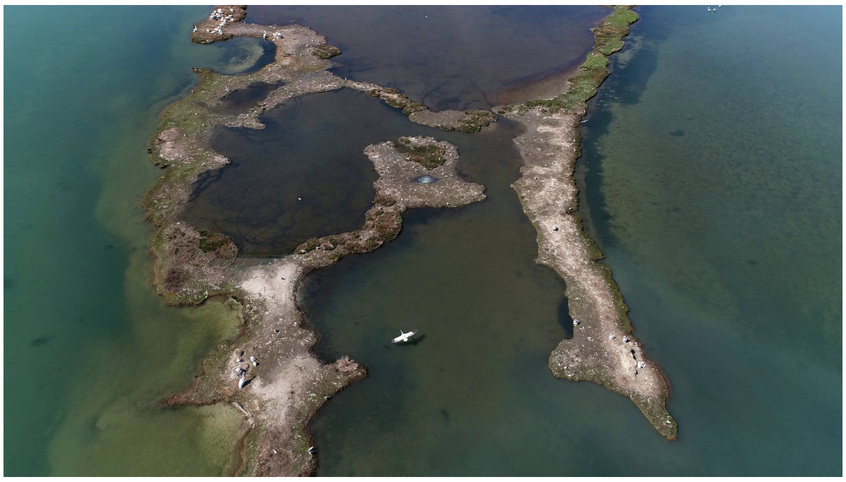


Figure 1. An image depicting wildlife surveillance using drones over an islet, recorded at 40 m altitude.

During the surveillance of the islands in the Amvrakikos Gulf, we acted in accordance with precise protocols based on the nesting seasons [5] of Dalmatian pelicans (*Pelecanus crispus*) [6]. Their nesting seasons typically span from mid-January to mid-June. We ensured strict compliance with the safety protocols while surveying each island, to refrain from disturbing the wildlife during their breeding season. Ensuring the safety of the avian wildlife of the Amvrakikos remained our uppermost concern, simultaneously our objective was to conduct a comprehensive survey of the wildlife population. This effort aimed to facilitate the development of action plans aimed at mitigating any damages from avian influenza that occurred in the past [7], leveraging the collection of knowledge and various tools throughout the study.

During each flight over the islets, two key parameters added complexity to our computer vision task. The drone's movement from one point of the island to another was continuous, in addition to some target objects being in motion. This added complexity posed challenges for this task; however, it also allowed us to track wildlife across the entire island in a single sweep. Computer vision with UAVs raises new questions, such as the effects of camera motion, also known as motion blur, resulting in less than optimal performance of the detection model. Figure 2 illustrates our flight plan methodology over wildlife nests, alongside details regarding the altitude safety protocols.

Some additional challenges we faced in using drones for wildlife surveillance included the interaction with the birds. The lower the altitude and the louder the drone, the more likely the birds would be disturbed, either by flying away or moving away from the drone. Furthermore, if birds feel threatened, they may collide with or attack the drone, creating a hazardous situation. In addition to these challenges, the 'Drone-vs-Bird Detection Grand Challenge' [8] highlighted the complexity of distinguishing drones from birds in video sequences, especially when drones operate in bird-populated environments, further complicating wildlife surveillance efforts.

The higher the altitude of the drone from the ground, the greater a challenge it became for the model to precisely track wildlife. Furthermore, detecting small- to medium-sized objects from high-resolution footage posed a challenge, due to the diverse number of factors that needed to be considered.

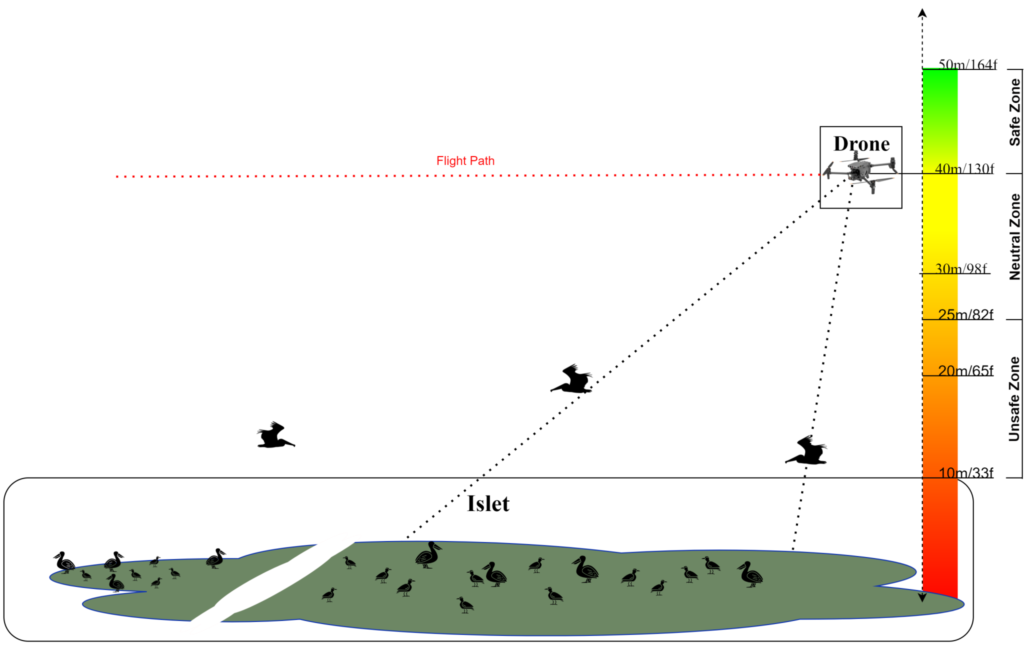


Figure 2. Drone surveillance guidelines and altitudes during nesting season.

However, traditional tracking methods have often been limited by their reliance on manual annotation and labor-intensive data processing. The advent of computer vision algorithms, particularly those based on deep learning architectures, has alleviated these challenges. These methodologies empower researchers to automate the identification and tracking of bird species in drone-captured footage, providing a more efficient and scalable solution.

Recent years have witnessed a surge in the application of state-of-the-art computer vision techniques for object detection, recognition, and tracking [9]. Notably, frameworks such as you only look once (YOLO [10]), single-shot multibox detector (SSD [11]), and faster R-CNN (region-based convolutional neural network [12]) have demonstrated remarkable success in real-time object detection, laying a solid foundation for their adaptation to ecological research.

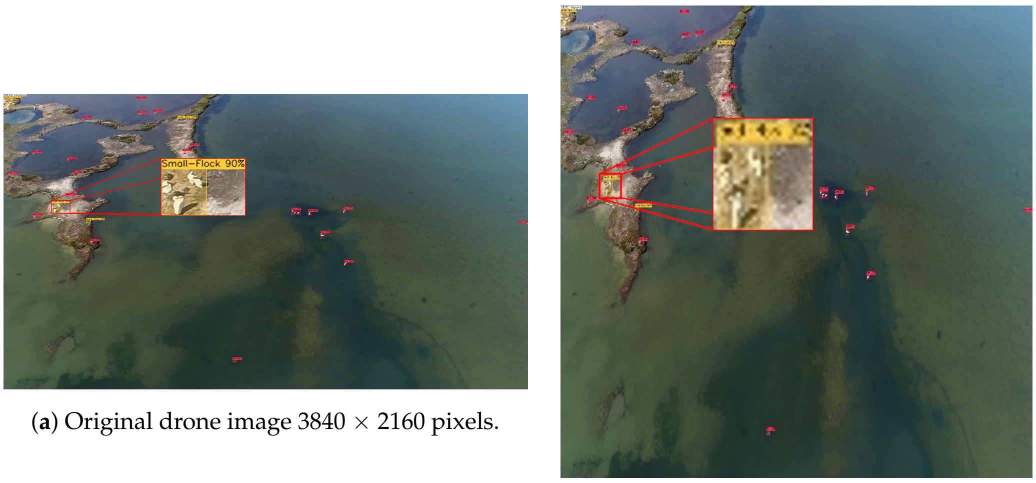
The current state-of-the-art models in computer vision for avian tracking involve the integration of convolutional neural networks (CNNs) trained on extensive datasets of annotated bird imagery. These models excel at recognizing complex patterns and shapes, enabling the accurate identification and tracking of individual birds or groups within a given scene [13]. By harnessing the power of deep learning, researchers can extract detailed information about bird movements, spatial distributions, and social interactions from vast amounts of aerial footage.

Previous studies that explored the application of state-of-the-art computer vision models on avian datasets such as VLIZ [14] examined footage or images similar to the ones in our study. Another study that examined the same problem under a specialized dataset such as ours focused its vision problem on high-altitude top-down detection of wild birds [9] using computer vision techniques up to YOLOv3 [15]. However, many of these studies did not employ remote wildlife surveillance using drones from a bird's-eye view perspective. Furthermore, datasets that focus on wild birds such as the Macaulay Library [16] and Caltech [17] are widely known datasets with the vast majority of bird species. While both serve as a valuable reference for our use case, our approach was more specialized, necessitating consideration of several additional parameters to effectively detect and track wildlife using drones. We focused our attention on modern technologies such as the detection models YOLOv7 [18] and YOLOv8 [19] and investigated their suitability for our objective.

Achieving high precision, as emphasized in similar studies [20], is a challenging task. In this work, we leveraged recent advancements in the field and explored cutting-edge deep learning architectures and techniques specifically designed for high-precision inference in a custom dataset featuring small objects.

The dataset we built consisted of high-resolution drone footage of about 3840 × 2160 109 pixels (width × height). Feeding such large images directly into any model with a significantly smaller input size, typically around 640 × 640 pixels (Width × Height), results in a substantial loss of information. This is due to the model automatically scaling the image to fit the input size of the model. Our objective was to make the models capable of using their full potential, without any loss of information during inference.

Figure 3, illustrates the information loss of scaling an image down to the model's network size. We can observe a significant loss of information and detail in the scaled image compared with the full-scale image.



(**a**) Original drone image 3840 × 2160 pixels.

(**b**) Scaled image 640 × 640.

Figure 3. Original drone image image scaled from 3840 × 2160 pixels down to 640 × 640 pixels.

We built upon the use of detection models for the task of identifying diverse scales of objects, using high-precision detection methods. Additionally, we explored the use of multi-object tracking (MOT) techniques. Our objective was the continuous tracking of multiple individual objects across multiple frames in a video. Object tracking endeavors to assign and maintain a unique identifier (ID) across multiple frames, as established in [21].

The complex interplay between object detection and tracking is of great importance for successful multi-object tracking. While robust object detection is the foundation of this task, various factors beyond detection call for consideration [22]. One such crucial aspect is the ability of the tracker to sustain an object's ID, even when the object detection model fails to detect an object between consecutive frames; this technique is called re-identification. Re-identification aims to compensate for any data loss that may occur during inference due to missed detections by the object detector.

This manuscript delves into the intricacies of applying cutting-edge computer vision methods to avian wildlife monitoring, elucidating the nuances of YOLO and other relevant frameworks in the context of UAV-captured data. The subsequent sections will expound upon the specific adaptations and optimizations carried out to tailor these algorithms for the challenges posed by bird tracking, considering factors such as varying lighting conditions, diverse bird species, and complex natural environments.

As we navigate through this exploration of computational vision tools, our aim is to not only showcase their current capabilities but also to inspire further innovation in the realm of ecological monitoring. The synthesis of UAV technology and advanced computer vision methodologies not only augments the precision of avian tracking but also opens avenues for interdisciplinary research at the intersection of computer science and ecology, fostering a deeper understanding of avian behaviors and ecological dynamics.

The overarching goal of this research is to contribute to the growing body of knowledge on avian ecology by leveraging the capabilities of UAVs and sophisticated computer vision algorithms. As biodiversity faces increasing threats and challenges, understanding the dynamics of wildlife populations becomes crucial for effective conservation strategies. This manuscript elucidates the development and application of novel computational tools designed to track and analyze wild bird species, with a focus on their movements, group dynamics, and habitat preferences.

Through this interdisciplinary approach, merging insights from ecology, computer science, and remote sensing, our research contributes to the advancement of wildlife monitoring methodologies. The findings presented herein not only offer valuable contributions to the scientific community but also pave the way for enhanced conservation strategies that are grounded in a deeper understanding of avian behaviors and ecological interactions.

Our study presents several key contributions to the field of wildlife monitoring using advanced computer vision techniques:

- We introduce a novel state-of-the-art computer vision model, ORACLE, designed to enhance the accuracy and efficiency of wildlife bird tracking from drone footage.
- The ORACLE model demonstrated exceptional object detection capabilities, with a mean average precision (mAP) of 91.89% at 50% intersection over union (IoU), addressing the challenge of detecting small- to medium-sized wildlife from high altitudes.
- Our methodology incorporates advanced multi-object tracking techniques that maintain consistent identification numbers across frames, which is crucial for long-term behavioral studies and population monitoring.
- ORACLE facilitates detailed behavioral and population analytics, which are critical for conservation efforts, providing environmentalists and researchers with valuable insights into wildlife dynamics.
- The application of our model extends to remote and inaccessible regions, demonstrating its robustness under challenging environmental conditions where traditional monitoring methods are not feasible.

Furthermore, the impact of avian influenza in 2022 on Dalmatian pelicans was a severe disaster [7]. Our goal is to observe the wildlife and the environment of the Amvrakikos Gulf in Greece and to conserve it in the long run with the assistance of our model.

2. Materials and Methods

This section provides extensive information regarding the methodologies and implementations used to develop not just ORACLE, but also a high-accuracy computer vision model capable of a robust object detection system. Our primary emphasis lay on addressing the challenges of a specialized problem regarding computer vision in high-resolution drone footage with small- to medium-sized objects.

2.1. YOLO Models and Their Performance

A key objective of this research was not only to inform but to conduct an excessive review of several detection models. We chose to focus specifically on YOLO (you only look once) models for several reasons. Firstly, YOLO models are renowned for their efficiency and speed in object detection tasks, making them well-suited for real-time applications. Additionally, YOLO architectures have demonstrated strong performance across various datasets, including the challenging Microsoft COCO dataset [23], which contains a wide range of object classes and scenarios.

We aimed to thoroughly assess the accuracy and performance of different YOLO models. To achieve this, we conducted extensive evaluations using various YOLO architectures and sizes on the Microsoft COCO dataset. This dataset is widely recognized and utilized for benchmarking object detection algorithms, due to its large-scale and diverse collection of images. By testing multiple YOLO models on the COCO dataset, we were able to gather comprehensive insights into their capabilities and limitations. This rigorous evaluation process allowed us to compile a comprehensive set of results, enabling us to make informed decisions about which models performed best under different conditions and tasks.

In the following section, we explored several models with various sizes and architectures by evaluating them under the Microsoft COCO dataset and ultimately compiled this information into a comprehensive set of evaluation results.

The YOLO models utilize a size-based naming convention for their sub-models: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra large). Furthermore, some frameworks contain additional architectures to train based on the specialization of the dataset. The integration of an architecture in a model may vary based on the problem in question. Architectures such as P6 prioritize medium- to large-sized detections, whilst P2 and P3 architectures prioritize small- to medium-sized detections. Ultralytics contains an active repository with these architectures available and ready to use at any time, from P2 up to P7.

Due to the great number of models already available for use, it is difficult to determine which model fits best for a specialized dataset such as ours. As such, this part of the study delved into a comparative analysis of the models (v5, v7, and v8), including their sub-models. As mentioned previously, we evaluated their overall performance (mAP) on the MS COCO dataset [23] to clarify several key points in employing specific model sizes or architectures in comparison to the COCO dataset and ours.

The MS COCO dataset consists of annotations sized from small to extra-large objects. Table 1 describes the annotation size distributions in COCO. The most dominant size range of annotations are the small ones at a total of 41.43%, whilst the large detections consist of 24.25% of the overall dataset.

Table 1. MS COCO annotation size percentages as opposed to the image sizes. Table information cited from a rigorous study performed with the COCO dataset for small object detection [24].

Annotation Size	Size Range (%)	Annotation Distribution (%)
Small	0–0.3	41.44%
Medium	0.3–3	34.32%
Large	3–100	24.24%

In all our experiments, we made sure to evaluate each model with little to no imbalances. We achieved this by manually tuning some of the settings during the evaluation of the models. This was achieved by using the same confidence and IoU threshold [25] values for the post-processing function non-maximum-suppression (NMS) [26].

All models listed in Table 2 were evaluated using the pycocotools library from Microsoft COCO [23]. It should be noted that many YOLO models are distributed across multiple frameworks, such as Darknet [27], Ultralytics [19], and PyTorch [28]. This diversity might produce slight variations, typically within a range of 1–2%, in the evaluation results.

The evaluation results of YOLOv5, YOLOv7, and YOLOv8 in Figure 2 serve as an indication of the overall performance of each model on a dataset such as MS COCO [23].

The trained models of YOLOv5 with the P6 architecture produced satisfactory results, with YOLOv5x6 performing with the highest overall mAP compared to the other models in the COCO dataset. The sub-models of YOLOv5 with P6 prioritize medium, large, or extra-large detections, which explains the high accuracy on the COCO dataset.

YOLOv7 and its sub-models demonstrated optimal performance when utilizing the P6 architecture. However, it is noteworthy that large-sized models may exhibit much less improvement in overall mAP compared to some of the smaller-sized models. Despite that, larger-sized models play a crucial role in achieving extensive detection results in certain challenging tasks.

Table 2. Evaluation results of several YOLO models in various frameworks, starting from YOLOv5, up to YOLOv8 (Excluding v6). The mAP percentages colored red correlate to the lowest value in that column produced by the corresponding model (v5, v7, and v8), whilst the green percentages represent the highest value.

Model	Params (M)	FLOPs (G)	mAP$_{val}$ (50%)	mAP$_{val}$ (75%)	mAP$_{val}$ (50–95%)
YOLOv5n [19]	1.9	4.5	49.17%	36.99%	34.07%
YOLOv5s	7.2	16.5	59.42%	46.96%	42.84%
YOLOv5m	21.2	49.0	65.55%	53.60%	48.85%
YOLOv5l	46.5	109.1	68.74%	57.07%	52.19%
YOLOv5x	86.7	205.7	69.75%	58.13%	53.22%
YOLOv5n6	3.2	4.6	57.76%	46.04%	41.73%
YOLOv5s6	12.6	16.8	65.44%	53.35%	48.44%
YOLOv5m6	35.7	50.0	70.24%	58.82%	53.53%
YOLOv5l6	76.8	111.4	72.23%	60.98%	55.75%
YOLOv5x6	140.7	209.8	73.27%	62.24%	56.97%
YOLOv7 [18]	36.9	104.5	69.5%	55.86%	51.27%
YOLOv7x	71.3	189.7	70.90%	57.77%	53.00%
YOLOv7w6	70.4	89.9	67.77%	54.49%	50.16%
YOLOv7e6	97.2	128.6	69.26%	56.00%	51.62%
YOLOv7d6	133.7	175.4	69.58%	56.61%	51.97%
YOLOv7e6e	151.7	210.5	70.54%	57.63%	52.82%
YOLOv8n [19]	8.7	3.2	52.08%	40.30%	37.3%
YOLOv8s	11.2	28.6	61.30%	48.44%	44.9%
YOLOv8m	25.9	78.9	66.72%	54.55%	50.2%
YOLOv8l	43.7	165.2	69.45%	57.63%	52.9%
YOLOv8x	68.2	257.8	70.69%	58.87%	53.9%

In contrast to YOLOv7, which contains a limited number of architectures for small-object detection, YOLOv8 boasts a diverse range of models optimized for this task. We directed our focus towards YOLOv8 due to its exceptional efficiency in dynamically detecting small- to medium-sized objects with minimal inconsistencies.

Moving forward, our study expanded further in a methodology regarding small object detection. We examined the techniques employed to develop a tiling algorithm, as well as the challenges encountered alongside this technique. Subsequently, our study evaluated several models using a custom dataset that prioritizes high-resolution images with small-sized detections.

2.2. ORACLE-Five-Layered Model for Advanced Object Detection & Tracking

Our efforts in implementing a model capable of advanced object detection and tracking, as well as the extraction of valuable analytics, began by developing a five-layered model. During the development of this study, we implemented an advanced algorithm capable of processing footage recorded using high-grade industrial drones. The model's objective is to receive as input, footage recorded in wildlife environments and to extract important analytics observed from the wildlife, in the form of statistics of data visualizations. This model targets data analysts and environmentalists, to process and understand the activity of ecosystems.

ORACLE is a five-layered model capable of applying advanced object detection and tracking techniques, along with the extraction of valuable information and analytics from drone footage. It is capable of extracting data, such as estimating counts of wildlife, exporting videos using various visualization methods, adding masks to distinguish an object from the background, and more. ORACLE uses state-of-the-art computer vision models prioritizing small- to medium-sized objects. Figure 4 illustrates the sequential processes executed by our model to facilitate the extraction and exportation of significant analytics.

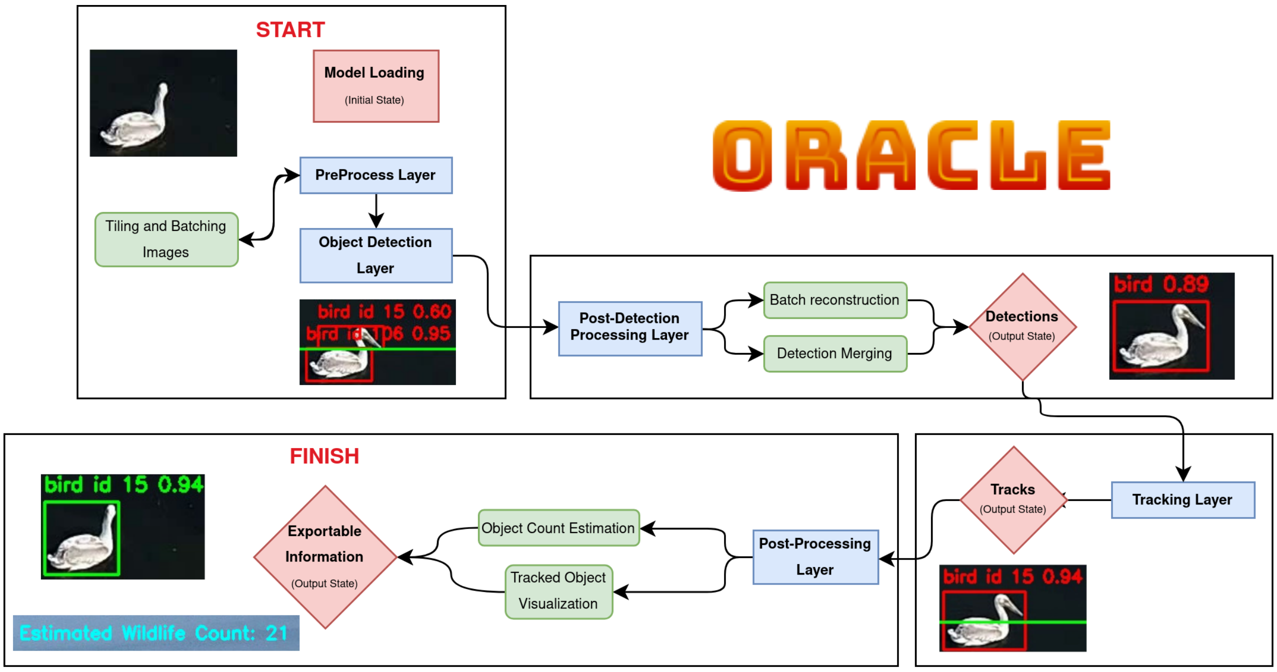


Figure 4. The process flow of the ORACLE model.

Layer One: Pre-Processing Layer

The pre-processing layer works by dynamically loading the models based on their individual framework. Additionally, each frame is enhanced with the use of the gamma correction technique [29], to reduce sunlight degradation. The footage is loaded in the form of a dataset. Each frame is processed individually and is later tiled during inference under a tensor that takes advantage of cuda technology [30], in order to achieve faster inference [31]. Figure 5, depicts a frame loaded into the dataset. This image is later processed in the second layer, the object detection layer.

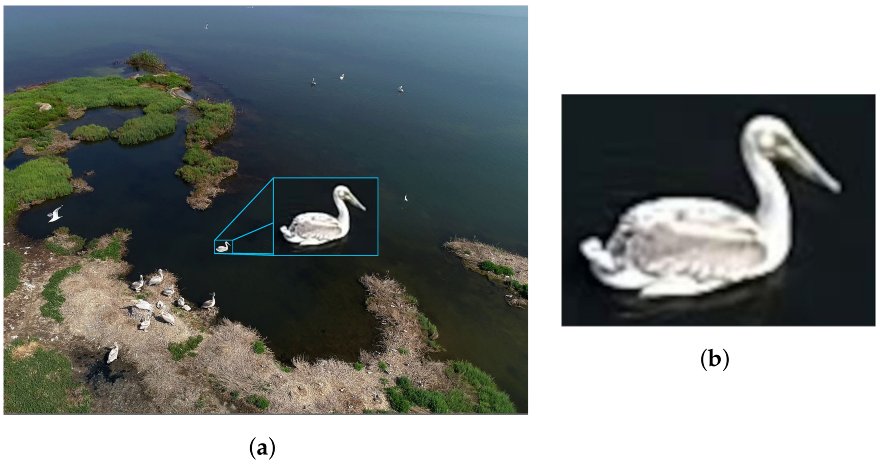


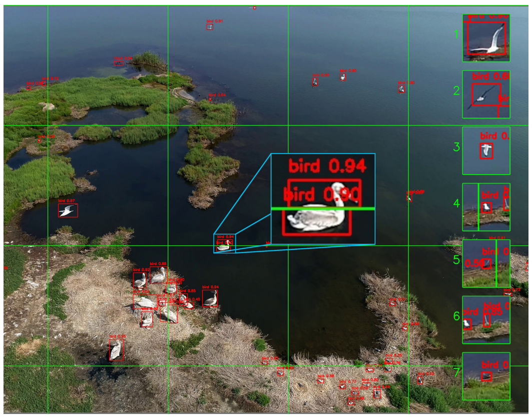
Figure 5. Layer one pre-processing: input frame example with a visible object to detect. Image frames located at Koronisia, Balla (Location at Koronisia, Nisida Balla [GPS Pin]), 22 June 2023. (**a**) Original drone image 3840 × 2160 pixels (width by height). (**b**) target object.

Layer two: object detection layer

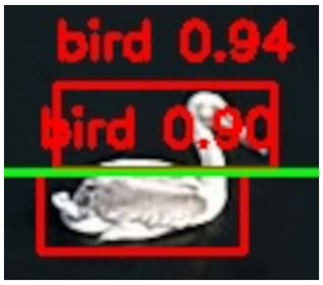
The object detection layer works by selectively applying inference based on the model version. This layer processes each frame individually, and this is repeated throughout the entire video. Each frame is segmented into multiple pieces and then passed onto the detection model [32].

This layer was developed to dynamically utilize specific inference functions based on the model version and the platform it was published in. Models published and developed in Ultralytics utilize SAHI inference [32]. Models published and developed in PyTorch take advantage of our tiling algorithm based on DarkHelp [33].

Figure 6 depicts the results using our tiling algorithm for most PyTorch models.



(**a**)



(**b**)

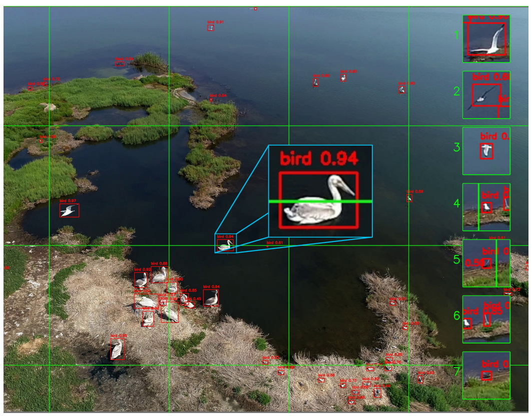
Figure 6. Layer two object detection output: target object depicts two detections. The green grid showcases the processed tiles. (**a**) Original drone image 3840 × 2160 pixels (width by height). (**b**) Target object layer two output.

Layer three: post-detection processing layer

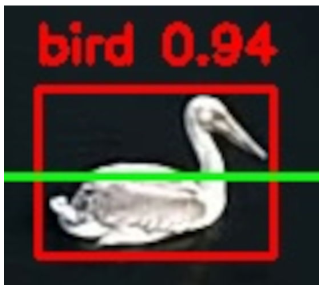
The post-detection processing layer merges the detections that were split during inference with tiling. This layer attempts to merge the detections that were split due to the image tiling algorithm, as depicted in Figure 6b.

The post-detection processing layer is only activated when using a PyTorch model to merge any detections that were split during inference due to tiling.

Figure 7b depicts the result of the post-detection processing layer, which attempts to merge the two detections resulting from the previous layer, Figure 6b.



(**a**) Output of layer three of ORACLE.

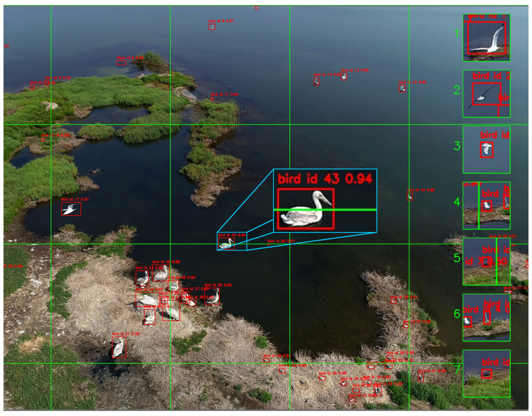


(**b**) Target object layer three output

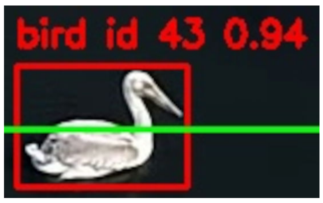
Figure 7. Layer three: post-detection processing layer: detection merging.

Layer four: tracking layer

The tracking layer is dedicated to the tracking model, which handles the outcomes derived from the previous layer. Its objective is to assign an identifier to each detected object and consistently retain this identifier across multiple frames. Figure 8 showcases the visible results of the model, with each detection displaying information such as the class name, identification number, and prediction confidence.



(**a**) Output of layer three of ORACLE.



(**b**) Target object layer four output

Figure 8. Layer four post-detection processing: detection merging.

In Figure 8, a green mask overlays the object we tracked. The goal of creating an overlay of the object is to target pixels that depict the tracked object's surface. Mask overlaying allows us to process those pixels in the future and give us a more in-depth analysis (e.g., targeting the exact temperature of the green pixels). We achieved this technique by applying color thresholding to each detection individually using a specific pixel value distinguishable on the object's surface compared to the background. As evident in the Figure 8b, there is a noticeable contrast that makes the masking process far easier.

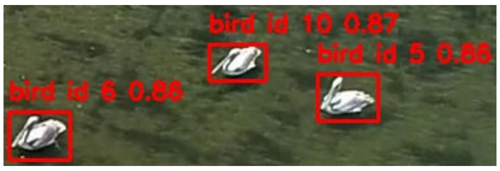
Layer five: post-processing layer

The post-processing layer is the final layer where ORACLE generates visualizations, as well as extracting valuable information regarding the environment depicted in the footage. In this layer, ORACLE uses several algorithms capable of producing insightful information from the data. An example of a post-processing layer algorithm is the estimation of wildlife detected within the footage.

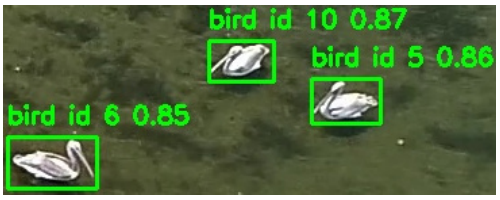
We can observe from the previous Figures 6–8 that during visualization we created a zoom effect for the oldest active tracks based on life-span sorted by ID. This visualization method was primarily used for better observation of the objects.

Figure 9 displays three tracked objects in four different timelines of a video. With the application of advanced object-detection techniques alongside tracking methods, we can observe how those models were capable of consistently maintaining the same track IDs for the same objects over extended periods of time.

Using ORACLE, we produced a video demonstrating the development of the model ORACLE Video Result.

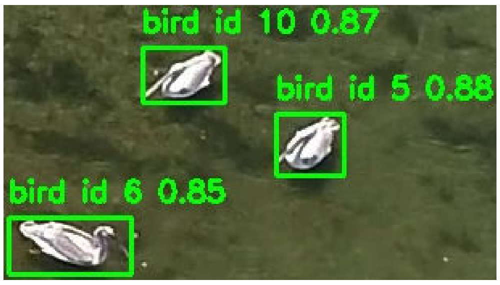


(**a**) Time-frame: 1 s

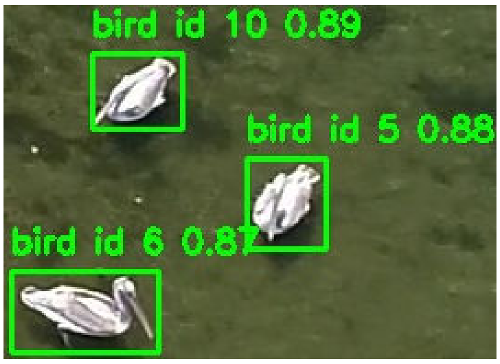


(**b**) Time-frame: 4 s

Figure 9. *Cont.*



(**c**) Time-frame: 8 s



(**d**) Time-frame: 12 s

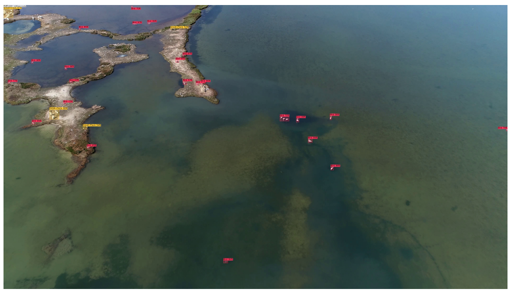
Figure 9. ORACLE fifth layer output in four different time-frames in a video.

2.3. Image Tiling

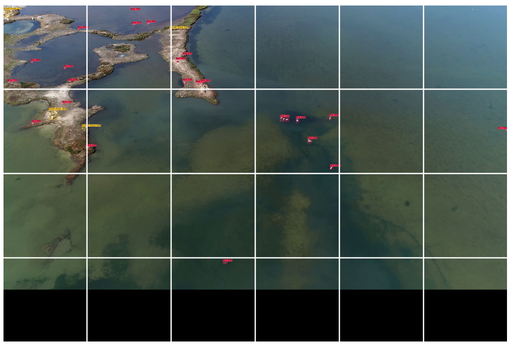
Our approach involves segmenting each full-scale image into smaller tiles and feeding them individually into the model. This methodology is consistently applied during both the inference and training stages, to sustain the same dimensions. However, existing libraries like PyTorch offer limited algorithms for specific models like YOLOv7. To address this issue, we developed our own custom tiling algorithm inspired by the implementation in DarkHelp [33]

During inference, the image is segmented into multiple tiles, and each tile is processed by the model to generate individual results. Subsequently, the tiled detections are stitched back together to reconstruct the complete image. This approach not only reduces memory consumption but also ensures that no information is lost during inference.

Given that our target network size is 640 $\times$ 640, a 4 k image should ideally be tiled approximately 20 times. However, considering the dimensions of the image and the network size, a portion of the image will not be present in the tiles $(3840 * 2160)/(640 * 640) = 20.25$. To address this issue, we padded the remaining information of the image, to compensate for any information that might be lost during inference. Figure 10 visually demonstrates this process.



(**a**)



(**b**)

Figure 10. Application of the image tiling technique in a high-resolution image (4 k). Tiling the image resulted in a total of 24 images sized at 640 $\times$ 640 pixels. (**a**) High-resolution image 3840 $\times$ 2160 (4 k) ready to be tiled. (**b**) A composition of all the tiles of an image, created by our tiling algorithm, including the padded space.

Image tiling maximizes the model's potential for higher-resolution images; however, many challenges are faced when using this technique. The larger the resolution of the images, the more tiles are segmented, leading to slower inference times.

A substantial issue we faced using image tiling was when we attempted to segment an image into multiple tiles and subsequently load each tile individually into the model for inference. This led to detections being split across tiles when the image was reconstructed. To compensate for this issue, we developed a detection merging technique based on the image tiling implementation in DarkHelp [33]. This detection merging technique works by sorting the candidate and non-candidate detections. Candidate detections, Figure 11, are the detections that are closest to the tiles based on a tile edge factor. The default tile edge factor was set to `0.3`.

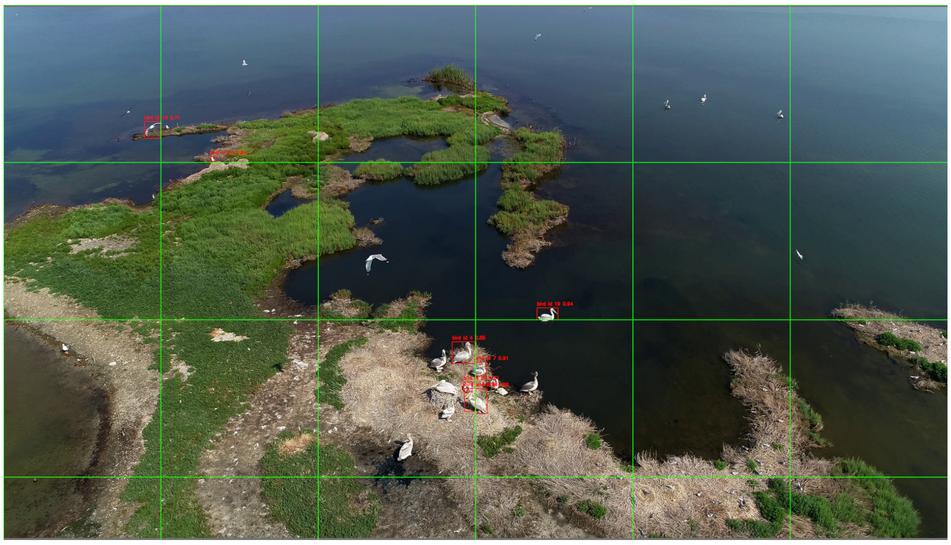


Figure 11. The candidate detection closest to the tiles for merging.

The determining factor for if a detection is near the edge of a tile is calculated using the distances of the detection in all directions, as opposed to the distance to the edges of each tile. If either of these distances are smaller than a minimum horizontal/vertical threshold, then the detection is a candidate. Equation (1), depicts the method used to calculate the minimum threshold values, including the distances.

$$\begin{aligned} \text{Minimum Horizontal Dst} &= \text{tileEdgeFactor} \times \text{det\_width} \\ \text{Minimum Vertical Dst} &= \text{tileEdgeFactor} \times \text{det\_height} \\ \text{Top Distance} &= \text{Det\_y1} - \text{tileSize} \times (\text{tileRow} + 1) \\ \text{Left Distance} &= \text{Det\_x1} - \text{tileSize} \times (\text{tileCol} + 1) \\ \text{Bot Distance} &= \text{tileSize} \times (\text{tileRow} + 1) - \text{Det\_y2} \\ \text{Right Distance} &= \text{tileSize} \times (\text{tileCol} + 1) - \text{Det\_x2} \end{aligned} \tag{1}$$

Det_width/height represents the width or height of the detection. TileEdgeFactor by default was set to 0.3. Top/left/bot/right distances were calculated based on the tile location and $(\text{tileSize} \times (\text{tileRow or tileCol} + 1))$.

After calculating the relative distances of the detections to the tiles, they are compared with each other to determine whether detection is a candidate or non-candidate. Once the candidate detections have been found, they are then merged based on a merging factor, that is typically set to `1.35`. Equation (2) (defines lhs_plus_rhs as the subtraction of the areas of the lhs and rhs bounding boxes, multiplied by a merge factor of 1.35, whereas union refers to the union of the two rectangles provided by the detections):

$$\begin{aligned} \text{lhs\_plus\_rhs} &= (\text{lhs\_area} + \text{rhs\_area}) \times \text{mergeFactor} \\ \text{UnionRect} &= \text{Union(lhs\_det\_bbox, rhs\_det\_bbox)} \end{aligned} \tag{2}$$

lhs_area: Area of the Left-Hand-Side Bounding Box (Detection);
rhs_area: Area of the Right-Hand-Side Bounding Box (Detection)

Finally, the logic behind whether two detections will be merged is determined using Equation (3).

$$\text{MergeDetections} = \begin{cases} \text{True} & \text{if } lhs\_plus\_rhs > \text{Union Area} \\ \text{False} & \text{otherwise} \end{cases} \tag{3}$$

Figure 12 is a visual representation describing the comparison across several detections. If two detections are in proximity to the edges of two tiles and their combined areas times a factor is greater than the union of both, then the detections are merged, otherwise the rhs detection is not merged with the lhs detection. The red squares outline the slicing of two tiles, the green rectangles represent the split detections resulting from image tiling, whilst the orange-dotted rectangle represents the union of the two detections in question.

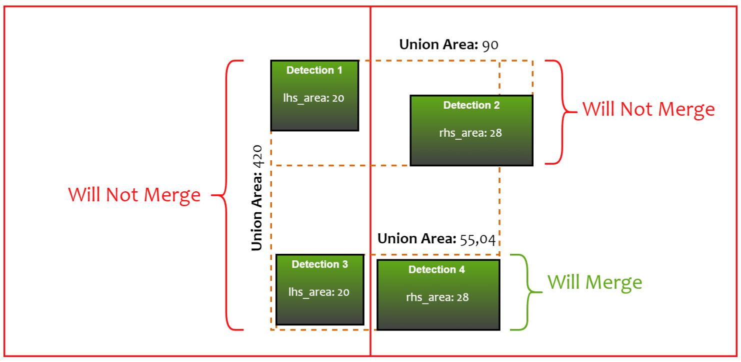


Figure 12. Descriptive image of the detection merging technique.

Additionally, the pseudocode Algorithm 1 of the merging technique thoroughly describes the analytical steps taken to merge the candidate detections.

Algorithm 1 Detection Merging

```
function MERGE_DETECTIONS(detections, tile_rect_factor)
    result ← []
    checked_indices ← {}
    for each lhs_index, lhs_pred in detections do
        if lhs_index is in checked_indices then
            continue
        end if
        append lhs_index to checked_indices
        merged ← False
        for each rhs_index, rhs_pred in detections do
            if rhs_index is in checked_indices or rhs_index equals lhs_index then
                continue
            end if
            calculate lhs_area and rhs_area
            create union_rect from lhs_rect and rhs_rect
            calculate union_rect_area
            lhs_plus_rhs ← (lhs_area + rhs_area) * tile_rect_factor
            if union_rect_area $\leq$ lhs_plus_rhs then
                create new_pred based on union_rect, lhs_pred and rhs_pred
                append new_pred to result
                mark rhs_index as checked
                merged ← True
                break
            end if
        end for
        if not merged then
            append lhs_pred to result
        end if
    end for
    return result
end function
```

Upon completion, each combined detection, alongside the detections with no neighboring detections for merging, is appended to a detection results list. An illustration of this technique can be observed in Figure 13.

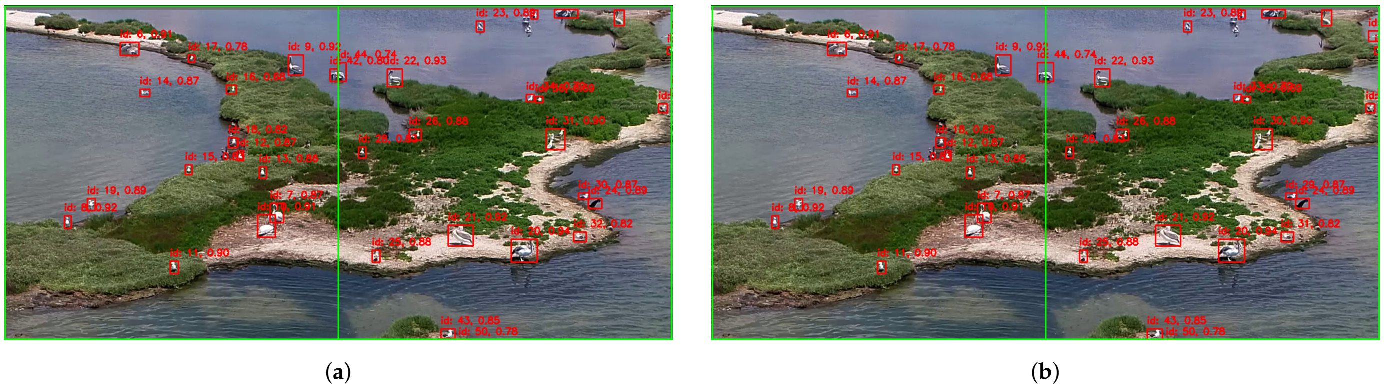


Figure 13. The effect of Image tiling in object detection in large-scale images. The green line represents the split between the two tiles, whilst the red boxes represent the detections. (**a**) Target object split in two detections due to image tiling. (**b**) Successfully merged target object after being split by the two tiles.

2.4. Model Fine-Tuning

Model fine-tuning is a standard process of developing high-precision detection models, such as the one we aimed to achieve for small object detection. Wildlife surveillance via drones poses a great challenge, due to the variety in the shapes and forms in which the birds may appear (e.g., spread/wings, different angles, different sizes, etc.). Fine-tuning aims to address these issues by enhancing the quality and quantity of the dataset and training, thereby improving the accuracy of the detection models.

There are two common fine-tuning methods: one involves enhancing the dataset through image augmentation techniques and the other focuses on fine-tuning the model by adjusting the augmentation parameters used during training. In addition to these, transfer learning is another method that significantly improves model performance by adapting pre-trained models to new tasks (datasets).

Our first approach in fine-tuning the YOLO detection model was through the application of transfer learning [34]. This method significantly enhances a model's accuracy by refining its detection capabilities from one dataset to another. Transfer learning is generally more effective than training from scratch, leading to substantial improvements in a model's overall performance.

Our second approach involved further enhancing the dataset through image augmentation techniques. Specifically, the application of zoom and crop significantly increased the dataset size by applying random zooms or crops to the images. The resulting images were resized to match the network's input size, ensuring consistency across all images.

Our final approach was to adjust the hyper-parameters during training. Training a detection model such as YOLOv8 [19] allows users to manually edit augmentation parameters. Our primary focus was to adjust some of these parameters to achieve higher performance in the task of detecting small objects.

Our final approach was to tune the augmentation hyper-parameters during training, with the goal of increasing accuracy. In the case of small object detection, adjusting certain parameters will assist the model to either have more or less variety during training, therefore increasing the performance of various detection models. Parameters such as mosaic, scale, and fliplr [35] generally have a positive effect in creating more variety during training; however, mixup will negatively impact the accuracy for small object detection tasks. Our primary focus was to adjust the parameters that applied augmentation to the image. Table 3 depicts the various adjustments to the hyper-parameters made across the frameworks.

Table 3. Fine-tuned hyper-parameters during training.

Hyperparameter	Default Value	Fine-Tuned Value	Description
mosaic	0.0	1.0	Mixes multiple randomly cropped images together into a new image. Figure 14
mixup	0.15	0.0	Blends pairs of images and their labels and creates a new image. (Reduced to prevent inconsistencies.)
scale	0.9	0.5	Scales the image based on a gain factor, simulating objects at different distances. (Decreased to match parameters across frameworks as well as decrease the gain of this augmentation algorithm.)
fliplr	0.5	0.4	(Flip left-right) Flips images from left to right randomly based on the given value.
cls	0.5	0.3	The weight of the classification loss in the total loss function, affecting the importance of correct class prediction relative to the other components.

Figure 14 displays the training batch of 16× images used to train the detection models whilst utilizing various augmentation hyper-parameters such as mosaic.

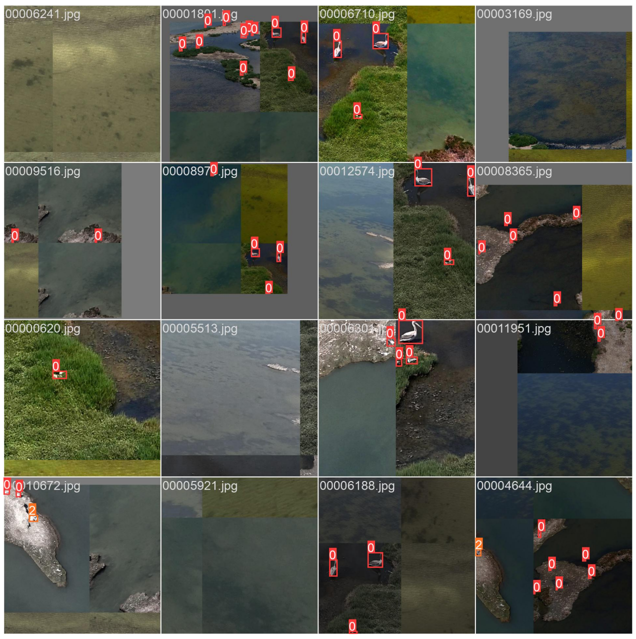


Figure 14. Training batch of our dataset containing images augmented using the mosaic algorithm.

2.5. Multi-Object Tracking Models

During inference, we used the highest performing advanced multi-object tracking, (MOT) model OSNet [36,37]. OSNet was trained on the MSMT17 dataset [38], which consists of a combination of various datasets perfect for our use-case of re-identification. They were trained on a large-scale dataset with small to medium-sized detections perfect for our use case. The tracker we used for all our use cases was DeepOC-Sort [39], an advanced algorithm prioritizing MOT with re-identification. Table 4 describes the evaluation results of the various Osnet model sizes under the dataset MSMT17 [38].

Table 4. Evaluation results of the different OSnet model sizes. The mAP percentages colored red represent the lowest value in that column produced by the corresponding model, whereas the green percentages represent the highest value.

Model [36,37]	Params (M)	FLOPs (G)	mAP (50%)	mAP (75%)
Osnet_x1_0	2.2	0.98	74.9%	43.8%
Osnet_x0_75	1.3	0.57	72.8%	41.4%
Osnet_x0_5	0.6	0.27	69.7%	37.5%
Osnet_x0_25	0.2	0.08	61.4%	29.5%

Integrating a tracker into our inference algorithm greatly increased the depth and value of information we could extract, instead of relying solely on an object detector to extract information such as an estimated number of objects visible in a video.

Simply using an object detector fell short in pinpointing the precise detection that we wanted to target within a frame. In short, the amount of information provided by an object detector is insufficient to pinpoint individual detections across frames. The only information gathered from an object detector is the coordinates (x1, y1, x2, y2), the confidence, and the class name. This indicates that the information provided solely by a detector is insufficient to track detections. However, the information provided by the detector is sufficient to append to a tracker such as DeepOC-Sort. Using a tracker allows us to not only identify a specific detection on the screen but also track the same detection across multiple frames. During the inference of DeepOC-Sort, we can extract the same amount of information as the object detector, but this time each detection now contains an identification number.

2.6. Tools & Equipment

The specifications for this study included a high-end computer equipped with an RTX 4090, which allowed us to train detection models quickly and run ORACLE at high speeds.

A low-noise drone was used for multiple flight plans over several days, to avoid disturbing the wildlife. The drone was equipped with a rotating gimbal camera that records up to 4 k videos and can fly for up to 30 min in ideal conditions.

Additional tools for this research included DarkMark [40], an advanced labeling tool used to develop, improve, and augment our dataset, which features automatic labeling, dataset statistics reviewing, and more.

3. AMVRADIA Dataset

The AMVRADIA dataset is named after the Amrvrakikos gulf, the largest gulf in Greece. It was named due to our original task of leveraging the protection of ecosystems using drones and surveying the environment's wildlife to obtain analytics of the ecosystem.

The dataset development commenced with the retrieval of wildlife footage from the Amvrakikos Gulf. Initially, our dataset consisted of three videos totaling 10 min of footage, as described in a prior study [13]. Subsequently, each video was processed and a selection of frames within the footage were used as images for our dataset. These images underwent manual or automatic annotation, tiling, and augmentation procedures to construct our dataset. The structure of the previous dataset is illustrated in Table 5.

Table 5. Dataset scale and improvements across the study.

Dataset Type	Image Count	Annotation Count
(2022) Initial	769	-
(2022) Augmented	10,653	34,536
(Current) Initial	1.104	27.189
(Current) Augmented	16.354	90.581

The current dataset annotation structure is described in Table 6.

Table 6. Dataset statistics of the full-scale images. Dataset statistics retrieved using DarkMark [40].

Class ID	Class Name	Count	Images	Avg Size	Max Size	Max Annotations
0	Bird	25.398	535	27 × 26	188 × 334	129
1	Flock	181	106	88 × 67	247 × 130	5
2	Small-flock	1.610	405	37 × 30	176 × 113	21
3	Empty images	58	58	3840 × 2160	3840 × 2160	1
-	**Total**	27.189	1.104	-	-	-

The AMVRADIA dataset consists of 4 k (3840 × 2160) resolution images; however, in order to utilize the full potential of our model, we made sure to tile every image, to stop the model from resizing the images down to the network size, as mentioned in the previous Figure 3. The tool used to apply the tiling to our dataset was DarkMark [40].

We divided the AMVRADIA dataset, Table 6, into two distinct datasets. The first dataset, labeled "Initial Dataset", contained only the tiles of every image from the initial dataset. The second dataset, labeled "Augmented", contained the tiles of every image, alongside added zoom/crop augmentation. Our objective in segregating the dataset depicted in Table 6 was to research and optimize it to determine the best capabilities for our model performance using both the initial and augmented datasets.

3.1. Initial Dataset

Table 7 depicts the statistics of the initial dataset where no augmentations were employed. This methodology used only the tiled images, with no additional images.

Table 7. Dataset statistics of the tiled images without any augmentation. Dataset statistics retrieved using DarkMark [40]. (Small-sized annotations are scaled to a minimum of 10 × 10).

Class ID	Class Name	Count	Images	Avg Size	Max Size	Max Annotations
0	Bird	22.360	4.422	28 × 28	189 × 294	63
1	Flock	193	150	76 × 68	195 × 115	3
2	Small-flock	1.466	885	38 × 32	153 × 119	10
3	Empty Images	6175	6175	640 × 640	640 × 640	1
-	**Total**	24.019	11.632	-	-	-

3.2. Augmented Dataset

Table 8 depicts the statistics of the augmented dataset, where zoom/crop augmentation was employed to increase the overall performance of our model.

We can observe in Table 8 that using random zoom/crop resulted in an increased count of annotations, at the expense of smaller annotation sizes. In some cases, the annotations became excessively small, rendering them nearly undetectable by the model. The negative aspect of zoom/cropping is the blurriness caused after the augmentation. This distortion results in a reduction of visible information within the images. This method will likely reduce the overall accuracy during training; however, despite this drawback, it could serve as a reasonable trade-off for enhancing the overall effectiveness of our detection models during inference.

Table 8. Dataset statistics of the tiled images using random zoom/crop augmentation. Dataset statistics retrieved using DarkMark [40]. (No scaling was applied to the annotation sizes).

Class ID	Class Name	Count	Images	Avg Size	Max Size	Max Annotations
0	Bird	85,090	7095	16×16	189×294	123
1	Flock	581	409	49×41	195×115	5
2	Small-flock	4910	2108	22×18	153×119	18
3	Empty Images	6742	6742	640×640	640×640	1
-	**Total**	90.581	16.354	-	-	-

4. Results

This section provides a detailed analysis of the results of the various experiments we conducted to assess the performance of the various models on our AMVRADIA dataset. This analysis included a comprehensive assessment of both the tracker and object detector across multiple IoU (intersection over union) [41] thresholds using annotated video data retrieved from footage using drones. Additionally, we examined the accuracy of our count estimation algorithm based on the annotated video. Finally, we identified the best-performing model for our use case, as well as others, in the tasks of object detection and MoT (multi-object tracking) of small- to medium-sized detections.

4.1. Training Phase Datasets

During training, we made sure to split each dataset (initial/augmented) into three different subsets: training, testing, and validation datasets. The split percentage of the datasets was 70–30%. We split the training dataset into 70%, while the test and validation datasets were both split into 15%. We made sure to split our datasets in an unbiased and random manner, to avoid over-fitting or uncorrelated results.

4.2. Detector & Tracker Evaluation Method

Our evaluation methodology for both the detector and tracker was achieved by annotating a video containing a large number of annotations with identifications. The target object count in the video was 100 objects to track/detect. Our goal was to evaluate both the detector and tracker using the annotated video and to extract various accuracy values.

We evaluated our tracker by counting the total correct tracks in between individual frames, while at the same time counting any possible changes that occurred in the track's ID.

Algorithmically, we began by initializing dictionary annotated track IDs as keys. During inference, we fetched the annotations corresponding to each frame of the annotated video. We then matched the specific track annotation IDs with the predicted track IDs based on multiple IoU thresholds. If two track IDs across two or more frames were the same, then these tracks were counted as correct; otherwise, if the track ID differed from the previous frame, we registered this as a change. This approach is summarized in Algorithm 2

Each predicted track ID the same as the previous one was counted as a correct track. If the track ID was not the same as the previous matched detection ID, then we counted this as a changed track.

Algorithm 2 Detection/Tracker Evaluator Optimized

```
1:  Requires:
2:      matched_dets: Dictionary,          ▷ Annotation IDs as keys, Tracker Evaluators as values.
3:      objectKey: STR,                    ▷ Current Annotation ID
4:      det_id: INT,                       ▷ Current Detection ID
5:      iou: FLOAT,                        ▷ Intersection over Union value
6:      IOU_THRESHOLDS: LIST of FLOATs,    ▷ Threshold values for evaluation
7:  Returns:
8:      matched_dets: Dictionary,          ▷ Updated with evaluation results
9:  procedure EVALUATOR(matched_dets, objectKey, det_id, iou, IOU_THRESHOLDS)
10:     indices ← [index for index, iou_thres in enumerate(IOU_THRESHOLDS) if iou $\geq$ iou_thres]
11:     for index in indices do
12:         if not matched_dets.contains(objectKey) then
13:             matched_dets[objectKey] ← new TrackerEvaluator()
14:         end if
15:         matched_dets[objectKey].track_total(index)    ▷ Increment total count for the index
16:         if matched_dets[objectKey].current_det_id == 0 then
17:             matched_dets[objectKey].current_det_id ← det_id    ▷ Set detection ID
18:             matched_dets[objectKey].track_correct(index)    ▷ Mark as correct for index
19:         else if matched_dets[objectKey].current_det_id == det_id then
20:             matched_dets[objectKey].track_correct(index)    ▷ Increment correct count
21:         else
22:             matched_dets[objectKey].track_changed(det_id, index)    ▷ Track has changed
23:         end if
24:     end for
25:     return matched_dets
26: end procedure
```

4.2.1. Detection Model Evaluation

The 2022 study [13] that was conducted utilizing the AMVRADIA 2022 dataset, as described in Table 5, was evaluated with the models YOLOv4 and YOLOv4-tiny [42]. Table 9 describes the inference results of these two models and the underlying problems that were later confirmed with the improvements of the current study's dataset.

Table 9. YOLOv4 and YOLOv4-tiny evaluation results.

YOLO Model	mAP (50)	Average IoU (50)
YOLOv4	91.28%	65.10%
YOLOv4-tiny	85.64%	47.90%

On a similar scale to Table 2, we trained several models on the datasets labeled as 'Initial' and 'Augmented', and evaluated each model with its equivalent or opposite dataset. Each model underwent evaluations with multiple mAP results based on an IoU threshold vector. The IoU threshold vector spanned from 0.5 to 0.95 IoU, with intervals of 0.05. Our assessment for each model was carefully examined to produce results with the highest possible mAP and overall model performance we could achieve.

Table 10 illustrates the training and evaluation results under the split dataset for AMVRADIA. The 'Model Name' column refers to the models we used for training/evaluation. 'Representing Dataset' refers to the dataset used to train the corresponding model. Finally, the columns 'Evaluated under Initial' and 'Evaluated under Augmented' refer to which of the two datasets, described in Sections 3.1 and 3.2, the model was evaluated on.

The results provided in Table 10 present a significant increase between each model. However, in the case of YOLOv7x, there was an evident reduction in the overall mAP compared to using YOLOv7. This was likely due to a complexity issue caused by using a larger-sized model or due to its inability to detect small-sized detections. For YOLOv8, this was not the case, as there was an increase in overall mAP from YOLOv8 to YOLOv8x. This shows that certain models may behave differently with unique or challenging datasets.

Table 10. The evaluation results of multiple models trained and evaluated under the split AMVRADIA dataset. (The mAP percentages colored red correlate to the lowest value in that column produced based on the corresponding model, whereas the green percentages represent the highest value).

		Evaluated under Initial			**Evaluated under Augmented**		
Model Name	**Representing Dataset**	**mAP (50%)**	**mAP (75%)**	**mAP (50–95)**	**mAP (50%)**	**mAP (75%)**	**mAP (50–95)**
YOLOv7 [18]	Initial	72.38%	45.17%	41.67%	43.51%	23.73%	23.60%
YOLOv7	Augmented	92.37%	78.32%	64.86%	82.14%	53.05%	50.31%
YOLOv7x [18]	Initial	75.16%	47.56%	44.44%	44.38%	24.10%	24.29%
YOLOv7x	Augmented	90.40%	77.41%	63.60%	82.04%	53.04%	49.80%
YOLOv8 (m) [19]	Initial	84.69%	65.57%	57.82%	58.38%	36.45%	35.32%
YOLOv8 (m)	Augmented	92.80%	82.35%	70.18%	82.07%	56.69%	52.66%
YOLOv8x [19]	Initial	85.72%	66.92%	58.25%	61.49%	37.36%	36.75%
YOLOv8x	Augmented	94.13%	84.36%	70.15%	79.73%	53.19%	50.54%
YOLOv8x-p2 [19]	Initial	89.84%	77.83%	63.82%	63.98%	43.13%	40.22%
YOLOv8x-p2	Augmented	95.69%	89.88%	75.88%	87.56%	64.92%	58.24%

4.2.2. Tracking Model Evaluation

This subsection includes the evaluation results extracted from our object detection and tracker evaluator Algorithm 2 using the best-performing models based on mAP from our previous evaluation results, Table 10. For our tracker and object detection evaluator, we used the best performing tracking model described in Table 4.

The results described in Table 11 highlight the significant impact of utilizing models that prioritized the detection of small objects. Notably, YOLOv8x-p2 with the augmented dataset emerged as the top-performing model, demonstrating superior performance to the rest of the evaluated models. Interestingly, however, YOLOv8x was slightly behind YOLOv8x-p2, nearly matching the performance of the model using the p2 module.

Table 11. Object detection and tracker evaluation results for each best-performing detection model. The NMS IoU was set to 0.5 IoU. The green text represents the best-performing model. The red text represents the worst-performing model. The lifespan we used for each track to be estimated as a count was 2 s.

Detection. Model	**mAP (50%)**	**mAP (75%)**	**mAP (50–95)**	**Estimated Count**
yolov7 Initial	64.97% 68.11% 0.55 IoU	34.47%	37.67%	73
yolov7 Augmented	74.25% (74.33% 0.55 IoU)	42.33%	42.08%	79
yolov7x Initial	69.42% (70.35% 0.55 IoU)	37.68%	38.38%	73
yolov7x Augmented	71.27% (72.54% 0.55 IoU)	42.82%	42.70%	78
yolov8 Initial	71.96%	35.37%	40.30%	64
yolov8 Augmented	84.33%	50.04%	52.07%	63
yolov8x Initial	71.75%	40.85%	42.10%	63
yolov8x Augmented	91.48%	58.82%	55.03%	60
yolov8x-p2 Initial	71.02%	43.18%	43.99%	64
yolov8x-p2 Augmented	91.89%	58.82%	56.24%	65
-	-	-	**Target Estimated Count:**	100

While YOLOv7 augmented appears to display a slight increase in the overall mAP compared to YOLOv7x Augmented in Table 10, there was a small increase in the overall mAP when evaluating the precision of the detector and tracker, Table 11. If our goal was to take advantage of the best possible results for our estimation count algorithm, using the yolov8x-p2 detection model with the augmented dataset was the best approach.

4.3. Analytics & Visualization

In our five-layered model ORACLE, we implemented various analytics extraction and visualization algorithms that used the tracks retrieved from the post-detection or tracking layers. Nearly all of the algorithms we implemented utilized the ID from the tracks.

1. **Object count estimation**: Our object count estimation algorithm leverages the IDs from the tracks of each detection. We estimate the total count in a video by tracking the lifespan of all tracked detections with the same ID. This method involves initializing a total lifespan to count a track and display the detection with the color green. Our objective is to count the IDs that remain active on the screen for over 2 s (equivalent to 60 frames in a 30 fps video).Based on the best-performing model provided in Table 11, we evaluated our object count estimation algorithm over several seconds (lifespans) to count a track. Figure 15 depicts the decrease in the total estimated counts detected in the annotated video. The dotted horizontal line represents the target count of the annotated video, whilst the polylines represent the total estimated count for different lifespans.

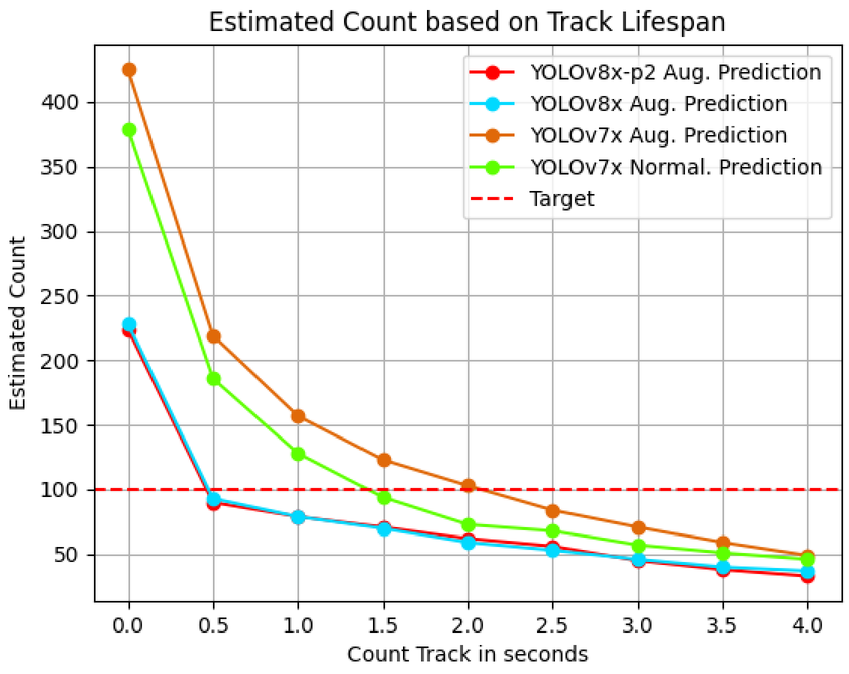


Figure 15. Estimation count of the objects in the annotated video per life span of the tracks, using three different models YOLOv7x (Normal), YOLOv7x (Augmented), YOLOv8x (Augmented), and YOLOv8x-p2 (Augmented). (The results differed from the previously retrieved results due to improvements in the code, Table 11).

In Figure 15, we can observe the estimated count results of three individual models all trained under the augmented dataset. Increasing the seconds to count a track from 0 or higher subsequently decreased the total `Estimated Count`. To achieve equilibrium and determine the optimal point for estimating the total count of wildlife, we needed to utilize a model capable of high-accuracy inference. The most optimal scenario would involve ORACLE predicting a total `Estimated Count` of 100 with a total lifespan of 0.0 s to count a track; however, this means that both the object detection and tracking models would need to be 100% accurate, which is realistically not possible.

False positive detections, or even miss-tracks, pose a common challenge in estimating the total count of wildlife, which is essentially considered noise. The predicted `Estimated Count` closest to the target was similar for the two models: YOLOv8x (93 estimated count) and YOLOv8x-p2 (90 estimated count) for 0.5 s. Furthermore, YOLOv7x trained on normal and augmented datasets produced an overestimated count compared to YOLOv8x. Meanwhile, YOLOv7x Augmented estimated the highest count compared to the rest of the models, proving the existence of a large count of false positive detections or miss-tracks. Therefore, YOLOv7x was an inferior candidate compared to the YOLOv8x models.

2. **Tracked Object Visualization**: This method was primarily implemented to visualize the tracked object in a small static box containing the detection in the image, with an added zoom. We also implemented it in cases where an ID was lost and found in between frames, and the box displayed a black background with its ID. If the lifespan of a non-visible detection on the screen expires, it will no longer be present and will be replaced by a new detection with the next smallest ID from the last one. An example of this algorithm in action is displayed in Figure 16, showing a visualization of tracks with static boxes and added zoom sorted by the ID. In addition, the image displays information such as the Estimated Wildlife Count in the top left corner.

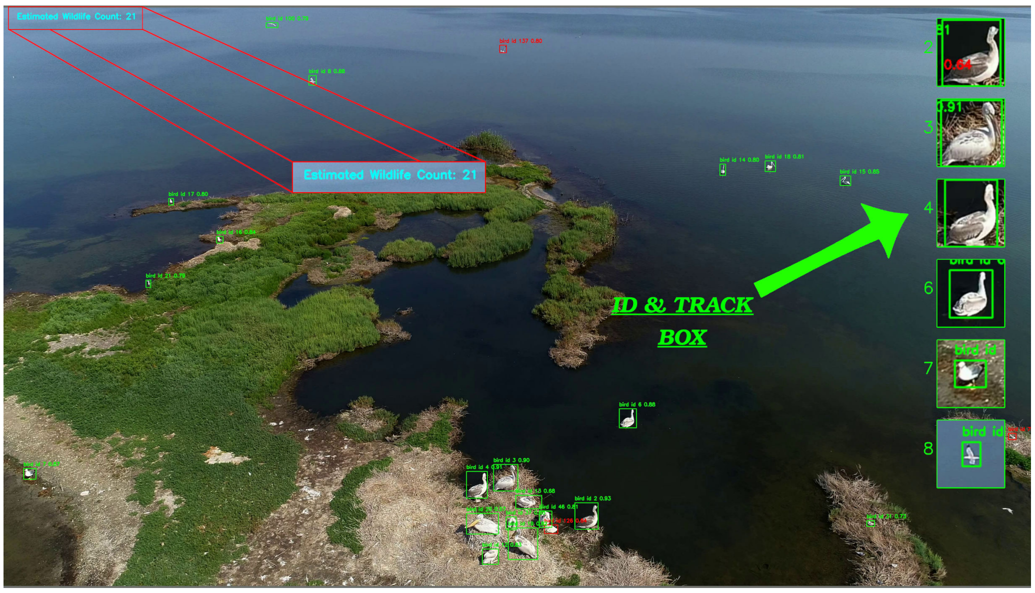


Figure 16. A visualization example of our five-layered model displaying advanced object detection, tracking, object count estimation, and track cisualization.

5. Conclusions

Our objective was to automate the task of surveying remote and inaccessible environments, without relying on manual labor. We initiated the process by securely retrieving footage suitable for training or evaluating detection models. However, merely employing object detection techniques falls short in comprehensively "surveying" a remote area and extracting valuable analytics. Thus, during the course of this study, we devised a sophisticated model titled ORACLE (optimized rigorous advanced cutting-edge model for leveraging protection of ecosystems). ORACLE not only performs various AI tasks related to computer vision, but also automates the task of surveillance and facilitates the extraction of valuable analytics.

This study delved deeper into the advancement of wild bird surveillance with the assistance of drones. Our previous study [13] focused on the development of a dataset capable of detecting small-sized objects using YOLOv4 [42] and YOLOv4-tiny. During inference, we achieved a total of 91.28% mAP with our previous dataset. However, this study advanced even further in the task of object detection, by significantly improving both the quality and quantity of the dataset.

In our new dataset, named after our project name, the AMVRADIA dataset, we achieved a peak accuracy of 95.96% (evaluated under 50% IoU), as depicted in Table 10, using a large-scale model that prioritized small- to medium-sized detections, YOLOv8x-p2 [19]. Moreover, we improved this study's quality using tracking techniques, with the assistance of the Deep OC-SORT [39] algorithm under the OS Net model [36].

Since tracking is a process that goes hand-in-hand with object detection, we evaluated both detection and tracking models on the same scale, to show a robust correlation of our evaluation results, as depicted in Table 11. This evaluation methodology proved the technicality and impact of the performance of our tracks using several detection models. It also proved how the detection layer is the basis of our surveillance model ORACLE and its overall performance.

In addition to the detection and tracking layers, which facilitate robust data extraction, we implemented and presented various algorithms for information extraction and visualization. Among these, the primary algorithm was our object count estimation algorithm (referenced as Item 1. While the implementation of this algorithm may appear straightforward, its accuracy is heavily dependent on the performance of both the detection and tracking layers.

Finally, this study concluded with the use of high-level state-of-the-art detection and tracking models that facilitate the task of data extraction from drone footage. We optimized and fine-tuned both of these techniques to a state of near-perfect accuracy. We not only evaluated the models but also developed algorithms capable of extracting valuable information, such as a wildlife estimation counts.

6. Future Work

This work introduced the ORACLE model and highlighted its capabilities. However, there is considerable room for improvement and additional features that could be implemented. As of the time of writing, we have already begun our efforts to expand our model and focus on the task of data collection for wildlife surveillance using drones. One task we have set our sights on is to leverage state-of-the-art language models and generate captions for each individual detection additionally.

The impact of avian influenza on Dalmatian pelicans in 2022 was a severe disaster [7]. Our goal is to observe the wildlife and the environment of the Amrakikos Gulf in Greece and conserve it in the long run with the assistance of our model ORACLE. Additionally, our study aimed to estimate colony sizes using ORACLE's data collected across multiple seasons and to apply machine learning techniques [43] to observe irregularities or states in colony sizes based on environmental impacts.

As of the time of writing, we have upgraded our hardware and resources. We obtained a high-end industrial-grade DJI drone capable of capturing thermal imagery. Due to this, we have set our sights on detecting information such as the observable temperature of each detection. This will allow us to classify if a bird, such as Dalmatian pelicans, could be carrying diseases like influenza [44]. Finally, the AMVRADIA dataset is single-domain, meaning it is limited to a single environment. However, the Amvrakikos Gulf consists of several islets containing wildlife. Our future plan is to expand the dataset into a multi-domain dataset, thereby increasing both its variety and quantity. As of the time of writing, surveillance is prohibited due to the nesting and breeding season of the wildlife of the Amvrakikos Gulf. The breeding season lasts from the start of January till the end of June [5]. Therefore, we are unable to retrieve more data during this period.

Author Contributions: Conceptualization, P.K.; methodology, P.K., D.M. and C.S.; software, D.M. and P.K.; validation, and P.K.; data collection, D.M.; writing—original draft preparation, D.M., C.S. and P.K.; writing—review and editing, D.M. and P.K.; supervision, P.K. and C.S.; project administration, C.S. and P.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received funding from the project "Research and development advanced computational techniques for remote monitoring of wild poultry to prevent the spread of avian influenza using drones and other devices" submitted in Priority Axis 3 "Research and Implementation", of the financing program "Physical Environment & Innovative actions 2022" of the Green Fund.

Data Availability Statement: Data not available for this study.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mazumdar, S. Drone Applications in Wildlife Research—A Synoptic Review. In *Environmental Informatics: Challenges and Solutions*; Paul, P.K., Choudhury, A., Biswas, A., Singh, B.K., Eds.; Springer Nature Singapore: Singapore, 2022; pp. 237–257. [CrossRef]
2. Han, Y.G.; Yoo, S.; Kwon, O. Possibility of applying unmanned aerial vehicle (UAV) and mapping software for the monitoring of waterbirds and their habitats. *J. Ecol. Environ.* **2017**, *41*, 21. [CrossRef]
3. Naeher, S.; Geraga, M.; Papatheodorou, G.; Ferentinos, G.; Kaberi, H.; Schubert, C.J. Environmental variations in a semi-enclosed embayment (Amvrakikos Gulf, Greece)–reconstructions based on benthic foraminifera abundance and lipid biomarker pattern. *Biogeosciences* **2012**, *9*, 5081–5094. [CrossRef]
4. Das, N.; Padhy, D.N.; Dey, N.; Mukherjee, A.; Maiti, A. Building of an edge enabled drone network ecosystem for bird species identification. *Ecol. Inform.* **2021**, *68*, 101540. [CrossRef]
5. Crivelli, A.J.; Hatzilacou, D.; Catsadorakis, G. The breeding biology of the Dalmatian Pelican Pelecanus crispus. *Ibis* **1998**, *140*, 472–481. [CrossRef]
6. Alain, C.; Catsadorakis, G.; Hatzilacou, D.; Naziridis, T. Pelecanus crispus Dalmatian Pelican. *Soc. Prot. Prespas Conserv. Res. Dep. BWP Update* **1997**, *1*, 149–153.
7. Alexandrou, O.; Malakou, M.; Catsadorakis, G. The impact of avian influenza 2022 on Dalmatian pelicans was the worst ever wildlife disaster in Greece. *Oryx* **2022**, *56*, 813. [CrossRef]
8. Coluccia, A.; Fascista, A.; Sommer, L.; Schumann, A.; Dimou, A.; Zarpalas, D. The Drone-vs-Bird Detection Grand Challenge at ICASSP 2023: A Review of Methods and Results. *IEEE Open J. Signal Process.* **2024**, 1–15. [CrossRef]
9. Hong, S.J.; Han, Y.; Kim, S.Y.; Lee, A.; Kim, G. Application of Deep-Learning Methods to Bird Detection Using Unmanned Aerial Vehicle Imagery. *Sensors* **2019**, *19*, 1651. [CrossRef]
10. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
11. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Proceedings of the Computer Vision–ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 21–37.
12. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef]
13. Mpouziotas, D.; Karvelis, P.; Tsoulos, I.; Stylios, C. Automated Wildlife Bird Detection from Drone Footage Using Computer Vision Techniques. *Appl. Sci.* **2023**, *13*, 7787. [CrossRef]
14. T'Jampens, R.; Hernandez, F.; Vandecasteele, F.; Verstockt, S. Automatic detection, tracking and counting of birds in marine video content. In Proceedings of the 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), Oulu, Finland, 12–15 December 2016; pp. 1–6. [CrossRef]
15. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.
16. Betancourt, I.; McLinn, C. Teaching with the Macaulay Library: An Online Archive of Animal Behavior Recordings. *J. Microbiol. Biol. Educ.* **2012**, *13*, 86–88. [CrossRef] [PubMed]
17. Wah, C.; Branson, S.; Welinder, P.; Perona, P.; Belongie, S. The Caltech-UCSD Birds200-2011 Dataset. In *Advances in Water Resources-ADV WATER RESOUR*; California Institute of Technology: Pasadena, CA, USA, 2011.
18. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors. In Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 18–22 June 2023; pp. 7464–7475. [CrossRef]
19. Jocher, G.; Chaurasia, A.; Stoken, A.; Borovec, J.; Nano; Kwon, Y.; Michael, K.; Xie, T.; Fang, J.; imyhxy; et al. ultralytics/yolov5: v7.0-YOLOv5 SOTA Realtime Instance Segmentation. 22nd of November. Available online: https://zenodo.org/records/7347926 (accessed on 2 February 2024).
20. Li, J.; Liang, X.; Wei, Y.; Xu, T.; Feng, J.; Yan, S. Perceptual Generative Adversarial Networks for Small Object Detection. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 16 June 2017; pp. 1951–1959. [CrossRef]
21. Hassan, S.; Mujtaba, G.; Rajput, A.; Fatima, N. Multi-object tracking: A systematic literature review. *Multimed. Tools Appl.* **2023**, *83*, 43439–43492. [CrossRef]
22. Milan, A.; Schindler, K.; Roth, S. Challenges of Ground Truth Evaluation of Multi-target Tracking. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition Workshops, Portland, OR, USA, 23–28 June 2013; pp. 735–742. [CrossRef]
23. Lin, T.Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R.; Hays, J.; Perona, P.; Ramanan, D.; Zitnick, C.L.; Dollár, P. Microsoft COCO: Common Objects in Context. *arXiv* **2014**, arXiv:1405.0312.
24. Kisantal, M.; Wojna, Z.; Murawski, J.; Naruniec, J.; Cho, K. Augmentation for small object detection. In Proceedings of the Conference: 9th International Conference on Advances in Computing and Information Technology, Sydney, Australia, 21–22 December 2019.

25. Bożko, A.; Ambroziak, L. Influence of Insufficient Dataset Augmentation on IoU and Detection Threshold in CNN Training for Object Detection on Aerial Images. *Sensors* **2022**, *22*, 9080. [CrossRef] [PubMed]
26. Gilg, J.; Teepe, T.; Herzog, F.; Wolters, P.; Rigoll, G. Do We Still Need Non-Maximum Suppression? Accurate Confidence Estimates and Implicit Duplication Modeling with IoU-Aware Calibration. *arXiv* **2023**, arXiv:2309.03110.
27. Charette, S. Darknet/YOLO. April 2022. Available online: https://github.com/hank-ai/darknet (accessed on 2 February 2024).
28. Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; Lerer, A. Automatic Differentiation in PyTorch. In Proceedings of the NIPS 2017 Workshop on Autodiff, Long Beach, CA, USA, 28 October 2017.
29. Mpouziotas, D.; Mastrapas, E.; Dimokas, N.; Karvelis, P.; Glavas, E. Object Detection for Low Light Images. In Proceedings of the 2022 7th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM), Ioannina, Greece, 23–25 September 2022; pp. 1–6. [CrossRef]
30. Kirk, D. NVIDIA CUDA software and GPU parallel computing architecture. In Proceedings of the 6th International Symposium on Memory Management, Montreal, QC, Canada, 21–22 October 2007; Volume 7, pp. 103–104. [CrossRef]
31. Yi, R.; Cao, T.; Zhou, A.; Ma, X.; Wang, S.; Xu, M. Boosting DNN Cold Inference on Edge Devices. In Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services, Helsinki, Finland, 18–22 June 2022. [CrossRef]
32. Akyon, F.C.; Onur Altinuc, S.; Temizel, A. Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection. In Proceedings of the 2022 IEEE International Conference on Image Processing (ICIP), Bordeaux, France, 16–19 October 2022; pp. 966–970. [CrossRef]
33. Charette, S. DarkHelp, C++ Wrapper Library for Darknet 24 June 2022. Available online: https://github.com/stephanecharette/DarkHelp (accessed on 2 February 2024).
34. Chen, R.; Guo, Y.; Zheng, H.; Jiang, H. A Comprehensive Approach for UAV Small Object Detection with Simulation-based Transfer Learning and Adaptive Fusion. *arXiv* **2021**, arXiv:2109.01800.
35. Cossio, M. Augmenting Medical Imaging: A Comprehensive Catalogue of 65 Techniques for Enhanced Data Analysis. *arXiv* **2023**, arXiv:2303.01178.
36. Zhou, K.; Yang, Y.; Cavallaro, A.; Xiang, T. Omni-Scale Feature Learning for Person Re-Identification. *arXiv* **2019**, arXiv:1905.00953.
37. Zhou, K.; Yang, Y.; Cavallaro, A.; Xiang, T. Learning Generalisable Omni-Scale Representations for Person Re-Identification. *arXiv* **2019**, arXiv:1910.06827.
38. Wei, L.; Zhang, S.; Gao, W.; Tian, Q. Person Transfer GAN to Bridge Domain Gap for Person Re-identification. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Los Alamitos, CA, USA, 16–20 June 2017; pp. 79–88. [CrossRef]
39. Maggiolino, G.; Ahmad, A.; Cao, J.; Kitani, K. Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification. *arXiv* **2023**, arXiv:2302.11813.
40. Charette, S. DarkMark C++ GUI Tool for Darknet-Code Run. 2019–2023. Available online: https://www.ccoderun.ca/darkmark/ (Accessed on 2 February 2024).
41. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection over Union. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019.
42. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
43. Paolanti, M.; Romeo, L.; Felicetti, A.; Mancini, A.; Frontoni, E.; Loncarski, J. Machine Learning approach for Predictive Maintenance in Industry 4.0. In Proceedings of the 2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Oulu, Finland, 2–4 July 2018; pp. 1–6.
44. Beyit, A.; Meki, I.; Barry, Y.; Haki, M.; El Ghoutoub, A.; Hamma, S.; Abdelwahab, N.; Doumbia, B.; Benane, H.; Daf, D.; et al. Avian influenza H5N1 in a great white pelican (*Pelecanus onocrotalus*), Mauritania 2022. *Vet. Res. Commun.* **2023**, *47*, 2193–2197. [CrossRef] [PubMed]