NEMO TOOLS

Next-generation monitoring & mapping tools to assess marine ecosystems & biodiversity

Deliverable D3.2

Report on UAV imagery detection of marine megafauna

Greece 2.0
NATIONAL RECOVERY AND RESILIENCE PLAN

Funded by the European Union
NextGenerationEU

This project is carried out within the framework of the National Recovery and Resilience Plan Greece 2.0, funded by the European Union – NextGenerationEU (Implementation body: HFRI).

Views and opinions expressed are however those of the beneficiaries only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

DOCUMENT INFORMATION AND VERSION CONTROL

Project Acronym	NEMO-Tools
Project Title	Next-generation monitoring and mapping tools to assess marine ecosystems and biodiversity
Project Number	016035
Work Package	WP2
Related Task(s)	T3.2
Deliverable Number	D3.2
Deliverable Name	Report on UAV imagery detection of marine megafauna
Due Date	14 December 2025
Date Delivered	14 December 2025
Dissemination Level	Public — fully open (automatically posted online on the Project Results platforms)

VERSION CONTROL

Revision-N°	Date	Description	Prepared By	Reviewed By
	20/11/2025	1st Draft	A. Mazaris	S. Katsanevakis
	12/12/2025	Final Draft	A. Mazaris	

Executive Summary

This Deliverable 3.2 – "Report on UAV imagery detection of marine megafauna" – presents an automated detection system for monitoring sea turtle populations by combining low-cost commercial UAV platforms with deep-learning object detection algorithms. The system employed a lightweight YOLOv8-nano architecture that processed high-resolution RGB imagery through window-sliding tiling methodology, achieving robust detection performance with F1-scores of 0.93, precision of 0.97, and mAP@0.5 of 0.973. The model was trained on 60 manually annotated images collected from Koutavos lagoon, Kefalonia, using consumer-grade GPU hardware, demonstrating the technology's feasibility even in resource-limited conservation settings. The methodology addressed inherent class imbalance through targeted data augmentation while preventing spatial data leakage, enabling reliable identification of loggerhead sea turtles (Caretta caretta) at or near the water surface in heterogeneous coastal environments.

The demonstrated advantages include dramatically reduced operational costs compared to traditional manned aerial or boat-based surveys, substantially expanded spatial and temporal coverage, and rapid processing turnaround, typically hours rather than weeks for manual analysis. Detection outputs provided spatially explicit data that can directly support identification of critical habitat areas, coastal development impact assessments, marine protected area optimization, and detection of climate-driven habitat use shifts. The system integrates seamlessly with standard conservation monitoring frameworks and geographic information systems, enabling evidence-based spatial planning and adaptive management decisions.

As an operationally proven, scalable technology requiring minimal technical barriers to implementation, this automated UAV-based detection approach demonstrates the feasibility of transitioning from opportunistic observations to systematic, data-driven sea turtle monitoring programs essential for effective conservation and management.

TABLE OF CONTENTS

DOCUMENT INFORMATION AND VERSION CONTROL 3
VERSION CONTROL 3
Executive Summary 4
TABLE OF CONTENTS 5
CONTRIBUTORS 6
1. Introduction 7
2. Materials and Methods 10
2.1 Data acquisition 10
2.2 Data Annotation and labelling 11
2.3 Image Tiling and data augmentation 12
2.4 Model training 13
3. Results 13
4. Discussion 14
5. Technical Support 16
References 17
Appendices 24
Appendix I 24
Appendix II 28

CONTRIBUTORS

TABLE 1 NAMES AND ROLES OF CONTRIBUTORS TO THIS DELIVERABLE.

Name	Affiliation	WP Lead	Task Lead
Antonios Mazaris	Aristotle University of Thessaloniki	AUTH	
Dhouha Ouerfelli	Aristotle University of Thessaloniki		
Christos Adam	Aristotle University of Thessaloniki		
Olympos Andreadis	Aristotle University of Thessaloniki		
Jennifer Pistevos	Aristotle University of Thessaloniki		
Valentini Stamatiadou	Aristotle University of Thessaloniki		
Aikaterini Konsta	Aristotle University of Thessaloniki		
Dimitra Dalla	Aristotle University of Thessaloniki		
Savvas Genitsaris	National and Kapodistrian University of Athens		
Stelios Katsanevakis	University of Aegean		

1. Introduction

Marine megafauna, including cetaceans, elasmobranchs, and sea turtles, play a critical ecological role in marine ecosystems, contributing to trophic regulation, nutrient cycling, and ecosystem resilience (Heithaus et al., 2008). Many of these species are long-lived, slow to mature, and highly sensitive to anthropogenic pressures such as habitat degradation, fisheries interactions, vessel traffic, coastal development, and climate change (Lewison et al., 2014). As a result, a substantial proportion of marine megafauna species are currently classified as threatened or endangered at regional and global scales (Davidson et al., 2012; IUCN, 2023). Effective conservation and management of these species rely fundamentally on robust, spatially explicit, and temporally consistent information on their distribution, abundance, habitat use, and behavior (Mazaris et al., 2017; Hazen et al., 2013).

Sea turtles, in particular, represent a flagship group within marine megafauna conservation. All seven extant species are listed under the IUCN Red List, with varying degrees of extinction risk (Wallace et al., 2011). Sea turtles exhibit complex life cycles that span vast oceanic areas and multiple jurisdictions, encompassing nesting beaches, neritic foraging grounds, migratory corridors, and pelagic habitats (Bolten, 2003). This spatial and ecological complexity poses significant challenges for monitoring and management, especially in coastal and offshore environments where human activities are concentrated. Traditional monitoring approaches, while invaluable, often face limitations related to spatial coverage, observer bias, cost, and logistical constraints (Marsh & Sinclair, 1989). Counting 50–100 million years of presence on the planet (Poloczanska et al., 2009), sea turtles represent an ideal model as they have a complex life cycle with distinct foraging and breeding grounds that may be distant (up to 2800 km for hard-shelled sea turtles; Hays & Scott, 2013). While the breeding grounds of all sea turtle species globally are generally well documented, occurring primarily in tropical and subtropical areas (Wallace et al., 2010), we are still in the process of identifying foraging grounds, which extend to the polar seas for some species, with debate remaining about which grounds should be considered "key" (Rees et al., 2016). Furthermore, foraging habitats contain mixed cohorts (McClellan & Read, 2007) and mature animals that breed at a given rookery could use distinct and very distant foraging sites (Schofield et al., 2013). The identification of important areas for such highly migratory marine species is a primary and essential step to enhance our conservation capacity (Hays et al., 2019). For example, towards this direction, efforts have been made to determine important areas for marine mammals, as a means to delineate discrete portions of habitat significant to these species in order to prioritize conservation measures (Corrigan et al., 2014). The Mediterranean loggerheads are subjected to multiple human-related threats at sea with the need for their protection being highlighted (Mazaris et al., 2019; Rees et al., 2013).

In recent years, the rapid development of Unmanned Aerial Vehicles (UAVs) has created new opportunities for marine wildlife monitoring (Hodgson et al., 2013; Kiszka et al., 2016; Schofield et al., 2017). UAV-based imagery has emerged as a powerful

tool for detecting, identifying, and quantifying marine megafauna across a range of spatial and temporal scales (Christiansen et al., 2016; Schofield et al., 2017a). Conventional methods for monitoring sea turtles and other marine megafauna include boat-based visual surveys, aerial surveys using manned aircraft, satellite telemetry, acoustic monitoring, and in-water capture or observation techniques. While these methods have generated critical ecological knowledge, each presents inherent limitations. Boat-based surveys are often restricted by weather conditions, sea state, and limited spatial coverage, while observer fatigue and perception bias can affect detection rates (Buckland et al., 2001). Manned aerial surveys allow for broader spatial coverage but are associated with high operational costs, safety concerns, and limited temporal flexibility (Pollock et al., 2006). Satellite telemetry provides detailed movement data for individual animals but is invasive, costly, and typically limited to small sample sizes (Godley et al., 2008; Jeffers and Godley, 2016). In-water methods, such as snorkelling or diving surveys, are constrained by depth, visibility, and diver safety, and may disturb animals or habitats (Seminoff et al., 2014). These constraints are particularly pronounced in shallow coastal waters, where sea turtles frequently forage and interact with human activities. There is therefore a growing need for complementary, non-invasive monitoring tools that can provide high-resolution spatial data while minimizing disturbance and cost.

UAVs equipped with high-resolution optical sensors offer a flexible, cost-effective, and minimally invasive alternative for marine megafauna monitoring (Linchant et al., 2015; Schofield et al., 2017a; Rees et al., 2018). Advances in battery life, sensor quality, positioning systems, and flight stability have significantly enhanced the capacity of UAVs to collect detailed imagery over marine environments (Anderson & Gaston, 2013). For sea turtles, UAV imagery enables direct visual detection at the sea surface and in shallow waters, where individuals can often be distinguished based on body shape, size, coloration, and behavior (Schofield et al., 2017). Unlike traditional aerial surveys, UAVs can operate at lower altitudes and slower speeds, improving detection probability and allowing for repeated surveys over the same area (Chabot & Bird, 2015). This makes UAVs particularly well suited for fine-scale habitat assessments, population counts in foraging areas, and monitoring of nearshore habitats adjacent to nesting beaches.

Beyond simple presence–absence data, UAV imagery can support the estimation of relative abundance, density, and spatial distribution patterns (Hodgson et al., 2018). In some cases, it can also provide information on body condition, behavior (e.g. resting, swimming, feeding), and interactions with anthropogenic structures or activities (Bevan et al., 2018). When combined with environmental data such as water depth, substrate type, or proximity to human pressures, UAV-derived observations can significantly enhance habitat-use analyses (Schofield et al., 2019b; Dickson et al., 2022a). One of the key advantages of UAV-based monitoring is its non-invasive nature. UAVs can be operated at altitudes that minimize disturbance while still providing sufficient image resolution for species identification (Vas et al., 2015; Schofield et al., 2017). This is particularly important for sea turtles, which may alter their

behavior in response to close vessel approaches or in-water observers (Hazel et al., 2007). UAVs also offer high spatial and temporal resolution. This is especially relevant for sea turtles, whose presence in coastal areas may vary with temperature, food availability, or reproductive cycles (Hays et al., 2002). From a logistical perspective, UAV surveys are relatively cost-efficient compared to manned aerial surveys and can be deployed rapidly in response to management needs, such as assessing turtle presence prior to coastal development activities or evaluating the effectiveness of spatial protection measures (Colefax et al., 2017; Johnston et al., 2017; Schofield et al., 2017). UAVs are also well suited for monitoring areas that are difficult to access by boat or on foot, including shallow reefs, seagrass meadows, and remote coastal zones.

A major recent advancement in UAV-based monitoring is the integration of automated image analysis techniques, including machine learning and artificial intelligence (AI) (Gray et al., 2019a). Manual analysis of UAV imagery can be time-consuming and subject to observer bias, particularly when large datasets are involved (Hodgson et al., 2016). Automated detection algorithms offer the potential to increase efficiency, consistency, and scalability (Kellenberger et al., 2018). For sea turtles, automated approaches can be trained to recognize species-specific visual features, even under variable environmental conditions (Gray et al., 2019b). The integration of UAV imagery with automated analysis pipelines represents a critical step toward operational monitoring systems that can support routine conservation and management applications (Weinstein, 2018).

The application of UAV imagery for detecting sea turtles and other marine megafauna has significant implications for conservation planning, impact assessment, and adaptive management. High-resolution spatial data can inform the designation and evaluation of marine protected areas, identify critical habitats and corridors, and support the assessment of cumulative impacts from human activities (Maxwell et al., 2015). In the context of climate change, UAV-based monitoring can contribute to detecting shifts in distribution and habitat use, providing early warning signals of ecological change. For sea turtles, this is particularly relevant as warming seas, changing productivity patterns, and coastal development alter the availability and suitability of traditional foraging and nesting areas (Hawkes et al., 2009). By providing timely, reliable, and spatially explicit information, UAV imagery can enhance evidence-based decision-making and support the implementation of effective conservation measures. As regulatory frameworks and ethical guidelines for UAV use continue to evolve (Hodgson & Koh, 2016), UAV-based monitoring is poised to become an integral component of marine megafauna research and management (Dickson et al., 2022b).

This report presents an automated species detection system based on advanced deep-learning methods, developed and trained using an imagery dataset collected by project team members in the Ionian Sea. Specifically, we apply a YOLO-based detection pipeline and assess the extent to which a YOLOv8 model trained on a limited ("tiny") dataset can reliably detect sea turtles in heterogeneous nearshore coastal waters using low-cost UAV imagery.

2. Materials and Methods

2.1 Data acquisition

This study was conducted in the marine area of Koutavos lagoon, Argostoli, Kefalonia in Greece situated within the geographical coordinates 38.173° N and 20.490° E, representing a very well known habitat for sea turtle (Caretta caretta) threatened by intensive human related activities (Papazekou et al., 2024). UAV surveys were performed in 2024 using a DJI unmanned aerial vehicle equipped with DJI FC3411 camera over the water lagoon aiming to cover the area for detecting turtles at or near the surface waters. All the obtained images were high resolution RGB imagery.

Reference map could be found here (file:///C:/turtle_detection/turtle_2025/geospatial_analysis/drone_flight_map.html).

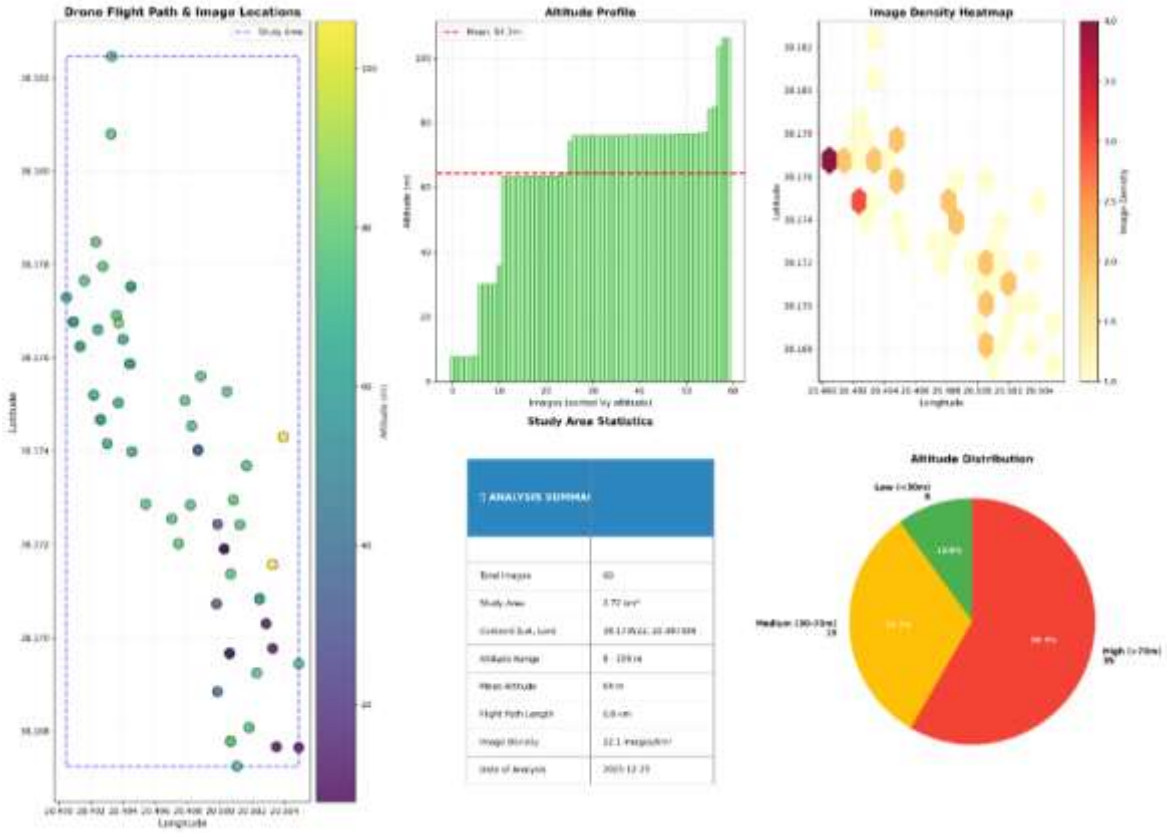
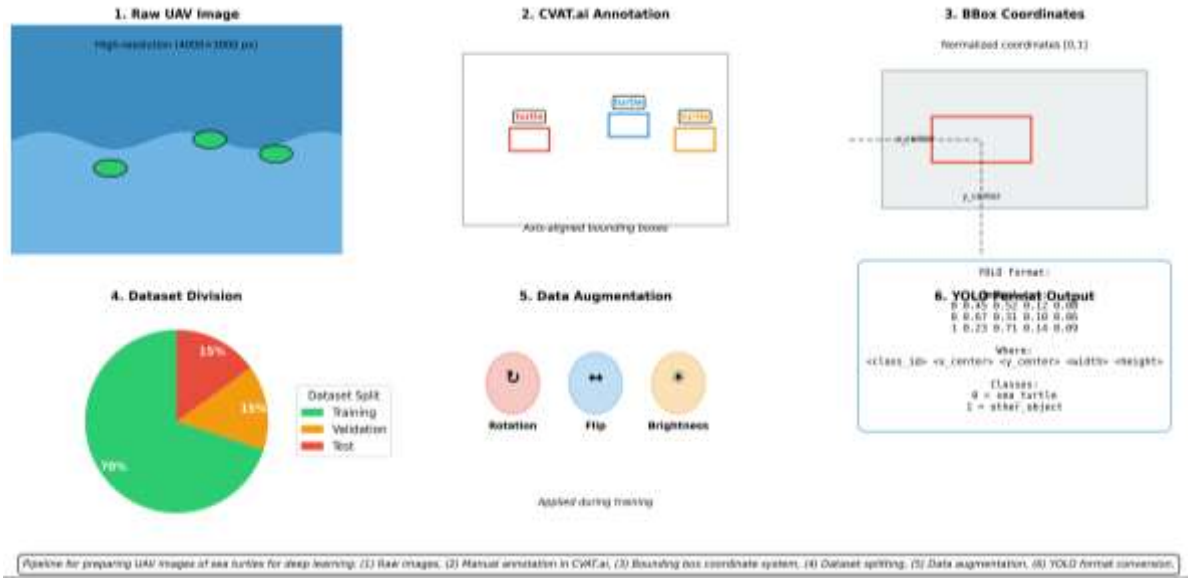


Figure 1. Location and spatial analysis of the captured images via the drone

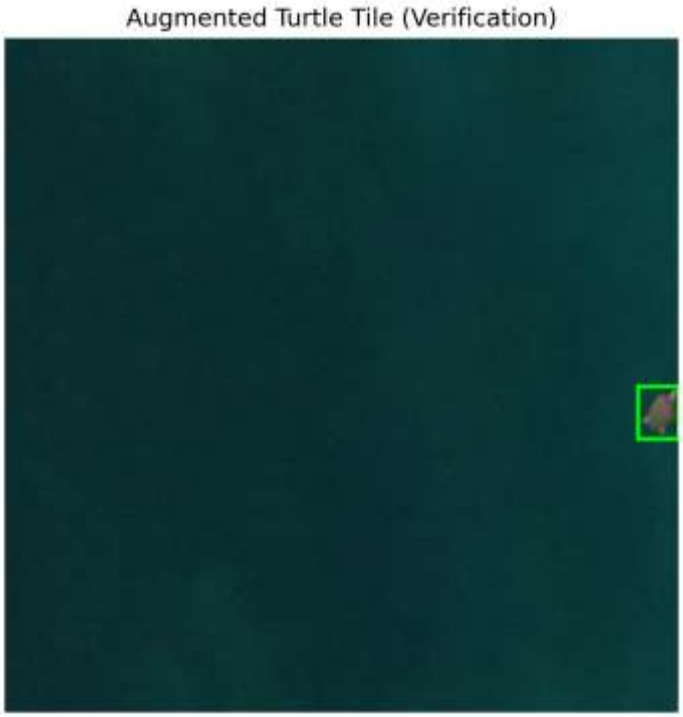
2.2 Data Annotation and labelling



A total of 60 images acquired by the drone were manually annotated using CVAT.ai (Computer Vision Annotation Tool). Individual sea turtles visible in the aerial images were labeled using rectangular bounding boxes aligned with the axes. A single object class (turtles) was defined. All the annotations were then exported in YOLO format, where each instance was represented by a normalized bounding box defined by its center coordinates (xc, yc) and its width and height (w, h) relative to each image (Vijayakumar & Vairavasundaram, 2024).

2.3 Image Tiling and data augmentation

Due to the high spatial resolution and the small relative size of the sea turtle, window-sliding tiling was employed by subdividing each image into 640 × 640 pixels considering an overlap of 20% between adjacent tiles, resulting in a total of 4620 tiles. This overlap was intended to prevent potential truncating of the turtle (Reina et al., 2020). Annotations were then uploaded and geometrically intersected with each tile. Bounding boxes were then recalculated and normalized to YOLO format based on tile dimensions. Tiles without valid turtle annotations were retained as negative samples.



This process yielded only 5.5% positive images of the total dataset. In order to balance this gap, data augmentation of these tiles with existing sea turtles was applied both to the tiles and their corresponding bounding boxes, complying to the YOLO format, using the albumentations library (Buslaev et al., 2020; Nanni et al., 2021). Augmentation procedures were limited to:

- Horizontal flipping with 50% probability
- Rotations (±10 °)
- Random scaling (±10 %)
- Limited contrast enhancement using CLAHE
- Brightness and contrast adjustment
- Low intensity motion blur
- Subtle hue and saturation shifts

To avoid invalid labels, boxes with less than 30% visibility were filtered out. The final augmented tiles were equal to 1275 instances. To ensure balance, 60% of the

negative raw tiles (765 images) were added to final positive raw tiles along with the augmented tiles. Then, the dataset was split into training, validation and test images with a percentage of 70%, 20% and 10%, respectively and it was performed at the parent image level so that each tile originating from the same UAV image were exclusively assigned together in order to prevent any potential spatial data leakage. All the folder data followed the standard YOLO directory structure, with full labelling to each image.

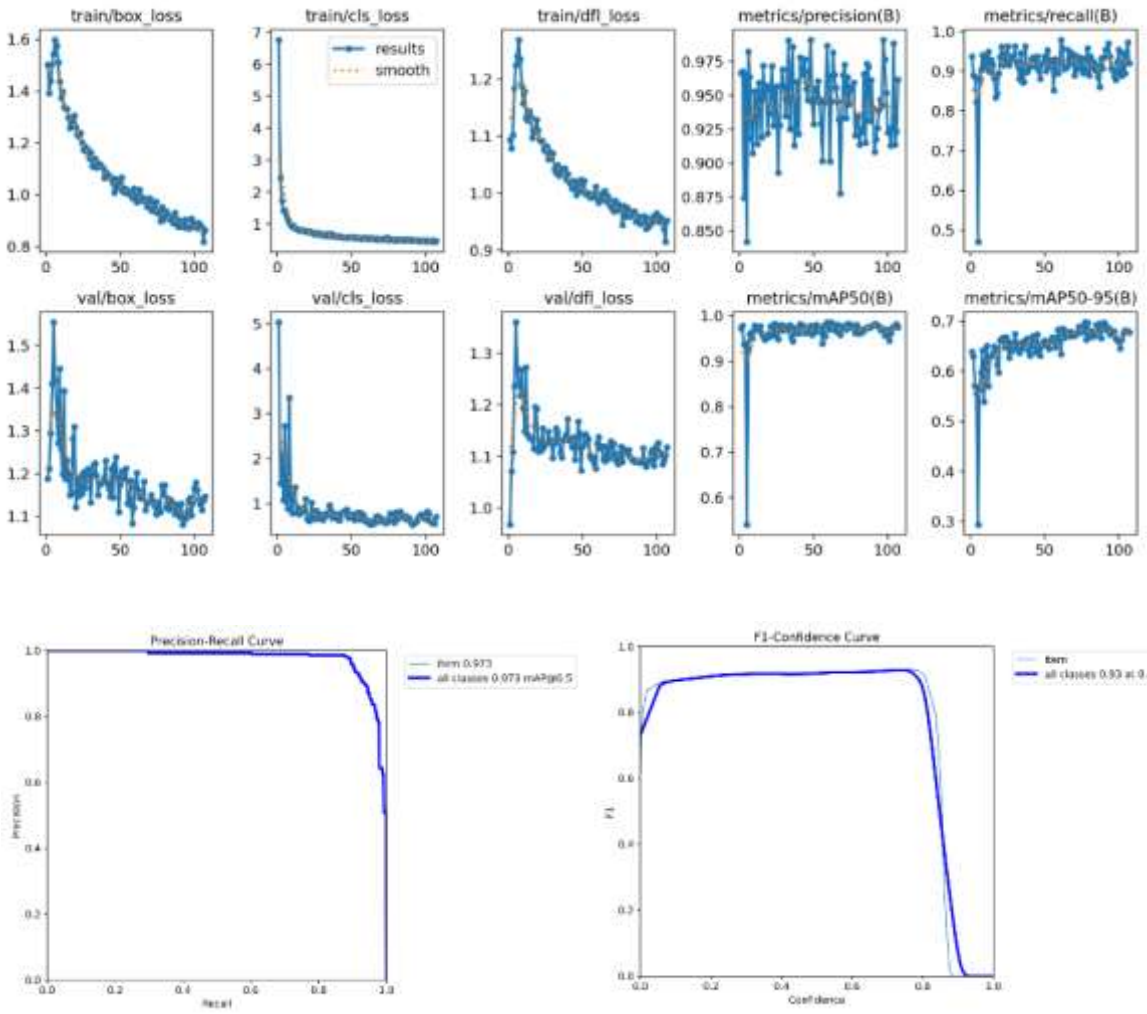
2.4 Model training

A YOLOv8-nano model was fine-tuned for single-class detection using the ultralytics package on PyTorch 2.x (Jocher et al., 2023). Training was conducted for up to 300 epochs with early stopping using patience equal to 30, thus avoiding overfitting (Varghese & Shobana, 2024; Wang et al., 2023). We utilized a Stochastic Gradient Descent (SGD) optimizer with cosine learning rate scheduling (initial LR=0.01) and a batch size of 16 (Zhang et al., 2024). All the analyses were conducted on an NVIDIA RTX 3080 GPU (10 GB VRAM) and Python as our developing environment.

For inference and model testing, the full resolution UAV imagery of the tiles assigned in the testing subset were used and SAHI (Slicing Aided Hyper Inference) was employed (Akyon et al., 2022; Muzammul et al., 2024). Images were subdivided into 640 × 640 pixels with an overlapping of 20% and checked using the best saved checkpoint determined by the highest mAP50-95 on the validation set. Detections were then merged using Non-Maximum Suppression (NMS) with an IoU threshold of 0.5 (Gong et al., 2021; Zaghari et al., 2021). Model performance was evaluated using precision, recall and F1-score (Naidu et al., 2023; Jordan et al., 2008; Reddy & Prasad, 2022). Moreover, mean Average Precision (mAP@50-95) was computed (Zhao, 2024; Guo & Guo, 2023).

3. Results

The YOLOv8n-based detection model achieved strong performance on the test dataset, with a maximum F1-score of 0.93 at an optimal confidence threshold of approximately 0.74. Precision and recall analyses remained consistently high throughout training, reaching approximately 0.97 and 0.93, respectively. In addition, the model achieved a mAP@0.5 of 0.973, reflecting strong detection precision for sea turtles. The lower mAP@50-95 highlighted the difficulty of precise localization of partially submerged sea turtles in certain waters.



4. Discussion

This report demonstrates the strong potential of combining low-cost UAV imagery with deep-learning–based object detection to support the monitoring of marine megafauna, with a particular focus on sea turtles in nearshore coastal environments. The high performance achieved by the YOLOv8-nano model, despite being trained on a relatively small dataset, highlights the feasibility of deploying lightweight, computationally efficient detection pipelines for operational monitoring purposes. Similar findings have been reported in recent UAV-based wildlife studies, where compact convolutional neural networks have proven effective for detecting marine fauna under constrained data conditions (Gray et al., 2019a; Kellenberger et al., 2018).

The strong F1-score and mAP@0.5 values obtained in this study suggest that UAV-based detection systems can reliably identify sea turtles at or near the water surface in shallow coastal habitats. The lower mAP@50–95 values observed are consistent with known challenges in accurately delineating object boundaries in marine environments, especially when animals are partially submerged or visually blended

with the background (Schofield et al., 2017; Hodgson et al., 2018). Nevertheless, from a conservation and management perspective, accurate detection and localization at a broader scale may be more critical than precise bounding box delineation, particularly when the primary objective is presence detection, relative abundance estimation, or spatial pattern analysis.

A key strength of the approach presented here lies in its scalability and cost-effectiveness. The use of low-cost UAV platforms and a lightweight YOLOv8 architecture reduces both financial and technical barriers, making the methodology accessible to conservation practitioners, protected area managers, and small research teams. This is especially important in regions where long-term monitoring budgets are limited, yet pressures on marine megafauna are high. Previous studies have emphasized that UAV-based surveys can substantially reduce costs while maintaining or improving detection accuracy compared to manned aerial surveys (Anderson & Gaston, 2013; Colefax et al., 2017).

The integration of tiling and slicing-aided inference proved essential for detecting small objects such as sea turtles in high-resolution imagery, reinforcing the importance of adapting computer-vision workflows to the specific characteristics of UAV data. The reliance on data augmentation to address class imbalance reflects a common challenge in wildlife detection studies, where positive detections are inherently rare (Hodgson et al., 2016). While augmentation improved model robustness, it also underscores the need for continued data collection across seasons, environmental conditions, and habitats to improve model generalization and reduce false detections.

From a conservation perspective, the implications of this work are substantial. High-resolution, spatially explicit detection data can directly inform the identification of important foraging areas, support impact assessments for coastal developments, and improve the design and evaluation of marine protected areas and other effective area-based conservation measures. In the context of climate change, repeated UAV surveys combined with automated detection offer a powerful means to detect shifts in habitat use and species distribution at fine spatial scales, providing early warning signals of ecological change.

In conclusion, this study provides clear evidence that UAV imagery combined with deep-learning detection pipelines represents a robust, non-invasive, and scalable tool for monitoring sea turtles and other marine megafauna in coastal environments. While challenges remain, particularly in relation to data availability, environmental variability, and model transferability, the approach presented here offers a practical pathway toward routine, data-driven monitoring that can substantially enhance conservation planning and adaptive management. As regulatory frameworks and ethical guidelines for UAV use continue to mature (Hodgson & Koh, 2016), such integrated technological approaches are poised to become a cornerstone of modern marine megafauna conservation.

5. Technical Support

Python packages used

Core Detection Framework

- Ultralytics: Primary object detection model
- PyTorch: Deep learning backend
- SAHI: Slicing strategy for high-resolution drone imagery

Image Processing

- OpenCV (cv2): Image loading, transformation, and manipulation
- Albumentations: Data augmentation for model robustness

Data Processing & Analysis

- NumPy: Numerical operations on detection arrays
- Pandas: Results organization and statistical analysis

Visualization

- Matplotlib: Detection visualization and plotting
- Seaborn: Statistical visualization of results

References

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); October 2022; pp. 966–970.

Anderson, K., & Gaston, K. J. (2013). Lightweight unmanned aerial vehicles will revolutionize spatial ecology. Frontiers in Ecology and the Environment, 11(3), 138-146.

Bevan, E., Whiting, S., Tucker, T., Guinea, M., Raith, A., & Douglas, R. (2018). Measuring behavioral responses of sea turtles, saltwater crocodiles, and crested terns to drone disturbance to define ethical operating thresholds. PLOS ONE, 13(3), e0194460.

Bolten, A. B. (2003). Variation in sea turtle life history patterns: Neritic vs. oceanic developmental stages. In The Biology of Sea Turtles (Vol. 2, pp. 243-257).

Buckland, S. T., Anderson, D. R., Burnham, K. P., Laake, J. L., Borchers, D. L., & Thomas, L. (2001). Introduction to Distance Sampling: Estimating Abundance of Biological Populations. Oxford University Press.

Buslaev, A.; Iglovikov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. Albumentations: Fast and Flexible Image Augmentations. Information 2020, 11, 125, doi:10.3390/info11020125.

Chabot, D., & Bird, D. M. (2015). Wildlife research and management methods in the 21st century: Where do unmanned aircraft fit in? Journal of Unmanned Vehicle Systems, 3(4), 137-155.

Christiansen, F., Dujon, A. M., Sprogis, K. R., Arnould, J. P., & Bejder, L. (2016). Noninvasive unmanned aerial vehicle provides estimates of the energetic cost of reproduction in humpback whales. Ecosphere, 7(10), e01468.

Colefax, A. P., Butcher, P. A., & Kelaher, B. P. (2017). The potential for unmanned aerial vehicles (UAVs) to conduct marine fauna surveys in place of manned aircraft. ICES Journal of Marine Science, 75(1), 1-8.

Corrigan, C. M., Ardron, J. A., Comeros-Raynal, M. T., Hoyt, E., Notarbartolo di Sciara, G., & Carpenter, K. E. (2014). Developing important marine mammal area criteria: Learning from ecologically or biologically significant areas and key biodiversity areas. Aquatic Conservation: Marine and Freshwater Ecosystems, 24(S2), 166-183.

Davidson, A. D., Boyer, A. G., Kim, H., Pompa-Mansilla, S., Hamilton, M. J., Costa, D. P., ... & Brown, J. H. (2012). Drivers and hotspots of extinction risk in marine mammals. Proceedings of the National Academy of Sciences, 109(9), 3395-3400.

Dickson, L. C., Negus, S. R., Eizaguirre, C., Katselidis, K. A., & Schofield, G. (2022b). Aerial drone surveys reveal the efficacy of a protected area network for marine megafauna and the value of sea turtles as umbrella species. Drones, 6(10), 291.

Dickson, L. C., Tugwell, H., Katselidis, K. A., & Schofield, G. (2022a). Aerial drones reveal the dynamic structuring of sea turtle breeding aggregations and minimum survey effort required to capture climatic and sex-specific effects. Frontiers in Marine Science, 9, 864694.

Godley, B. J., Blumenthal, J. M., Broderick, A. C., Coyne, M. S., Godfrey, M. H., Hawkes, L. A., & Witt, M. J. (2008). Satellite tracking of sea turtles: Where have we been and where do we go next? Endangered Species Research, 4(1-2), 3-22.

Gong, M.; Wang, D.; Zhao, X.; Guo, H.; Luo, D.; Song, M. A Review of Non-Maximum Suppression Algorithms for Deep Learning Target Detection. In Proceedings of the Seventh Symposium on Novel Photoelectronic Detection Technology and Applications; SPIE, March 12 2021; Vol. 11763, pp. 821–828.

Gray, P. C., Bierlich, K. C., Mantell, S. A., Friedlaender, A. S., Goldbogen, J. A., & Johnston, D. W. (2019a). Drones and convolutional neural networks facilitate automated and accurate cetacean species identification and photogrammetry. Methods in Ecology and Evolution, 10(9), 1490-1500.

Gray, P. C., Fleishman, A. B., Klein, D. J., McKown, M. W., Bezy, V. S., Lohmann, K. J., & Johnston, D. W. (2019b). A convolutional neural network for detecting sea turtles in drone imagery. Methods in Ecology and Evolution, 10(3), 345-355

Guo, Z.; Guo, L. Small Target Detection in Aerial Photography Based on Improved YOLOv5. In Proceedings of the 2023 3rd International Conference on Robotics, Automation and Intelligent Control (ICRAIC); November 2023; pp. 409–413.

Hawkes, L. A., Broderick, A. C., Godfrey, M. H., & Godley, B. J. (2009). Climate change and marine turtles. Endangered Species Research, 7(2), 137-154.

Hays, G. C., & Scott, R. (2013). Global patterns for upper ceilings on migration distance in sea turtles and comparisons with fish, birds and mammals. Functional Ecology, 27(3), 748-756.

Hays, G. C., Broderick, A. C., Glen, F., Godley, B. J., Houghton, J. D., & Metcalfe, J. D. (2002). Water temperature and internesting intervals for loggerhead (Caretta caretta) and green (Chelonia mydas) sea turtles. Journal of Thermal Biology, 27(5), 429-432.

Hays, G. C., Ferreira, L. C., Sequeira, A. M., Meekan, M. G., Duarte, C. M., Bailey, H., ... & Thums, M. (2016). Key questions in marine megafauna movement ecology. Trends in Ecology & Evolution, 31(6), 463-475.

Hazel, J., Lawler, I. R., Marsh, H., & Robson, S. (2007). Vessel speed increases collision risk for the green turtle Chelonia mydas. Endangered Species Research, 3(2), 105-113.

Hazen, E. L., Jorgensen, S., Rykaczewski, R. R., Bograd, S. J., Foley, D. G., Jonsen, I. D., ... & Block, B. A. (2013). Predicted habitat shifts of Pacific top predators in a changing climate. Nature Climate Change, 3(3), 234-238.

Heithaus, M. R., Frid, A., Wirsing, A. J., & Worm, B. (2008). Predicting ecological consequences of marine top predator declines. Trends in Ecology & Evolution, 23(4), 202-210.

Hodgson, A., Kelly, N., & Peel, D. (2013). Unmanned aerial vehicles (UAVs) for surveying marine fauna: A dugong case study. PLOS ONE, 8(11), e79556.

Hodgson, J. C., & Koh, L. P. (2016). Best practice for minimising unmanned aerial vehicle disturbance to wildlife in biological field research. Current Biology, 26(10), R404-R405.

Hodgson, J. C., Baylis, S. M., Mott, R., Herrod, A., & Clarke, R. H. (2016). Precision wildlife monitoring using unmanned aerial vehicles. Scientific Reports, 6(1), 22574.

Hodgson, J. C., Mott, R., Baylis, S. M., Pham, T. T., Wotherspoon, S., Kilpatrick, A. D., ... & Koh, L. P. (2018). Drones count wildlife more accurately and precisely than humans. Methods in Ecology and Evolution, 9(5), 1160-1167.

IUCN. (2023). The IUCN Red List of Threatened Species. Version 2023-1. https://www.iucnredlist.org

Jeffers, V. F., & Godley, B. J. (2016). Satellite tracking in sea turtles: How do we find our way to the conservation dividends?. Biological Conservation, 199, 172-184.

Jocher, G.; Qiu, J.; Chaurasia, A. Ultralytics YOLO 2023.

Johnston, D. W., Dale, J., Murray, K. T., Josephson, E., Newton, E., & Wood, S. (2017). Comparing occupied and unoccupied aircraft surveys of wildlife populations: Assessing the gray seal (Halichoerus grypus) breeding colony on Muskeget Island, USA. Journal of Unmanned Vehicle Systems, 5(4), 178-191.

Jordan, F.; Jelks, H.L.; Bortone, S.A.; Dorazio, R.M. Comparison of Visual Survey and Seining Methods for Estimating Abundance of an Endangered, Benthic Stream Fish. Environ Biol Fish 2008, 81, 313–319, doi:10.1007/s10641-007-9202-0.

Kellenberger, B., Marcos, D., & Tuia, D. (2018). Detecting mammals in UAV images: Best practices to address a substantially imbalanced dataset with deep learning. Remote Sensing of Environment, 216, 139-153.

Kiszka, J. J., Mourier, J., Gastrich, K., & Heithaus, M. R. (2016). Using unmanned aerial vehicles (UAVs) to investigate shark and ray densities in a shallow coral lagoon. Marine Ecology Progress Series, 560, 237-242.

Lewison, R. L., Crowder, L. B., Wallace, B. P., Moore, J. E., Cox, T., Zydelis, R., ... & Safina, C. (2014). Global patterns of marine mammal, seabird, and sea turtle bycatch reveal taxa-specific and cumulative megafauna hotspots. Proceedings of the National Academy of Sciences, 111(14), 5271-5276.

Linchant, J., Lisein, J., Semeki, J., Lejeune, P., & Vermeulen, C. (2015). Are unmanned aircraft systems (UASs) the future of wildlife monitoring? A review of accomplishments and challenges. Mammal Review, 45(4), 239-252.

Marsh, H., & Sinclair, D. F. (1989). Correcting for visibility bias in strip transect aerial surveys of aquatic fauna. The Journal of Wildlife Management, 53(4), 1017-1024.

Martin, C., Parkes, M., Wehking, J., Oprandi, A., Zhang, A., Caleb, J., ... & Jaffe, J. S. (2020). Determining the position, orientation, and swimming trajectories of a whale shark using a small remotely piloted aerial system. Methods in Ecology and Evolution, 11(12), 1574-1585.

Maxwell, S. M., Hazen, E. L., Lewison, R. L., Dunn, D. C., Bailey, H., Bograd, S. J., ... & Costa, D. P. (2015). Dynamic ocean management: Defining and conceptualizing real-time management of the ocean. Marine Policy, 58, 42-50.

Mazaris, A. D., Schofield, G., Gkazinou, C., Almpanidou, V., & Hays, G. C. (2017). Global sea turtle conservation successes. Science Advances, 3(9), e1600730.

McClellan, C. M., & Read, A. J. (2007). Complexity and variation in loggerhead sea turtle life history. Biology Letters, 3(6), 592-594.

Muzammul, M.; Algarni, A.; Ghadi, Y.Y.; Assam, M. Enhancing UAV Aerial Image Analysis: Integrating Advanced SAHI Techniques With Real-Time Detection Models on the VisDrone Dataset. IEEE Access 2024, 12, 21621–21633, doi:10.1109/ACCESS.2024.3363413.

Naidu, G.; Zuva, T.; Sibanda, E.M. A Review of Evaluation Metrics in Machine Learning Algorithms. In Proceedings of the Artificial Intelligence Application in Networks and Systems; Silhavy, R., Silhavy, P., Eds.; Springer International Publishing: Cham, 2023; pp. 15–25.

Nanni, L.; Paci, M.; Brahnam, S.; Lumini, A. Comparison of Different Image Data Augmentation Approaches. Journal of Imaging 2021, 7, 254, doi:10.3390/jimaging7120254.

Papazekou, M.; Dimitriadis, C.; Dalla, D.; Comis, C.M.; Spinos, E.; Vavasis, C.; Kapellaki, K.; Michalopoulou, A.; Valli, A.-T.; Barelos, D.; et al. The Ionian Sea in the Eastern

Mediterranean: Critical Year-Round Habitats for Sea Turtles and Diverse Marine Megafauna, Spanning All Life Stages and Genders. Ocean & Coastal Management 2024, 251, 107054, doi:10.1016/j.ocecoaman.2024.107054.

Pollock, K. H., Marsh, H. D., Lawler, I. R., & Alldredge, M. W. (2006). Estimating animal abundance in heterogeneous environments: An application to aerial surveys for dugongs. The Journal of Wildlife Management, 70(1), 255-262.

Poloczanska, E. S., Limpus, C. J., & Hays, G. C. (2009). Vulnerability of marine turtles to climate change. Advances in Marine Biology, 56, 151-211.

Raoult, V., Colefax, A. P., Allan, B. M., Cagnazzi, D., Castelblanco-Martínez, N., Ierodiaconou, D., ... & Butcher, P. A. (2020). Operational protocols for the use of drones in marine animal research. Drones, 4(4), 64.

Reddy, B.H.; R, K.P. Classification of Fire and Smoke Images Using Decision Tree Algorithm in Comparison with Logistic Regression to Measure Accuracy, Precision, Recall, F-Score. In Proceedings of the 2022 14th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS); November 2022; pp. 1–5.

Rees, A. F., Alfaro-Shigueto, J., Barata, P. C., Bjorndal, K. A., Bolten, A. B., Bourjea, J., ... & Godley, B. J. (2016). Are we working towards global research priorities for management and conservation of sea turtles? Endangered Species Research, 31, 337-382.

Rees, A. F., Carreras, C., Broderick, A. C., Margaritoulis, D., Stringell, T. B., & Godley, B. J. (2017). Linking loggerhead locations: Using multiple methods to determine the origin of sea turtles in feeding grounds. Marine Biology, 164(1), 30.

Rees, A. F., Jony, M., Marella, P., & Papathanasopoulou, N. (2018). The potential of unmanned aerial systems for sea turtle research and conservation: A review and future directions. Endangered Species Research, 35, 81-100.

Reina, G.A.; Panchumarthy, R.; Thakur, S.P.; Bastidas, A.; Bakas, S. Systematic Evaluation of Image Tiling Adverse Effects on Deep Learning Semantic Segmentation. Front. Neurosci. 2020, 14, doi:10.3389/fnins.2020.00065.

Schofield, G., Bishop, C. M., MacLean, G., Brown, P., Baker, M., Katselidis, K. A., ... & Hays, G. C. (2007). Novel GPS tracking of sea turtles as a tool for conservation management. Journal of Experimental Marine Biology and Ecology, 347(1-2), 58-68.

Schofield, G., Esteban, N., Katselidis, K. A., & Hays, G. C. (2019a). Drones for research on sea turtles and other marine vertebrates–A review. Biological Conservation, 238, 108214.

Schofield, G., Katselidis, K. A., Lilley, M. K., Reina, R. D., & Hays, G. C. (2017). Detecting elusive aspects of wildlife ecology using drones: New insights on the mating dynamics and operational sex ratios of sea turtles. Functional Ecology, 31(12), 2310-2319.

Schofield, G., Papafitsoros, K., Haughey, R., & Katselidis, K. (2019b). Aerial and underwater surveys reveal temporal variation in cleaning-station use by sea turtles at a temperate breeding area. Marine Ecology Progress Series, 575, 153-164.

Schofield, G., Scott, R., Dimadi, A., Fossette, S., Katselidis, K. A., Koutsoubas, D., ... & Hays, G. C. (2013). Evidence-based marine protected area planning for a highly mobile endangered marine vertebrate. Biological Conservation, 161, 101-109.

Seminoff, J. A., Allen, C. D., Balazs, G. H., Dutton, P. H., Eguchi, T., Haas, H. L., ... & Zárate, P. (2014). Status review of the green turtle (Chelonia mydas) under the Endangered Species Act. NOAA Technical Memorandum, NOAA-NMFS-SWFSC-539.

Thums, M., Whiting, S. D., Reisser, J. W., Pendoley, K. L., Pattiaratchi, C. B., Proietti, M., ... & Meekan, M. G. (2013). Tracking sea turtle hatchlings—A pilot study using acoustic telemetry. Journal of Experimental Marine Biology and Ecology, 440, 156-163.

Varghese, R.; M., S. YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness. In Proceedings of the 2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS); April 2024; pp. 1–6.

Vas, E., Lescroël, A., Duriez, O., Boguszewski, G., & Grémillet, D. (2015). Approaching birds with drones: First experiments and ethical guidelines. Biology Letters, 11(2), 20140754.

Vijayakumar, A.; Vairavasundaram, S. YOLO-Based Object Detection Models: A Review and Its Applications. Multimed Tools Appl 2024, 83, 83535–83574, doi:10.1007/s11042-024-18872-y.

Wallace, B. P., DiMatteo, A. D., Bolten, A. B., Chaloupka, M. Y., Hutchinson, B. J., Abreu-Grobois, F. A., ... & Mast, R. B. (2011). Global conservation priorities for marine turtles. PLOS ONE, 6(9), e24510.

Wallace, B. P., DiMatteo, A. D., Hurley, B. J., Finkbeiner, E. M., Bolten, A. B., Chaloupka, M. Y., ... & Spotila, J. R. (2010). Regional management units for marine turtles: A novel framework for prioritizing conservation and research across multiple scales. PLOS ONE, 5(12), e15465.

Wang, G.; Chen, Y.; An, P.; Hong, H.; Hu, J.; Huang, T. UAV-YOLOv8: A Small-Object-Detection Model Based on Improved YOLOv8 for UAV Aerial Photography Scenarios. Sensors 2023, 23, 7190, doi:10.3390/s23167190.

Weinstein, B. G. (2018). A computer vision for animal ecology. Journal of Animal Ecology, 87(3), 533-545.

Zaghari, N.; Fathy, M.; Jameii, S.M.; Shahverdy, M. The Improvement in Obstacle Detection in Autonomous Vehicles Using YOLO Non-Maximum Suppression Fuzzy Algorithm. J Supercomput 2021, 77, 13421–13446, doi:10.1007/s11227-021-03813-5.

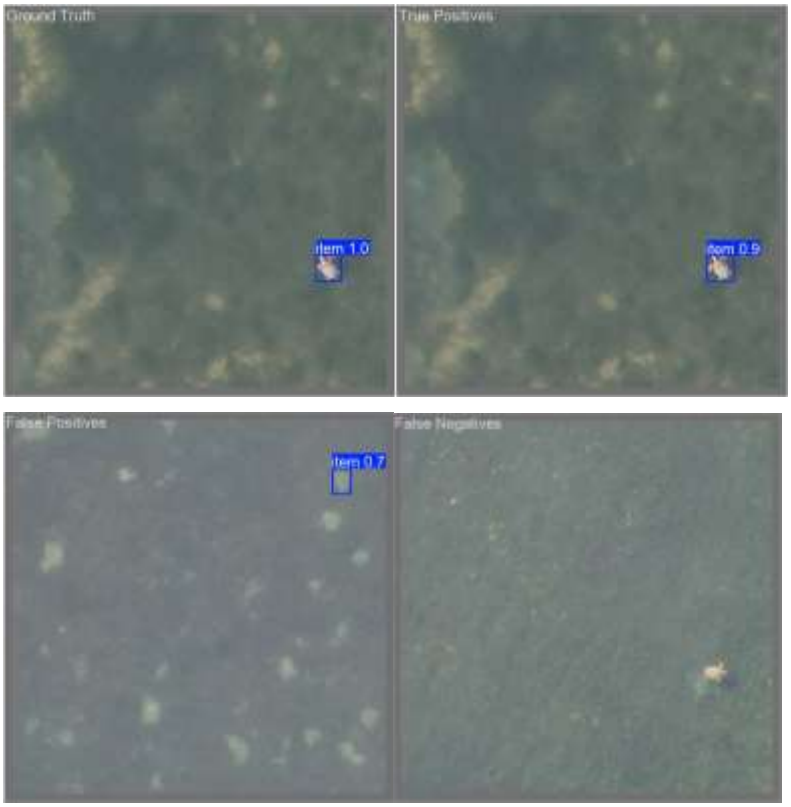
Zhang, Z.; Qu, Y.; Wang, T.; Rao, Y.; Jiang, D.; Li, S.; Wang, Y. An Improved YOLOv8n Used for Fish Detection in Natural Water Environments. Animals 2024, 14, 2022, doi:10.3390/ani14142022.

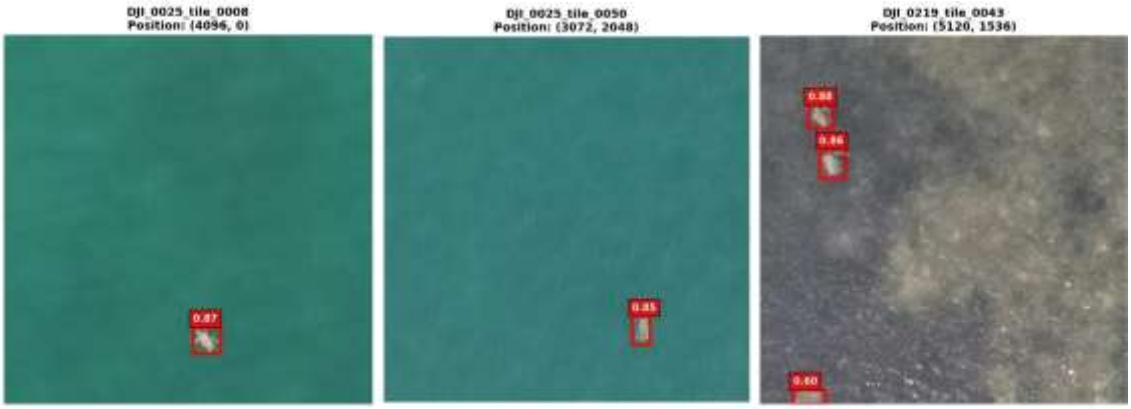
Zhao, Y. Research on Adaptive Weight and Frequency Domain Enhancement Fusion Method for Small Target Detection. In Proceedings of the 2024 IEEE 2nd International Conference on Electrical, Automation and Computer Engineering (ICEACE); December 2024; pp. 186–190.

Appendices

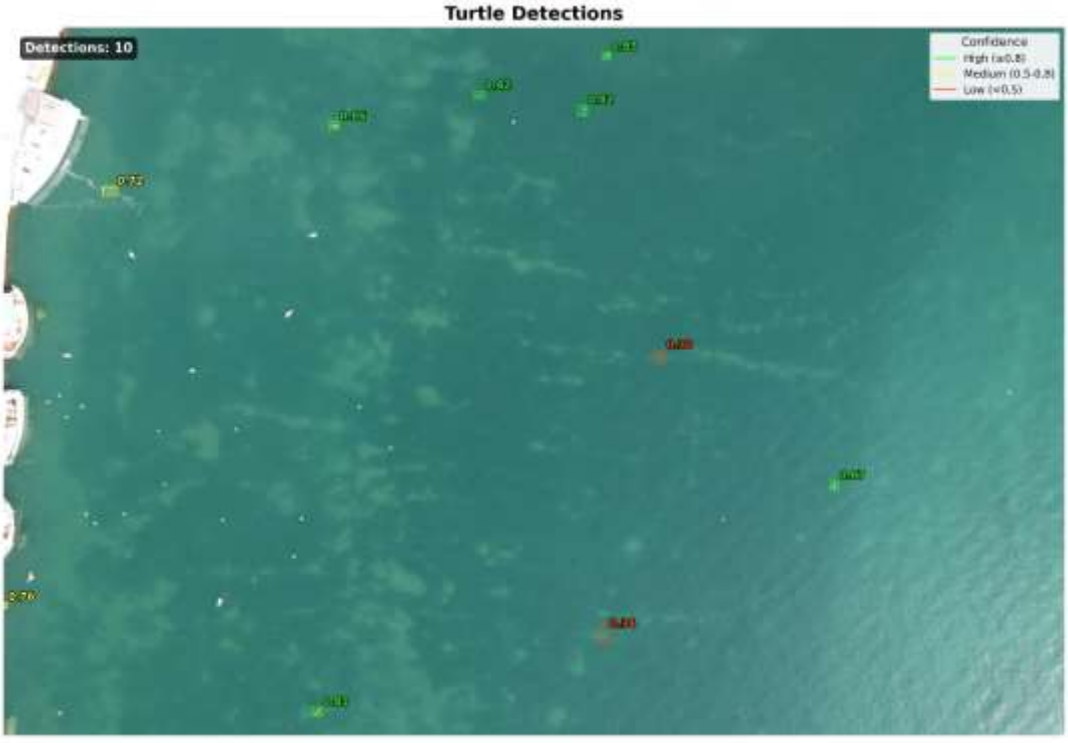
Appendix I

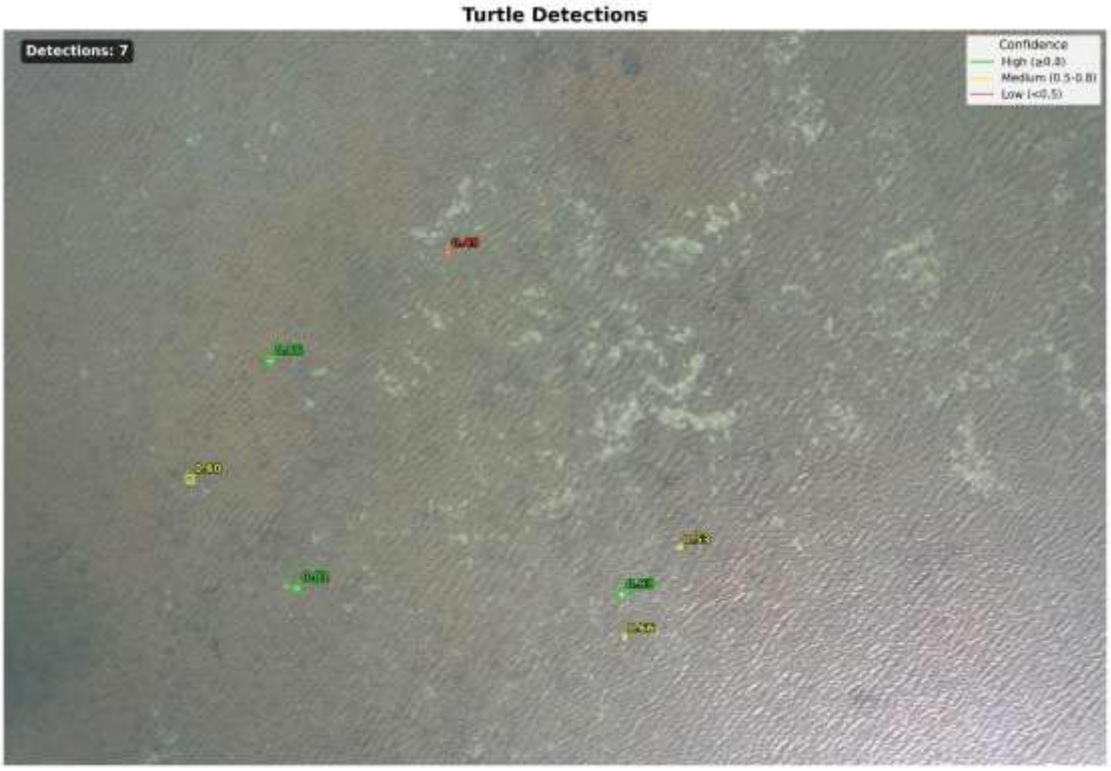
Examples of detection results:

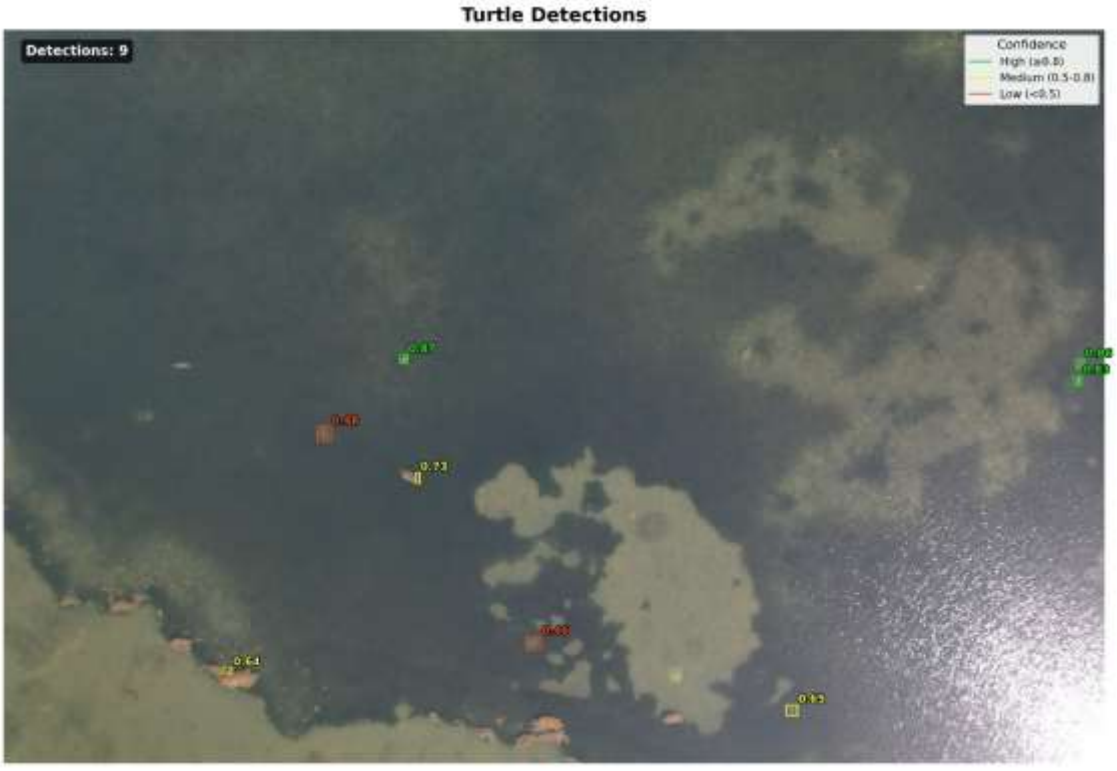





Turtle Detections
Detections: 2
Confidence
High (≥0.8)
Medium (0.5-0.8)
Low (<0.5)


Turtle Detections
Detections: 10
Confidence
High (≥0.8)
Medium (0.5-0.8)
Low (<0.5)


Turtle Detections
Detections: 7
Confidence
High (≥0.8)
Medium (0.5-0.8)
Low (<0.5)


Turtle Detections
Detections: 9
Confidence
High (≥0.8)
Medium (0.5-0.8)
Low (<0.5)


Turtle Detections
Detections: 4
Confidence


Turtle Detections
Detections: 2
Confidence

Appendix II

The code used in Python.

```
from pathlib import Path
# Base dataset path
DATA_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\raw")
IMAGES_DIR = DATA_ROOT / "images"
LABELS_DIR = DATA_ROOT / "labels"
# Print paths
print(f"Dataset root : {DATA_ROOT}")
print(f"Images path  : {IMAGES_DIR}")
print(f"Labels path  : {LABELS_DIR}")
print()
# Collect files
image_files = sorted([p for p in IMAGES_DIR.iterdir() if p.suffix.lower() in [".jpg", ".jpeg", ".png"]])
label_files = sorted(LABELS_DIR.glob("*.txt"))
print(f" Images found: {len(image_files)}")
print(f"Labels found: {len(label_files)}")

# Verify Labels on a Sample Image (YOLO format)
sample_image = image_files[0]
sample_label = LABELS_DIR / f"{sample_image.stem}.txt"
print(f"Image : {sample_image.name}")
print(f"Label : {sample_label.name}")
# Load image
img = cv2.imread(str(sample_image))
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
h, w = img.shape[:2]
if not sample_label.exists():
    print ("Label file not found for this image")
    plt.figure(figsize=(10, 8))
    plt.imshow(img)
    plt.title("No label found", color="red")
    plt.axis("off")
```

```
    plt.show()
else:
    print(" Label found")
    with open(sample_label, "r") as f:
        lines = f.readlines()
    print(f"Turtle instances: {len(lines)}")
    for i, line in enumerate(lines):
        class_id, x_c, y_c, bw, bh = map(float, line.split())
        # Convert YOLO → pixel coordinates
        x1 = int((x_c - bw / 2) * w)
        y1 = int((y_c - bh / 2) * h)
        x2 = int((x_c + bw / 2) * w)
        y2 = int((y_c + bh / 2) * h)
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(
            img,
            f"Turtle {i+1}",
            (x1, y1 - 8),
            cv2.FONT_HERSHEY_SIMPLEX,
            2,
            (0, 255, 0),
            3, )
        print(f"  Turtle {i+1}: class={int(class_id)}")
    plt.figure(figsize=(10, 8))
    plt.imshow(img)
    plt.title(f"{sample_image.name} — {len(lines)} turtles")
    plt.axis("off")
    plt.show()
```

#Tiling

```
from pathlib import Path
import cv2
import numpy as np
# Paths
DATA_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\raw")
IMAGES_DIR = DATA_ROOT / "images"
LABELS_DIR = DATA_ROOT / "labels"
```

```
OUTPUT_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\tiled")
OUT_IMAGES = OUTPUT_ROOT / "images"
OUT_LABELS = OUTPUT_ROOT / "labels"
# Tiling parameters
TILE_SIZE = 640
OVERLAP = 0.20
STRIDE = int(TILE_SIZE * (1 - OVERLAP))
print("Tile size:", TILE_SIZE)
print("Overlap:", OVERLAP)
print("Stride:", STRIDE)

def load_yolo_labels(label_path):
    """Load YOLO labels from file"""
    labels = []
    if not label_path.exists():
        return labels
    with open(label_path, "r") as f:
        for line in f:
            parts = line.strip().split()
            if len(parts) == 5:
                labels.append(list(map(float, parts)))
    return labels
def yolo_to_pixel(label, img_w, img_h):
    """YOLO → pixel bbox"""
    _, xc, yc, w, h = label
    bw = w * img_w
    bh = h * img_h
    cx = xc * img_w
    cy = yc * img_h
    x1 = cx - bw / 2
    y1 = cy - bh / 2
    x2 = cx + bw / 2
    y2 = cy + bh / 2
    return x1, y1, x2, y2
def tile_image_with_labels(image_path):
```

```
    image = cv2.imread(str(image_path))
    h, w = image.shape[:2]
    label_path = LABELS_DIR / f"{image_path.stem}.txt"
    labels = load_yolo_labels(label_path)
    tiles_x = list(range(0, max(w - TILE_SIZE, 0) + 1, STRIDE))
    tiles_y = list(range(0, max(h - TILE_SIZE, 0) + 1, STRIDE))
    if tiles_x[-1] != w - TILE_SIZE:
        tiles_x.append(w - TILE_SIZE)
    if tiles_y[-1] != h - TILE_SIZE:
        tiles_y.append(h - TILE_SIZE)
    tile_count = 0
    for y in tiles_y:
        for x in tiles_x:
            tile = image[y:y+TILE_SIZE, x:x+TILE_SIZE].copy()
            tile_labels = []
            for lbl in labels:
                cls, xc, yc, bw, bh = lbl
                bx1, by1, bx2, by2 = yolo_to_pixel(lbl, w, h)
                # Intersection with tile
                ix1 = max(x, bx1)
                iy1 = max(y, by1)
                ix2 = min(x + TILE_SIZE, bx2)
                iy2 = min(y + TILE_SIZE, by2)
                if ix1 < ix2 and iy1 < iy2:
                    iw = ix2 - ix1
                    ih = iy2 - iy1
                    inter_area = iw * ih
                    box_area = (bx2 - bx1) * (by2 - by1)
                    if inter_area / box_area >= 0.3:
                        # Convert to tile-normalized YOLO
                        txc = ((ix1 + ix2) / 2 - x) / TILE_SIZE
                        tyc = ((iy1 + iy2) / 2 - y) / TILE_SIZE
                        tw = iw / TILE_SIZE
                        th = ih / TILE_SIZE
                        tile_labels.append([cls, txc, tyc, tw, th])
            # Save tile
```

```
            OUT_IMAGES.mkdir(parents=True, exist_ok=True)
            OUT_LABELS.mkdir(parents=True, exist_ok=True)
            tile_name = f"{image_path.stem}_tile_{tile_count:04d}"
            cv2.imwrite(str(OUT_IMAGES / f"{tile_name}.jpg"), tile)
            with open(OUT_LABELS / f"{tile_name}.txt", "w") as f:
                for l in tile_labels:
                    f.write(f"{int(l[0])} {l[1]:.6f} {l[2]:.6f} {l[3]:.6f} {l[4]:.6f}\n")
            tile_count += 1
    print(f" {image_path.name}: {tile_count} tiles created")
image_files = sorted(IMAGES_DIR.glob("*.jpg"))
for img_path in image_files:
    tile_image_with_labels(img_path)

import cv2
import random
import matplotlib.pyplot as plt
from pathlib import Path
TILED_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\tiled")
TILED_IMAGES = TILED_ROOT / "images"
TILED_LABELS = TILED_ROOT / "labels"
# All tiled images
tile_images = sorted(TILED_IMAGES.glob("*.jpg"))
print(f"Total tiled images: {len(tile_images)}")
# Pick tiles that HAVE labels
labeled_tiles = [
    img for img in tile_images
    if (TILED_LABELS / f"{img.stem}.txt").exists()
    and (TILED_LABELS / f"{img.stem}.txt").stat().st_size > 0]
print(f"Tiled images with turtles: {len(labeled_tiles)}")

#Augmentation of sea turtle only
import cv2
import random
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
```

```
import albumentations as A
# PATHS
TILED_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\tiled")
TILED_IMAGES = TILED_ROOT / "images"
TILED_LABELS = TILED_ROOT / "labels"
AUG_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\augmented")
AUG_IMAGES = AUG_ROOT / "images"
AUG_LABELS = AUG_ROOT / "labels"
AUG_IMAGES.mkdir(parents=True, exist_ok=True)
AUG_LABELS.mkdir(parents=True, exist_ok=True)

tile_images = sorted(TILED_IMAGES.glob("*.jpg"))
labeled_tiles = [
    img for img in tile_images
    if (TILED_LABELS / f"{img.stem}.txt").exists()
    and (TILED_LABELS / f"{img.stem}.txt").stat().st_size > 0]
print(f"Total tiles: {len(tile_images)}")
print(f"Tiles with turtles: {len(labeled_tiles)}")

def read_yolo_labels(label_path):
    labels = []
    with open(label_path, "r") as f:
        for line in f:
            parts = line.strip().split()
            if len(parts) == 5:
                labels.append(list(map(float, parts)))
    return labels
def get_augmentation_pipeline():
    return A.Compose(
        [
            # Geometry
            A.HorizontalFlip(p=0.5),
            A.Rotate(
                limit=10,
                p=0.4,
                border_mode=cv2.BORDER_REFLECT_101
```

```
        ),
        A.RandomScale(scale_limit=0.10, p=0.4),

        # Lighting & color (VERY MODERATE)
        A.RandomBrightnessContrast(
            brightness_limit=0.15,
            contrast_limit=0.15,
            p=0.5
        ),
        A.HueSaturationValue(
            hue_shift_limit=5,
            sat_shift_limit=10,
            val_shift_limit=10,
            p=0.3
        ),

        # Noise & blur (EXTREMELY LIGHT)
        #A.GaussNoise(var_limit=(2, 6), p=0.15),
        A.MotionBlur(blur_limit=3, p=0.15),

        # Contrast enhancement (low probability)
        A.CLAHE(
            clip_limit=1.5,
            tile_grid_size=(8, 8),
            p=0.2
        ), ],
    bbox_params=A.BboxParams(
        format="yolo",
        label_fields=["class_labels"],
        min_visibility=0.3 ) )

def visualize_augmented_tile():
    transform = get_augmentation_pipeline()

    img_path = random.choice(labeled_tiles)
    lbl_path = TILED_LABELS / f"{img_path.stem}.txt"
```

```
    image = cv2.imread(str(img_path))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # Read YOLO labels: [cls, xc, yc, w, h]
    labels = []
    with open(lbl_path, "r") as f:
        for line in f:
            labels.append(list(map(float, line.split())))

    bboxes = [l[1:] for l in labels]
    class_labels = [int(l[0]) for l in labels]

    augmented = transform(
        image=image,
        bboxes=bboxes,
        class_labels=class_labels
    )
    aug_img = augmented["image"]
    aug_boxes = augmented["bboxes"]
    h, w = aug_img.shape[:2]
    plt.figure(figsize=(6, 6))
plt.imshow(aug_img)
    for box in aug_boxes:
        xc, yc, bw, bh = box
        x1 = (xc - bw / 2) * w
        y1 = (yc - bh / 2) * h
        x2 = (xc + bw / 2) * w
        y2 = (yc + bh / 2) * h

        plt.gca().add_patch(
            plt.Rectangle(
                (x1, y1),
                x2 - x1,
                y2 - y1,
                edgecolor="lime",
```

```
            facecolor="none",
            linewidth=2
        ) )
    plt.title("Augmented Turtle Tile (Verification)")
    plt.axis("off")
    plt.show()

def create_augmented_dataset(augment_per_image=5):
    transform = get_augmentation_pipeline()

    AUG_IMAGES.mkdir(parents=True, exist_ok=True)
    AUG_LABELS.mkdir(parents=True, exist_ok=True)

    for img_path in labeled_tiles:
        lbl_path = TILED_LABELS / f"{img_path.stem}.txt"

        image = cv2.imread(str(img_path))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

        labels = []
        with open(lbl_path, "r") as f:
            for line in f:
                labels.append(list(map(float, line.split())))

        bboxes = [l[1:] for l in labels]
        class_labels = [int(l[0]) for l in labels]
        for i in range(augment_per_image):
            augmented = transform(
                image=image,
                bboxes=bboxes,
                class_labels=class_labels
            )
            if not augmented["bboxes"]:
                continue
            aug_img = cv2.cvtColor(
                augmented["image"], cv2.COLOR_RGB2BGR
```

```
            )
            img_name = f"{img_path.stem}_aug{i}.jpg"
            lbl_name = f"{img_path.stem}_aug{i}.txt"
            cv2.imwrite(str(AUG_IMAGES / img_name), aug_img)
            with open(AUG_LABELS / lbl_name, "w") as f:
                for cls, box in zip(
                    augmented["class_labels"],
                    augmented["bboxes"]
                ):
                    f.write(
                        f"{cls} {box[0]:.6f} {box[1]:.6f} "
                        f"{box[2]:.6f} {box[3]:.6f}\n"
                    )
# generate augmented dataset
create_augmented_dataset(augment_per_image=5)
# stats
print(f"Original turtle tiles : {len(labeled_tiles)}")
print(f"Augmented images      : {len(list(AUG_IMAGES.glob('*.jpg')))}")
print(f"Augmented labels      : {len(list(AUG_LABELS.glob('*.txt')))}")

#Prepare dataset for YOLO
import random
import shutil
from pathlib import Path
random.seed(42)  # reproducibility
# Original tiled data
TILED_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\tiled")
TILED_IMAGES = TILED_ROOT / "images"
TILED_LABELS = TILED_ROOT / "labels"
# Augmented data
AUG_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\augmented")
AUG_IMAGES = AUG_ROOT / "images"
AUG_LABELS = AUG_ROOT / "labels"
# Final YOLO dataset
FINAL_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\yolo_dataset")
positive_images = []
```

```
# Original turtle tiles
for img in TILED_IMAGES.glob("*.jpg"):
    lbl = TILED_LABELS / f"{img.stem}.txt"
    if lbl.exists() and lbl.stat().st_size > 0:
        positive_images.append((img, lbl))
# Augmented turtle tiles
for img in AUG_IMAGES.glob("*.jpg"):
    lbl = AUG_LABELS / f"{img.stem}.txt"
    if lbl.exists() and lbl.stat().st_size > 0:
        positive_images.append((img, lbl))
empty_images = []
for img in TILED_IMAGES.glob("*.jpg"):
    lbl = TILED_LABELS / f"{img.stem}.txt"
    if not lbl.exists() or lbl.stat().st_size == 0:
        empty_images.append((img, None))
print(f"Available empty tiles: {len(empty_images)}")
NUM_EMPTY = 765
selected_empty = random.sample(empty_images, NUM_EMPTY)
import random
import shutil
from pathlib import Path
import re

# Set random seed for reproducibility
random.seed(42)

# Paths
TILED_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\tiled")
TILED_IMAGES = TILED_ROOT / "images"
TILED_LABELS = TILED_ROOT / "labels"

AUG_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\augmented")
AUG_IMAGES = AUG_ROOT / "images"
AUG_LABELS = AUG_ROOT / "labels"
```

```
# New dataset directory
FINAL_ROOT = Path(r"C:\turtle_detection\turtle_2025\data\yolo_dataset_balanced")

def extract_parent_image_id(filename):
    """Extract parent image ID from tile filename."""
    name = filename.replace('.jpg', '')
    if '_tile_' in name:
        return name.split('_tile_')[0]
    match = re.match(r'(DJI_\d+)', name)
    return match.group(1) if match else name

def build_balanced_dataset(target_pos_ratio=0.3):
    """
    Build dataset with balanced positive/negative ratio.
    target_pos_ratio: Target percentage of tiles WITH turtles (e.g., 0.3 = 30%)
    """

    print(" Collecting and balancing dataset...")

    # Clean and create directory
    if FINAL_ROOT.exists():
        shutil.rmtree(FINAL_ROOT)
    FINAL_ROOT.mkdir(parents=True)

    # Collect ALL data
    all_tiles = []

    # Collect ORIGINAL tiles
    for img_path in TILED_IMAGES.glob("*.jpg"):
        parent_id = extract_parent_image_id(img_path.name)
        lbl_path = TILED_LABELS / f"{img_path.stem}.txt"

        is_positive = lbl_path.exists() and lbl_path.stat().st_size > 0

        all_tiles.append({
            'img_path': img_path,
```

```
            'lbl_path': lbl_path,
            'parent_id': parent_id,
            'is_positive': is_positive,
            'is_augmented': False,
            'has_label': is_positive
        })

    # Collect AUGMENTED tiles (all are positive)
    for img_path in AUG_IMAGES.glob("*.jpg"):
        parent_id = extract_parent_image_id(img_path.name)
        lbl_path = AUG_LABELS / f"{img_path.stem}.txt"

        if lbl_path.exists():  # Augmented tiles should all have labels
            all_tiles.append({
                'img_path': img_path,
                'lbl_path': lbl_path,
                'parent_id': parent_id,
                'is_positive': True,
                'is_augmented': True,
                'has_label': True
            })

    print(f" Total tiles collected: {len(all_tiles)}")

    # Group by parent image
    parent_to_tiles = {}
    for tile in all_tiles:
        parent_id = tile['parent_id']
        if parent_id not in parent_to_tiles:
            parent_to_tiles[parent_id] = []
        parent_to_tiles[parent_id].append(tile)

    print(f" Grouped into {len(parent_to_tiles)} parent images")
    # Categorize parent images
    parents_with_turtles = []
    parents_without_turtles = []
```

```
for parent_id, tiles in parent_to_tiles.items():
    has_turtle = any(tile['is_positive'] for tile in tiles)
    if has_turtle:
        parents_with_turtles.append(parent_id)
    else:
        parents_without_turtles.append(parent_id)

print(f"\n Parent image analysis:")
print(f"  Parents WITH turtles: {len(parents_with_turtles)}")
print(f"  Parents WITHOUT turtles: {len(parents_without_turtles)}")

# Split parent images (70/20/10)
random.shuffle(parents_with_turtles)
random.shuffle(parents_without_turtles)

# Split parents WITH turtles
n_turtle_train = int(0.7 * len(parents_with_turtles))
n_turtle_val = int(0.2 * len(parents_with_turtles))

train_turtle_parents = set(parents_with_turtles[:n_turtle_train])
val_turtle_parents = set(parents_with_turtles[n_turtle_train:n_turtle_train + n_turtle_val])
test_turtle_parents = set(parents_with_turtles[n_turtle_train + n_turtle_val:])

# Split parents WITHOUT turtles (for negative samples)
n_empty_train = int(0.7 * len(parents_without_turtles))
n_empty_val = int(0.2 * len(parents_without_turtles))

train_empty_parents = set(parents_without_turtles[:n_empty_train])
val_empty_parents = set(parents_without_turtles[n_empty_train:n_empty_train + n_empty_val])
test_empty_parents = set(parents_without_turtles[n_empty_train + n_empty_val:])

print(f"\n Parent image split:")
print(f"  Train: {len(train_turtle_parents)} turtle parents + {len(train_empty_parents)} empty parents")
print(f"  Val: {len(val_turtle_parents)} turtle parents + {len(val_empty_parents)} empty parents")
print(f"  Test: {len(test_turtle_parents)} turtle parents + {len(test_empty_parents)} empty parents")
```

```
# Build splits
train_tiles = []
val_tiles = []
test_tiles = []

for parent_id, tiles in parent_to_tiles.items():
    if parent_id in train_turtle_parents or parent_id in train_empty_parents:
        train_tiles.extend(tiles)
    elif parent_id in val_turtle_parents or parent_id in val_empty_parents:
        val_tiles.extend(tiles)
    elif parent_id in test_turtle_parents or parent_id in test_empty_parents:
        test_tiles.extend(tiles)

print(f"\n Initial tile counts:")
print(f"  Train: {len(train_tiles)} tiles")
print(f"  Val: {len(val_tiles)} tiles")
print(f"  Test: {len(test_tiles)} tiles")

# BALANCE each split
def balance_split(tiles, target_pos_ratio):
    """Balance a split to have target ratio of positive tiles."""
    positives = [t for t in tiles if t['is_positive']]
    negatives = [t for t in tiles if not t['is_positive']]

    print(f"    Before: {len(positives)} pos, {len(negatives)} neg")

    # Calculate how many negatives to keep
    if len(positives) > 0:
        desired_negatives = int(len(positives) * ((1 - target_pos_ratio) / target_pos_ratio))
        if desired_negatives < len(negatives):
            # Keep only some negatives
            random.shuffle(negatives)
            negatives = negatives[:desired_negatives]
        elif desired_negatives > len(negatives):
            # We don't have enough negatives - keep all we have
```

```
            print(f"    Warning: Not enough negatives for ideal ratio")

    balanced = positives + negatives
    random.shuffle(balanced)
    print(f"    After: {len(positives)} pos, {len(negatives)} neg")
    return balanced

print(f"\n Balancing splits (target: {target_pos_ratio*100:.0f}% positive)...")
print("  Train:")
train_tiles = balance_split(train_tiles, target_pos_ratio)
print("  Val:")
val_tiles = balance_split(val_tiles, target_pos_ratio)
print("  Test:")
test_tiles = balance_split(test_tiles, target_pos_ratio)

# Create directories
for split in ["train", "val", "test"]:
    (FINAL_ROOT / "images" / split).mkdir(parents=True, exist_ok=True)
    (FINAL_ROOT / "labels" / split).mkdir(parents=True, exist_ok=True)

# Copy files
def copy_split(tiles, split_name):
    count = 0
    for tile in tiles:
        img_path = tile['img_path']
        lbl_path = tile['lbl_path']

        # Copy image
        dest_img = FINAL_ROOT / "images" / split_name / img_path.name
        shutil.copy2(img_path, dest_img)
        # Copy or create label
        if tile['has_label'] and lbl_path.exists():
            dest_lbl = FINAL_ROOT / "labels" / split_name / f"{img_path.stem}.txt"
            shutil.copy2(lbl_path, dest_lbl)
        else:
            # Create empty label
```

```
                dest_lbl = FINAL_ROOT / "labels" / split_name / f"{img_path.stem}.txt"
                dest_lbl.touch()
            count += 1
        return count
    print("\nCopying files...")
    train_count = copy_split(train_tiles, "train")
    val_count = copy_split(val_tiles, "val")
    test_count = copy_split(test_tiles, "test")

    print(f"\nFinal tile distribution:")
    print(f"  Train: {train_count} tiles")
    print(f"  Val: {val_count} tiles")
    print(f"  Test: {test_count} tiles")
    print(f"  Total: {train_count + val_count + test_count} tiles")

    # Create data.yaml
    yaml_content = f"""# turtle_dataset.yaml
path: {FINAL_ROOT.as_posix()}
train: images/train
val: images/val
test: images/test
# Number of classes
nc: 1
# Class names
names: ['turtle']
# Dataset statistics
# Total tiles: {train_count + val_count + test_count}
# Target positive ratio: {target_pos_ratio*100:.0f}%
# Parent images: {len(parent_to_tiles)} total
"""
    with open(FINAL_ROOT / "data.yaml", "w") as f:
        f.write(yaml_content)

    print(f"\ndata.yaml created at: {FINAL_ROOT / 'data.yaml'}")
    return train_tiles, val_tiles, test_tiles
# Build the balanced dataset (30% positive, 70% negative)
```

```
train_tiles, val_tiles, test_tiles = build_balanced_dataset(target_pos_ratio=0.3)

#Create YOLO files
from pathlib import Path

root = Path(r"C:\turtle_detection\turtle_2025\data\yolo_dataset_balanced")
for split in ["train", "val", "test"]:
    assert (root / "images" / split).exists()
    assert (root / "labels" / split).exists()
import sys
import torch
import ultralytics
import torch
from ultralytics import YOLO
from ultralytics.nn.tasks import DetectionModel
with torch.serialization.safe_globals([DetectionModel]):
    model = YOLO("yolov8n.pt")
data_path = r"C:\turtle_detection\turtle_2025\data\\yolo_dataset_balanced\data.yaml"

print(f"Using config: {data_path}")

model.train(
    data=data_path,

    # Training duration
    epochs=300,
    patience=30,

    # Hardware
    imgsz=640,
    batch=16,
    device=0,
    workers=4,



    # Optimization
```

```
    lr0=0.01,                  # Initial learning rate
    lrf=0.1,                   # Final learning rate factor
    momentum=0.937,            # SGD momentum
    weight_decay=0.0005,       # Optimizer weight decay
    warmup_epochs=5.0,         # Warmup epochs
    warmup_momentum=0.8,       # Warmup initial momentum
    warmup_bias_lr=0.1,        # Warmup bias learning rate

    # Cosine LR scheduler
    cos_lr=True,               # ENABLE cosine LR

    # Loss weights
    box=7.5,                   # Box loss gain
    cls=0.5,                   # Class loss gain
    dfl=1.5,                   # Distribution Focal Loss gain

    # Validation
    val=True,                  # Validate during training
    save=True,                 # Save checkpoints
    save_period=-1,            # Save every epoch (-1 for best only)
    plots=True,
    visualize=True,

    # Reproducibility
    deterministic=True,
    seed=42,

    project="turtle_yolo",
    name="yolov8n_turtle_v1",

    # Logging
    single_cls=True,
    verbose=True,
    save_json=True,
    pretrained=True,
    freeze=0,
```

```
)
#Validation
import cv2
import numpy as np
from pathlib import Path
import torch
from ultralytics import YOLO
from ultralytics.nn.tasks import DetectionModel
import matplotlib.pyplot as plt

# Load model
model_path = r"C:\turtle_detection\turtle_2025\turtle_yolo\yolov8n_turtle_v13\weights\best.pt"

with torch.serialization.safe_globals([DetectionModel]):
    model = YOLO(model_path)

print("Testing on validation images...")

# Get validation images
val_images_dir = Path(r"C:\turtle_detection\turtle_2025\data\yolo_dataset\images\val")
val_images = list(val_images_dir.glob("*.jpg"))[:5]  # Test first 5

if val_images:
    for img_path in val_images:
        print(f"\nProcessing: {img_path.name}")

        # Run inference
        results = model(img_path, conf=0.25, device=0)

        # Show results
        if results[0].boxes is not None:
            num_detections = len(results[0].boxes)
            avg_confidence = results[0].boxes.conf.mean().item()
            print(f"  Detections: {num_detections}")
            print(f"  Avg confidence: {avg_confidence:.3f}")
```

```
            # Get the annotated image
            annotated_img = results[0].plot()

            # Convert from BGR (OpenCV) to RGB (Matplotlib)
            annotated_img_rgb = cv2.cvtColor(annotated_img, cv2.COLOR_BGR2RGB)

            # Create plot
            plt.figure(figsize=(12, 8))
            plt.imshow(annotated_img_rgb)
            plt.title(f"Detection Results: {img_path.name}\nDetections: {num_detections}, Avg Confidence: {avg_confidence:.3f}")
            plt.axis('off')
            plt.show()

        else:
            print("  No turtles detected")

            # Show original image if no detections
            img = cv2.imread(str(img_path))
            img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

            plt.figure(figsize=(12, 8))
            plt.imshow(img_rgb)
            plt.title(f"No detections: {img_path.name}")
            plt.axis('off')
            plt.show()

else:
    print(f"No validation images found in {val_images_dir}")

#Inference Test
from sahi import AutoDetectionModel
from sahi.predict import get_sliced_prediction
from sahi.utils.cv import read_image, visualize_object_predictions
from pathlib import Path
import cv2
```

```
import numpy as np
import matplotlib.pyplot as plt
import torch

def inference_with_sahi(image_path, model_path, slice_size=640, overlap_ratio=0.2):
    """
    Use SAHI for sliced inference with overlap handling
    """
    # Load image
    image = read_image(image_path)
    print(f"Image shape: {image.shape}")

    # Create SAHI detection model
    detection_model = AutoDetectionModel.from_pretrained(
        model_type='yolov8',
        model_path=model_path,
        confidence_threshold=0.25,
        device='cuda:0' if torch.cuda.is_available() else 'cpu'
    )

    # Get sliced predictions
    result = get_sliced_prediction(
        image,
        detection_model,
        slice_height=slice_size,
        slice_width=slice_size,
        overlap_height_ratio=overlap_ratio,
        overlap_width_ratio=overlap_ratio,
        perform_standard_pred=False,
        postprocess_type="NMS",
        postprocess_match_metric="IOU",
        postprocess_match_threshold=0.5,
        verbose=1  # Set to 1 for more info
    )

    return result, image
```

```
def visualize_sahi_detections(image, sahi_result):
    """Visualize SAHI detections"""
    # Check if there are detections
    if not sahi_result.object_prediction_list:
        print("No detections found!")
        # Show original image
        plt.figure(figsize=(10, 10))
        plt.imshow(image)
        plt.title('No detections found')
        plt.axis('off')
        plt.show()
        return

    print(f"Number of detections: {len(sahi_result.object_prediction_list)}")

    # Try different visualization methods
    try:
        result_image = visualize_object_predictions(
            image=image,
            object_prediction_list=sahi_result.object_prediction_list,
            rect_th=2,
            text_size=0.8,
            text_th=2,
            color=(255, 0, 0)
        )

        if result_image is None:
            print("SAHI visualization returned None, using manual visualization")
            raise ValueError("SAHI visualization failed")

        # Convert to RGB for matplotlib
        result_image_rgb = cv2.cvtColor(result_image, cv2.COLOR_BGR2RGB)

    except Exception as e:
        print(f"SAHI visualization failed: {e}")
```

```
    # Method 2: Manual visualization
    result_image_rgb = image.copy()

    for pred in sahi_result.object_prediction_list:
        bbox = pred.bbox.to_xyxy()
        # Convert to integers
        x1, y1, x2, y2 = map(int, bbox)
        confidence = pred.score.value

        # Draw rectangle
        cv2.rectangle(result_image_rgb, (x1, y1), (x2, y2), (255, 0, 0), 3)

        # Add confidence text
        label = f"{pred.category.name}: {confidence:.2f}"
        cv2.putText(result_image_rgb, label, (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)

# Display
plt.figure(figsize=(12, 8))
plt.imshow(result_image_rgb)
plt.title(f'SAHI Detections: {len(sahi_result.object_prediction_list)} turtles')
plt.axis('off')
plt.tight_layout()
plt.savefig('sahi_detections.jpg', dpi=300, bbox_inches='tight')
plt.show()

# Print detailed results
print("\n=== Detailed Detections ===")
for i, pred in enumerate(sahi_result.object_prediction_list):
    bbox = pred.bbox.to_xyxy()
    print(f"Detection {i+1}:")
    print(f"  BBox: [{bbox[0]:.1f}, {bbox[1]:.1f}, {bbox[2]:.1f}, {bbox[3]:.1f}]")
    print(f"  Confidence: {pred.score.value:.3f}")
    print(f"  Category: {pred.category.name}")
```

```
image_path =
r"C:\turtle_detection\turtle_2025\data\yolo_dataset_balanced\images\test\DJI_0025_tile_0008_aug0.j
pg"

model_path = 'turtle_yolo/yolov8n_turtle_v14/weights/best.pt'

# Check if files exist

print(f"Image path exists: {Path(image_path).exists()}")

print(f"Model path exists: {Path(model_path).exists()}")

# Run inference

result, image = inference_with_sahi(image_path, model_path)

# Visualize results

visualize_sahi_detections(image, result)
```