

# Small UAS Landing Site Detection with Deep Learning based Computer Vision

Jadon Xu<sup>\*1</sup>, Can Vu<sup>†1</sup>, Abenezer Taye<sup>‡</sup>, Amin Tabrizian<sup>§</sup> and Peng Wei<sup>¶</sup>  
*George Washington University, Washington, DC, 20052*

Small unmanned aerial systems (UAS) have shown promises in commercial package delivery services. Numerous studies have shown ArUco markers to be an effective and robust method to enhance the identification of landing sites during UAS descent. Practical deployment of these markers at warehouse landing sites (such as a Walmart parking lot), where the UAS can land and load packages for delivery, is feasible. However, it seems an unlikely assumption that a customer or end-user would be able to set up an ArUco marker landing configuration at their home delivery site, such as a front yard, backyard, or on top of the apartment building with the correct ordering and proper dimensions. It is much more user-friendly to require only a generic landing pad that the UAS can recognize using deep learning techniques; Thus, this paper investigates the feasibility of using deep learning based computer vision models to recognize the landing pad without relying on ArUco markers. To that end, we describe a custom data collection methodology using flight videos featuring a landing pad, along with the dataset annotation procedure in Roboflow. This annotated dataset is used to train multiple YOLOv8 object detection models, each initialized with different weights, to accurately detect the landing site. These models are integrated into a proposed Obstacle and Landing Site Awareness System (OLSAS) designed to support safe and autonomous UAS package delivery landings. Finally, we compare the detection performance and reliability of the deep learning-based approach against an ArUco marker detection algorithm to evaluate their effectiveness in identifying landing sites.

## I. Introduction

The emergence of small unmanned aerial systems (UAS) for package delivery, enabled by advancements in technology and successful flight demonstrations, is showing great promises. The Federal Aviation Administration (FAA) recently approved companies like Zipline and Wing Aviation to conduct commercial UAS operations in the Dallas-Fort Worth (DFW) region with beyond-visual line of sight (BVLOS) mode [1–3]. This significant milestone is groundbreaking in the regulatory sector, encouraging the integration of routine drone deliveries in low-altitude airspace. It enables the delivery drone operators and the Unmanned Aircraft System Traffic Management (UTM) service providers to work together and maintain safe operations during package deliveries [2]. This development marks a crucial step toward achieving regular BVLOS operations in shared airspace, bolstered by advancements in air traffic management technology.

Wing provides direct delivery of lightweight packages from businesses to consumers. As a subsidiary of Alphabet, the company has developed a fleet of autonomous drones designed to efficiently transport small items to meet on-demand needs [4]. By October 2024, Wing has provided services to 26 cities and towns in the DFW metroplex in Texas. In collaboration with Walmart, Wing covers 60,000 households, with plans to serve the entire 8 million population in the area [5]. The average flight time is 3 minutes and 24 seconds from store to customer [6]. Meanwhile, Zipline is widely recognized for its pioneering efforts in medical supply delivery, particularly in African countries like Rwanda and Ghana. Operating one of the largest autonomous UAS delivery networks globally, Zipline also provides services in the United States, Japan, and other nations. Its system supports not only healthcare packages, but also retail and logistics by integrating with existing e-commerce platforms and parcel carriers [3].

As companies scale up their BVLOS UAS delivery operations, ensuring the safety and accuracy of UAS landings is critical. Vision-based landing systems are essential for achieving reliable autonomous deliveries, particularly in

---

<sup>\*</sup>Research Intern, George Washington University.

<sup>†</sup>Ph.D. Student, Department of Mechanical and Aerospace Engineering, George Washington University, AIAA Student Member.

<sup>‡</sup>Ph.D. Student, Department of Mechanical and Aerospace Engineering, George Washington University, AIAA Student Member.

<sup>§</sup>Ph.D. Student, Department of Computer Science, George Washington University, AIAA Student Member.

<sup>¶</sup>Associate Professor, Department of Mechanical and Aerospace Engineering, George Washington University, AIAA Associate Fellow.

<sup>1</sup>These authors contributed equally to this work.

complex environments where GPS may be unreliable. For enhanced autonomy during off-nominal landings, a UAS must recognize the landing site and its nearby obstacles such as people, pets, cars, and/or bicycles. [7].

Establishing landing sites based on artificial markers is costly and potentially difficult for the end-user using package delivery services straight to their front door. It is impractical to expect the customer to configure a set of ArUco markers with the right dimensions and orientation relative to the UAS performing delivery landings. With this motivation, this paper presents a custom landing pad dataset collected from real-world UAS flight tests for training a YOLOv8 [8] deep learning object detection model. We describe the data collection, annotation using Roboflow, preprocessing, and augmentation. The trained YOLOv8 model is integrated into a proposed safety awareness system and evaluated on real flight data. We compare the performance of the YOLOv8-based landing pad detector against an ArUco marker-based detection method in terms of detection consistency, bounding box accuracy, and processing speed. The results demonstrate the advantages of using deep learning for reliable and efficient landing site detection in UAS package delivery operations.

## II. Related Works

Previous works have investigated various methods of landing zone identification during the landing phase of UAS operations. Rasool et al. [9] proposed a machine learning solution to identify helipads, heliports, and other landing sites, from aerial images using CNNs. The authors compiled a dataset from various databases with satellite images from Google Earth and used the data to train different state-of-the-art CNN models for landing zone identification. The trained CNN was used to identify the presence of a landing pad and a sliding window approach was used on collected samples of the area to determine its location in the larger image. This approach resulted in an accuracy of 76% when tested on images from the Los Angeles (LA) region, although it was noted that more data would be required for widespread deployment in other areas. Yang, Sherer, and Zell [10] proposed an onboard monocular vision system that detected a landing pad consisting of an 'H' surrounded by a circle. The monocular camera captures images on the UAS and the 6 degrees of freedom (DOF) of the UAS relative to landing pad is determined using projective geometry. The authors reported that their algorithms were able to process up to 60 frames per second (FPS) on the onboard computer.

Deep learning computer vision algorithms such as RetinaNet [11] and the YOLO [12, 13] family of deep learning models are popular methods of object identification in autonomous sUAS applications due to their robustness and high performance. Yang et al. [14] developed a deep learning based computer vision model using RetinaNet for pedestrian detection from a small UAS during dynamic aerial missions. Their deep learning algorithm demonstrated high accuracy for detecting pedestrians and vehicles in the scene, despite the relatively small size of the objects from a bird's eye view. Zhao et al. [7] extended this work by comparing the RetinaNet and YOLOv5 [15] deep learning models for real-time obstacle detection, as well as implementing a safety checking algorithm using the obstacle's bounding box and landing pad coordinates. It was shown that YOLOv5 and RetinaNet yielded similar detection accuracy; however, YOLOv5 was selected for its higher inference speed as the object detector for the work. The investigation by Zhao et al. relied on the use of ArUco markers to precisely define the landing site for the UAS and to provide the corner coordinates of the landing zone for the safety checking algorithm. The reliance on these markers led to more complexity in the software to generate and detect the markers.

Another study by Wubben et al. [16] also proposed a solution for autonomous landing based on the use of ArUco markers. Unlike the previous work, Wubben et al. used a large ArUco marker to localize the UAS over the target landing site. Once the UAS detected the initial marker, it would descend until detecting the next largest marker, which would be used as the next reference point until the landing procedure was complete. The authors reported that accurate landing during testing was achieved without the use of the Global Positioning System (GPS) with a relatively small landing time; however, the tests were performed in clear weather conditions and more work was needed to land the UAS in less favorable weather. A study by Del Pizzo et al. [17] investigated a vision-based navigation system for a UAS landing procedure using a mounted camera to determine the UAS position and attitude with respect to a well-defined landing pattern. The landing pattern, different from the ArUco artificial markers, was composed of a metallic platform with several circular targets and a Circular Hough Transform Function from the OpenCV libraries was used to recognize the targets and extract the coordinates. Their method achieved high precisions in the position; however, the attitude angle precision was not comparable compared with commercial navigation systems such as GPS.

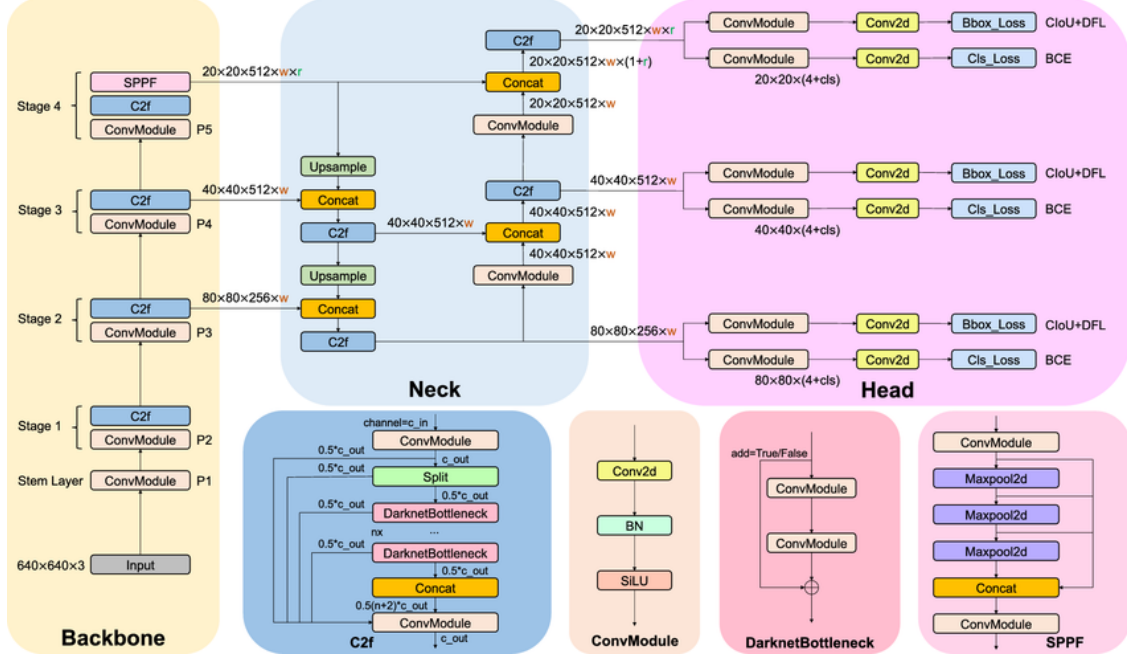


Fig. 1 A system block diagram of the backbone, neck, and head structures of the YOLOv8 object detection model [20].

### III. Obstacle and Landing Pad Detection System

#### A. Deep Learning Object Detection Model Selection

The deep learning based object detection model selected for this work was the Ultralytics YOLOv8 model [8]. The YOLOv8 model was first released in 2023, offering cutting-edge speed and accuracy in object detection tasks at the time. An improvement on the YOLOv5 model, YOLOv8 introduces key innovations and enhancements to performance, efficiency, and ease of use. The base architecture of YOLOv8 (like YOLOv5) consists of a backbone, neck, and head structure. The enhanced CSPDarknet backbone [18] convolutional network extracts and processes a set of meaningful features from the input images. The neck uses the Path Aggregation Network [19] (PANet) structure, building on the Feature Pyramid Network, to perform feature fusion and contextual information integration on the extracted features from the backbone. This boosts the performance on small object detection. The head, featuring an anchor-free design, is the final layer outputting the bounding boxes and confidence scores. The full architecture of YOLOv8 is given in Figure 1. YOLOv8 optimizes a composite loss function that is the weighted sum of three loss terms and is given in Equation 1,

$$L_{\text{total}} = \lambda_{\text{box}} \cdot L_{\text{box}} + \lambda_{\text{cls}} \cdot L_{\text{cls}} + \lambda_{\text{dfl}} \cdot L_{\text{dfl}} \quad (1)$$

where  $L_{\text{box}}$  is the bounding box regression loss,  $L_{\text{cls}}$  is the binary cross-entropy classification loss, and  $L_{\text{dfl}}$  is the distribution focal loss (dfl). The weighting coefficients  $\lambda_{\text{box}}$ ,  $\lambda_{\text{cls}}$ , and  $\lambda_{\text{dfl}}$  control the relative importance of each loss component. These values are tuned to balance localization accuracy, classification performance, and boundary precision.

The models in this work are trained for the obstacle and landing pad detection tasks in the UAS package delivery landing environment. The YOLOv8 models are trained for obstacle detection using the VisDrone [21] dataset. For landing pad detection, the YOLOv8 models are trained on a custom landing pad dataset, which was collected from real-world flight data, with each frame manually annotated with the appropriate label and bounding box for the landing pad object. Additional details on the training process for our models is described in Section V.B.

#### B. VisDrone Dataset for Aerial-View Object Detection

The VisDrone dataset is a comprehensive large-scale benchmark designed for object detection, tracking, and scene understanding in aerial imagery. It was developed by the AISKYEYE team [22] at the Lab of Machine Learning and Data Mining in Tianjin University, China. The dataset consists of a total of 261,908 video frames and 10,209 static



(a) Hoodman Drone Launch/Landing Pad 5 ft.



(b) DJI Mini 3 (DJI RC-N1).

**Fig. 2 The hardware used to capture the images of our dataset.**

images, captured from a variety of UAS platforms and camera sensors. These images were collected over a wide geographic range, including urban, suburban, and rural areas across multiple cities in China, providing rich diversity in spatial context and scene composition.

VisDrone captures a broad spectrum of real-world conditions, encompassing varying altitudes, camera angles (e.g., nadir and oblique views), resolutions, lighting conditions (daytime, dusk, and low-light), weather (clear, cloudy, and rainy), and object densities. This variability makes it particularly suitable for evaluating the robustness and generalization capabilities of vision algorithms under realistic and challenging aerial scenarios where UAS safety is a critical concern. All frames in the dataset are manually annotated with detailed bounding box labels for over 2.6 million object instances. These annotations span 10 object categories, including pedestrians, bicycles, cars, vans, trucks, buses, motorcycles, tricycles, awning-tricycles, and others. Each annotation includes class labels and visibility flags indicating partial or full occlusion and truncation, enabling more precise training and evaluation of detection models under occluded and cluttered conditions.

### C. Custom Landing Pad Detection Dataset

For the landing pad detection, we collected and annotated a custom landing pad dataset consisting of a variety of aerial-view landing pad images taken from real-world UAS flights. The custom landing pad dataset was formed by recording multiple six minute videos simulating the descent of a UAS onto a landing pad with the aim of capturing viewpoints of varying altitudes. The Landing pad is a Hoodman Drone Landing Pad with a five-foot diameter shown in Figure 2a. This landing pad serves as the landing target to localize the UAS in a simulated package delivery scenario. A DJI Mini 3 shown in Figure 2b is used to capture the videos at  $1920 \times 1080p$  resolution for processing. During data collection, the UAS was flown at various altitudes of 0 to 20 meters, 20 to 30 meters, and more than 60 meters. Examples of the landing pad from the point-of-view of the UAS is shown in Figure 6. The image was captured at a high distance above the landing zone, so the landing pad is extremely small. The orientation of the camera and position of the UAS is changed throughout each video to ensure variability in sequential frames.

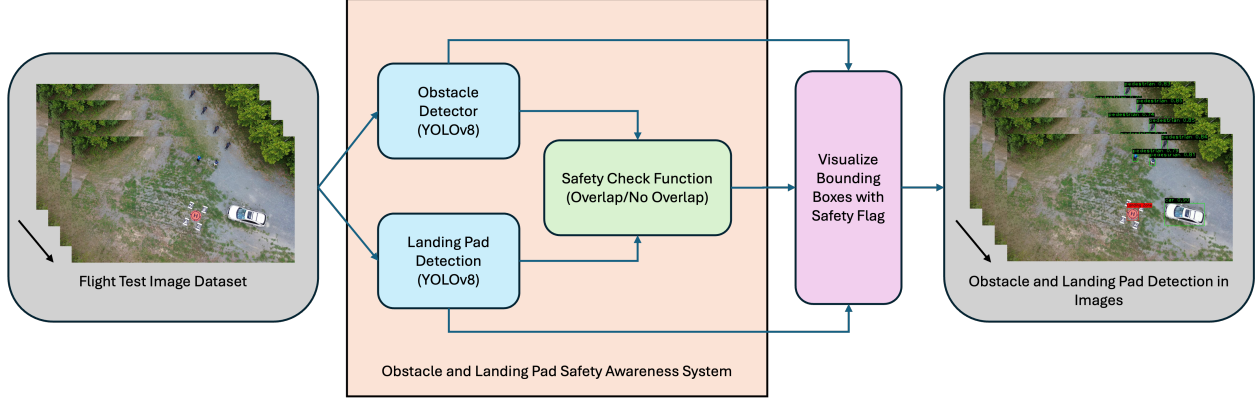
### D. System for Object and Landing Pad Detection in the Operational Landing Environment

Previous research has demonstrated that ArUco fiducial markers offer a promising solution for landing site identification in vision-based navigation systems. However, several limitations hinder their practical adoption. Detection reliability can be inconsistent, especially under challenging conditions such as variable lighting, occlusion, or low-resolution imagery. Moreover, deploying ArUco markers in real-world environments requires precise placement and orientation to ensure successful recognition by OpenCV-based detection algorithms, an approach that can be impractical or labor-intensive in dynamic or unstructured real-world settings.

To address these limitations, we propose the Obstacle and Landing Pad Safety Awareness System (OLSAS) that leverages a separate YOLOv8 object detection network to identify a generic landing pad without relying on fiducial markers. The OLSAS contains two independently trained YOLOv8 models. The first model detects obstacles in the landing environment that may pose a safety risk such as pedestrians, pets, or vehicles. The second model is dedicated solely to the detection of the landing pad. The landing pad detection model can be initialized to use either the ArUco marker detection algorithm or a deep learning based object detector.

Bounding box predictions from both detectors are passed to a safety check algorithm, adapted from [7], which





**Fig. 3 Proposed system for obstacle and landing pad safety awareness in a UAS package delivery operation.** The images captured by the UAS camera are fed to two YOLOv8 networks that independently detect the landing pad and surrounding obstacles in the scene. The bounding box coordinates are passed to a safety check function [7] that raises a flag when any obstacle’s bounding boxes overlap with the landing pad’s bounding box. If this happens, a flag is raised signaling an unsafe landing condition. The bounding box coordinates and flag are passed to a function that visualizes the detections. The Landing Pad Detection block in this system utilizes either the ArUco-based detection algorithms or the landing pad trained YOLOv8 model.

determines whether any detected obstacles overlap with the landing pad. If an overlap is detected, indicating a potential hazard to the UAS or people on the ground, the system raises a safety flag to alert the UAS of the unsafe condition. The final stage of the pipeline visualizes the YOLOv8 predictions along with the safety status for system monitoring and evaluation. The end-to-end architecture of the OLSAS is illustrated in Figure 3.

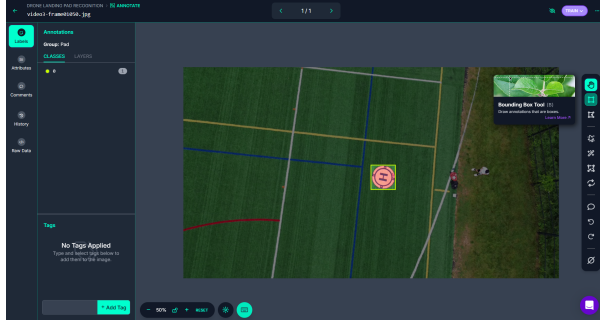
#### IV. Custom Landing Pad Dataset Annotation Process

The OpenCV Python library was used to extract every 10<sup>th</sup> frame from flight test videos to reduce dataset size and limit redundancy between consecutive frames. This frame sampling process resulted in a total of 3,392 images, a subset of which was used to construct the custom landing pad dataset. The landing pad detection model was trained on this manually annotated dataset, which was subsequently pre-processed and augmented using the Roboflow platform [23]. Roboflow was chosen as the annotation platform due to its comprehensive suite of tools for computer vision workflows, including data collection, annotation, model training, and deployment. For this study, 600 images (200 from each altitude range) were randomly selected to form the training and validation dataset for a preliminary YOLOv8 landing pad detection model. These images were uploaded to a public Roboflow project as a batch of JPEG files. Annotation was performed manually in Roboflow’s “Annotate” interface. Each image was labeled using a bounding box to encompass the landing pad object, with the class label assigned as “0”. Figure 4a illustrates the annotation interface and tools used.

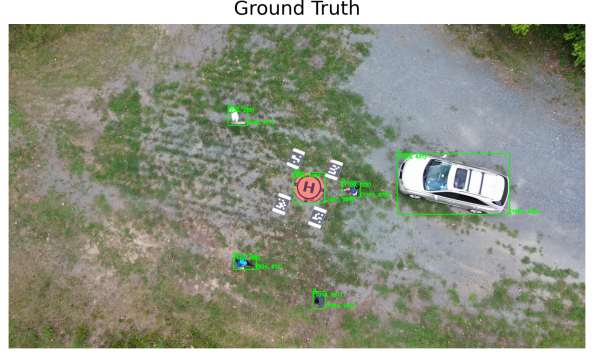
Following annotation, a new version of the dataset was generated within Roboflow. This version included the original 600 annotated frames, along with pre-processing and augmentation techniques to enhance dataset variability and model robustness. During pre-processing, all images were auto-oriented—meaning EXIF metadata was stripped—to ensure consistent alignment between the annotated and exported images. Each image was then resized to  $640 \times 640$  pixels to standardize input dimensions and reduce computational load. The preprocessing settings are summarized in Figure 5a.

To improve the models’ generalization to varied visual conditions, data augmentation was applied. For each original image in the training set, three augmented images were generated. These augmentations included random vertical and horizontal flipping, and random rotations between  $-15^\circ$  and  $+15^\circ$ . This variability helps the model detect the landing pad from multiple orientations, as encountered in real-world UAS landing scenarios. The specific augmentation parameters used are shown in Figure 5b. Additional pre-processing and augmentation options are available through Roboflow’s interface.

After pre-processing and augmentation, the dataset is split with 90% assigned to training set and the remaining 10% to the validation set. The final dataset was exported using the Roboflow API, which allows direct integration with Python notebooks via the Roboflow Python library. To export the dataset, the user selects “Download Dataset,” chooses an appropriate export format, and clicks “Show download code” to reveal the API snippet. The download code used in



(a) An example dataset frame in the Roboflow annotation window. The bounding box is tightly drawn around the landing pad and labeled as class “0” from the object class list.



(b) Annotated frame from a real-world flight test video showing a generic landing pad surrounded by pedestrians and a car. The scene simulates a realistic UAS landing environment.

**Fig. 4** An example frame from our flight test dataset and annotation tool used to create the landing pad detection dataset. The landing pad dataset contains examples that match the set up of the landing pad in our flight tests. This is the expected configuration of the environment for our UAS package delivery use-case. (a) Annotation interface in Roboflow with an example image from the custom landing pad dataset; (b) Annotated frame from our real-world UAS flight test landing scenario.

this project is shown in Figure 5c. Alternatively, the dataset can be downloaded locally as a ZIP archive in various formats such as JSON, XML, or TXT.

## V. Experimental Setup

### A. UAS Flight Test Videos

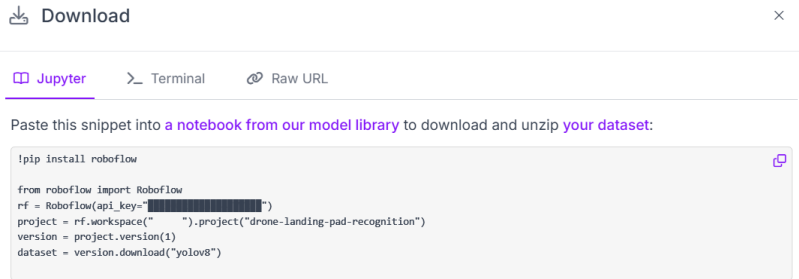
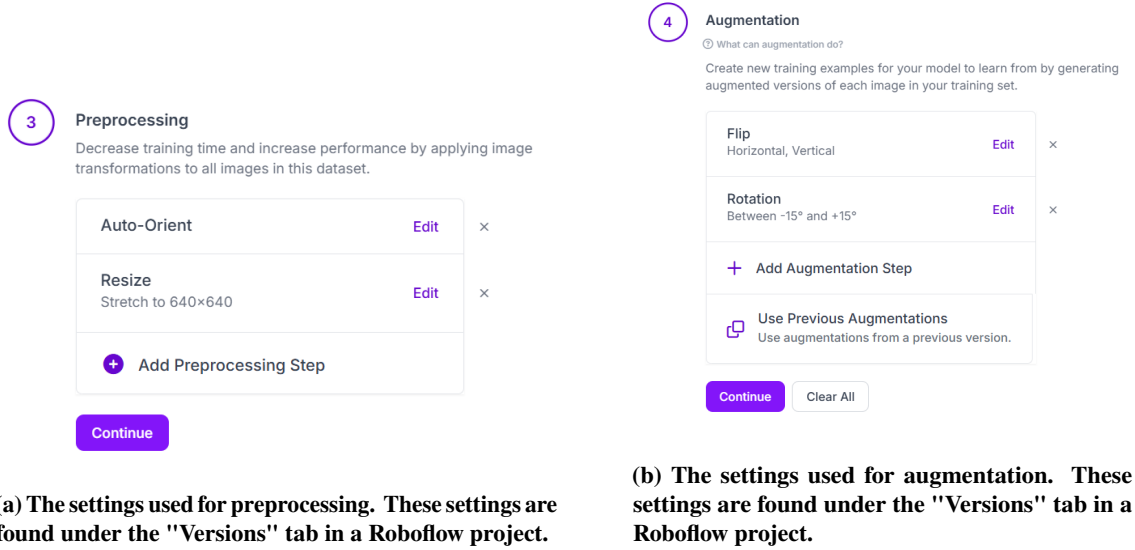
The flight test video data used for evaluating the obstacle and landing pad awareness system was collected with a DJI Mini 3 (DJI RC-N1) drone at  $1920 \times 1080p$  resolution with a frame rate of 30 frame per second (FPS). The flight test data consists of real-world scenarios replicating an autonomous UAS package delivery landing operation in a suburban neighborhood environment with four pedestrian and one car objects around the landing pad. In these scenarios, a UAS starts at a high altitude and descends toward a landing site marked by ArUco fiducial markers and an orange circular landing pad. The environment obstacles move around, or cross into, the designated landing site to simulate potential behaviors of pedestrians and cars in a suburban neighborhood setting that may result in a safety issue. An example frame from our annotated test dataset is shown in Figure 4b. The flight test video used for evaluation is 20 seconds long and the corresponding image dataset contains 600 frames.

### B. Training YOLOv8 on the Custom Dataset

The YOLOv8 model was trained using the Ultralytics framework, which provides a streamlined API for the efficient development and deployment of object detection models. All training and evaluation experiments are conducted on a workstation running Ubuntu 20.04.1 LTS, equipped with an AMD Ryzen Threadripper Pro 3975WX 32-core CPU (2.2 GHz) and two NVIDIA GeForce RTX 3090 GPUs.

Three YOLOv8 models were trained for the landing pad recognition task, each using a different weight initialization strategy to evaluate the impact of pre-training on detection performance. The first model was trained from scratch using the custom landing pad dataset, with randomly initialized weights. The second model was initialized with weights pre-trained on the COCO dataset [24] and then fine-tuned on the same custom dataset. The third model is pre-trained on COCO and further fine-tuned on the VisDrone dataset before final training on the custom dataset for landing pad detection.

All three YOLOv8 models were trained on the custom landing pad dataset using the same hyperparameter configuration to ensure a fair comparison across initialization strategies. Each model was trained for 150 epochs using



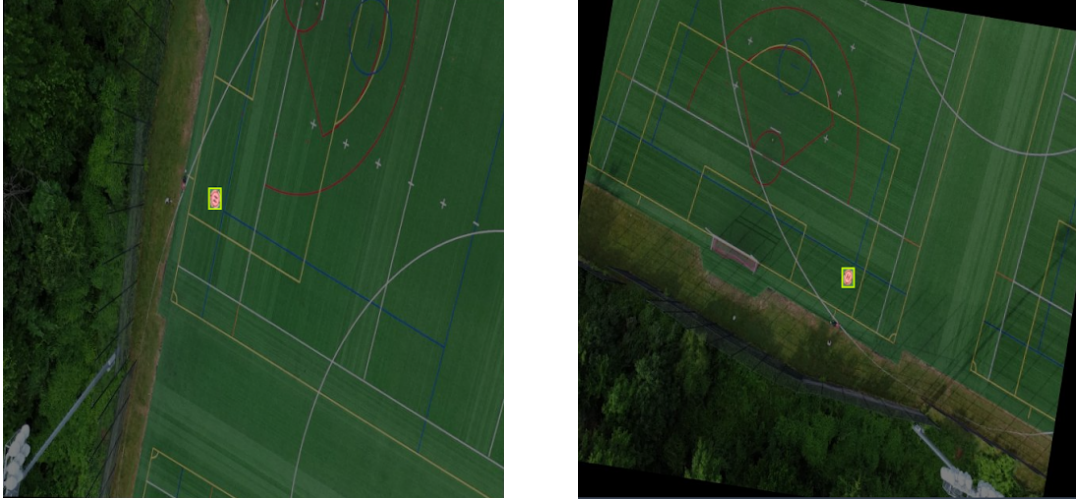
**(c) The export code used to access the landing pad dataset within a Python notebook.**

**Fig. 5 Roboflow options for data pre-processing, data augmentation, and export. Roboflow provides a variety of options to introduce variability and model compatibility for the annotated dataset such as image transformations, filtering, and resizing.**

the AdamW optimizer with a learning rate of 0.001 and cosine learning rate scheduling. The learning rate factor was 0.01. A batch size of 16 was used, with input images resized to  $720 \times 1280$  pixels. Early stopping was enabled with a patience of 15 epochs to ensure the models do not overfit during training, and data augmentation was applied using RandAugment, which included horizontal flipping, scaling, translation, HSV color jitter, and random erasing. For the two pre-trained models, the first 10 layers were frozen during early training. All other training settings were held constant to isolate the impact of model initialization. This includes the loss weights (box = 7.5, cls = 0.5, dfl = 1.5), input dimensions ( $720 \times 1280$ ), and augmentation strategies such as horizontal flipping (50%), scaling ( $\pm 50\%$ ), translation ( $\pm 10\%$ ), HSV augmentation (hsv\_h = 0.015, hsv\_s = 0.7, hsv\_v = 0.4), mixup (0.1), and erasing (0.1).

## VI. Results and Discussion

The results of the three YOLOv8 training experiments on the custom landing pad dataset are shown in Figure 7. In all cases, both training and validation losses decreased consistently over the course of training, while the mean average precision (mAP) either increased steadily or remained consistently high. The mAP@0.5 metric remained close to 1.0 throughout training for all models, indicating strong detection performance with lenient localization thresholds. In contrast, the mAP@0.5:0.95 metric showed progressive improvement over the epochs, reflecting enhanced localization precision. These trends demonstrate that all three models successfully learned the landing pad detection task and continued to improve in performance over time, indicating that the YOLOv8 object detection model is a promising candidate to use in the OLSAS for ensuring safe autonomous UAS package delivery during the landing phase.



**Fig. 6** An example of pre-processing on an image of a high-altitude view of the landing pad with the bounding box annotation. The image on the right is formed by rotating the image on the left by  $15^\circ$ .

A comparison of the three models in Figure 7 highlights the effect of weight initialization and pretraining on model performance. The model trained from scratch (Figure 7a) required more epochs to converge and exhibited higher initial losses, though it ultimately reached strong accuracy metrics in both mAP@50 and mAP@50-95. The model initialized with the COCO pre-trained weights (Figure 7b) demonstrated faster convergence, lower early-stage validation losses, and more rapid improvement in both mAP@0.5 and mAP@0.5:0.95. The best overall performance was observed in the pre-trained model that was first fine-tuned on the VisDrone dataset before training on the custom landing pad data (Figure 7c). Although all three models achieve similar final mAP@50-95 values, this model achieved the most stable loss curves and a marginally higher mAP@0.5:0.95, suggesting that domain-specific pretraining on aerial imagery can enhance detection precision in this use case. These results indicate an advantage to adapting, as opposed to training from scratch, pre-trained object detection models to specialized tasks such as landing pad detection in real-world UAS environments.

For the remainder of this discussion, we focus on the YOLOv8 model that was pre-trained on COCO, fine-tuned on the VisDrone dataset, and subsequently trained on our custom landing pad dataset. In the final OLSAS configuration described in 3, the obstacle detection model operates with a confidence threshold of 0.25 and an Intersection over Union (IoU) threshold of 0.50. These are standard values that provide a balanced trade-off between precision and recall for general object detection. In contrast, the landing pad detector is configured with a lower confidence threshold of 0.15 and an IoU threshold of 0.30. These relaxed thresholds help ensure that the landing pad is detected as early and reliably as possible, particularly under challenging or imperfect visual conditions, such as occlusions, high altitudes, or poor lighting. Reliable detection of the landing pad is critical for maintaining operational safety during descent, as missed detections may result in unsafe landing attempts near pedestrians or property. The system was evaluated on our real-world flight test dataset described in Section V.A.

Figures 8a, 8c, 8e, and 8g show frames collected after processing the flight test video through the OLSAS using ArUco marker-based landing pad detection. While ArUco markers were effective in identifying the approximate boundaries of the landing site, the detection performance was inconsistent, possibly due to the relative motion of the camera to the ArUco markers, which may distort the visual features necessary for successful recognition. In Figure 8a, the ArUco marker detection fails entirely, while only one of the markers is detected in Figure 8c. This inconsistent detection behavior is undesirable, as it can prevent the safety algorithm from reliably identifying unsafe events, such as when obstacles enter the landing zone. Additionally, to ensure aerial visibility, the ArUco markers must be placed with substantial spacing around the pad, effectively expanding the designated landing area. This may result in false unsafe-condition flags, even when no actual obstacles are present within the true landing pad.

Figures 8b, 8d, 8f, and 8h are frames collected after processing the flight test through the OLSAS with the YOLOv8-based landing pad detector enabled. The YOLOv8 model demonstrated more consistent and stable bounding box detections of the Hoodman landing pad across varying altitudes and environmental conditions. This is seen in

the right-hand column of Figure 8, where the YOLOv8-based detector reliably identifies the landing pad among the various obstacles found around the landing environment. Additionally, the YOLOv8 detector produces tighter and more centered bounding boxes compared to the ArUco-based detector shown in the left-hand column, which can fail to localize the pad when the ArUco markers are blurred, partially obstructed, or angled away from the camera. The processing speed of OLSAS using ArUco detection averages approximately 0.185 seconds per frame.

The results demonstrate that the YOLOv8-based landing pad detector, particularly when fine-tuned on domain-specific aerial imagery, offers an improvement in detection robustness and precision over the ArUco marker-based methods. The YOLOv8 model’s ability to consistently localize the landing pad under varying flight conditions and environmental complexities enhances the reliability of the OLSAS system. These improvements are essential for supporting safe, autonomous UAS landings in suburban environments where real-world variability poses challenges to traditional vision-based methods. The proposed OLSAS framework integrated with a deep learning based object detection model presents a promising step toward more adaptable and resilient autonomous landing systems for future UAS package delivery operations. The processing speed using the YOLOv8 detector averages approximately 0.025 seconds per frame. By leveraging GPU acceleration, the YOLOv8-based OLSAS achieves a processing speed that is approximately 7.4 times faster than that of the ArUco marker-based detection. Furthermore, the safety check algorithm in the OLSAS functioned as intended. When the bounding boxes of an obstacle and the detected landing pad intersect, the system raises a flag to indicate an unsafe landing condition. This behavior is shown in Figures 8e and 8f, where the UAS correctly identifies the presence of an obstacle within the designated landing zone.

## VII. Limitations

Due to the regulatory constraints of our geographical area, data collection is limited. Collection of training samples from areas in nearby suburban neighborhoods that better represent the expected operational environment of our UAS package delivery. This limits the ability to train our deep learning landing pad detector for off-nominal conditions such as obstacles beyond pedestrians or cars. Furthermore, the custom dataset lacks variability in weather conditions and lighting, which can be very difficult and costly to collect. Both of these limitations could be addressed by the use of synthetic data generation using high-fidelity rendering of realistic landing scenes for training. This would be a more cost effective approach to addressing variation in the dataset, as it is easier to inject adverse scene conditions as well as place objects in unexpected arrangements that can further improve performance on edge cases. Future work should investigate a more complex landing pad dataset that incorporate a wider range of adverse environmental conditions, such as varying lighting, weather, and partial occlusion. Expanding the diversity of the training data in this way can improve the model’s ability to generalize to out-of-distribution scenarios and enhance its robustness in real-world applications.

Additionally, this work evaluates the OLSAS implementation on a workstation equipped with high-performance computational resources, including a multi-core CPU and dedicated GPUs. While this environment facilitates real-time performance and supports deep learning inference with minimal latency, it does not reflect the operational constraints of a physical UAS platform operating in real-world landing scenarios. In real-world deployments, onboard computing systems typically have limited processing power, memory, and energy availability. As such, further investigation is needed to evaluate the feasibility and effectiveness of the proposed system under these constraints. Future work should the deployment of the OLSAS on embedded hardware (such as the NVIDIA Jetson or Coral TPU) to assess performance, inference speed, and reliability in actual flight conditions.

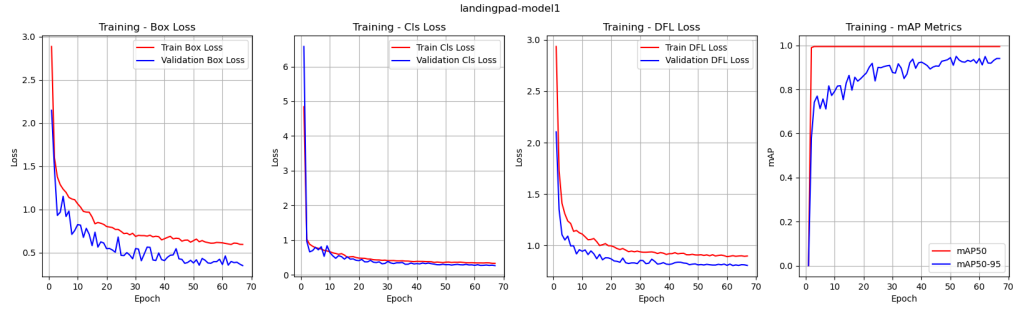
## VIII. Conclusion

In this work, we proposed a system that leverages the object detection capabilities of YOLOv8 to enable safe, autonomous UAS package delivery landings. Our results demonstrate that a deep learning-based detector provides more reliable and consistent identification of landing sites compared to ArUco marker-based methods under nominal environmental conditions, i.e. clear weather and good lighting. The YOLOv8 model, especially when fine-tuned on domain-relevant datasets, achieves high accuracy and operates with significantly reduced inference time, indicating that has potential for real-time deployment using the OLSAS architecture. Additionally, the integration of a safety check algorithm that flags unsafe landings based on object overlap further enhances the safety of autonomous flight in urban environments, where obstacles may move around the landing pad. While the current system was evaluated in a high-performance computing environment, future work will focus on deploying the solution to resource-constrained onboard platforms and expanding the dataset to include more diverse and challenging environmental conditions.

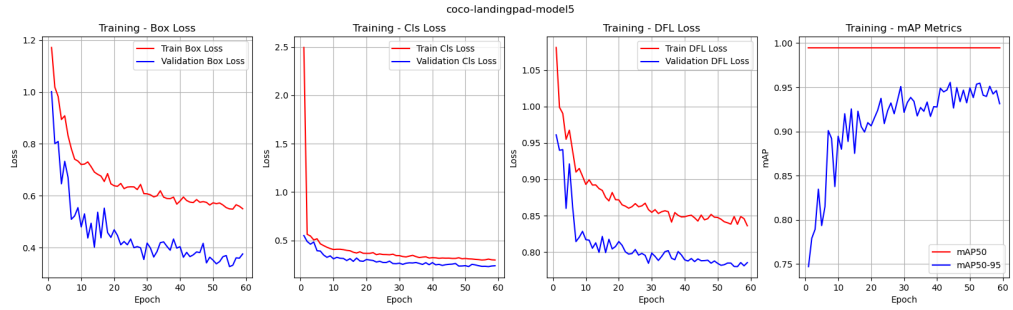
### **Acknowledgments**

We thank our Federal Aviation Administration (FAA) colleagues for their continued funding and support for our research.

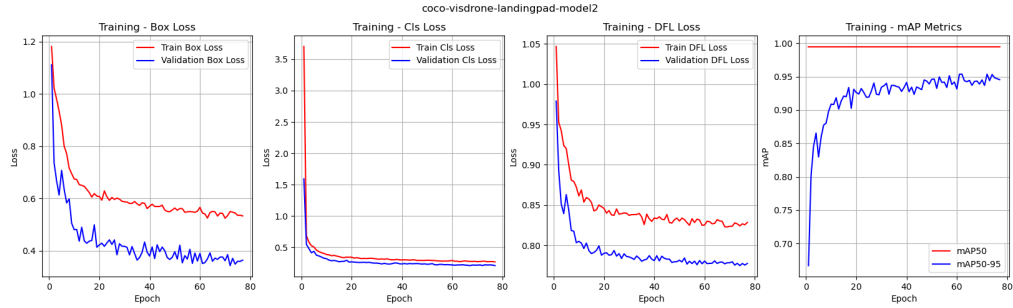




(a) Training loss metrics for the YOLOv8 model trained on the custom dataset from scratch.



(b) Training loss metrics for the YOLOv8 model initialized by COCO pre-trained weights prior to training on the custom dataset.



(c) Training loss metrics for the COCO pretrained YOLOv8 model that was further fine-tuned on VisDrone before further training on the custom dataset.

**Fig. 7** The loss plots include box regression loss, classification loss, and distribution focal loss (DFL), shown for both training and validation on our custom landing pad dataset. In all three models, the training and validation losses decrease steadily, indicating effective learning. The mAP@0.5 and mAP@0.5:0.95 curves rise accordingly, demonstrating improved detection accuracy over time. The consistently high mAP@0.5 scores and early convergence suggest the models are well-suited to the dataset, though they may also reflect the relative simplicity of the training data, which consists only of clear, centered landing pad examples currently. These trends confirm that the YOLOv8 models successfully learn to identify landing pads under the conditions represented in the training sets.



(a)



(b)



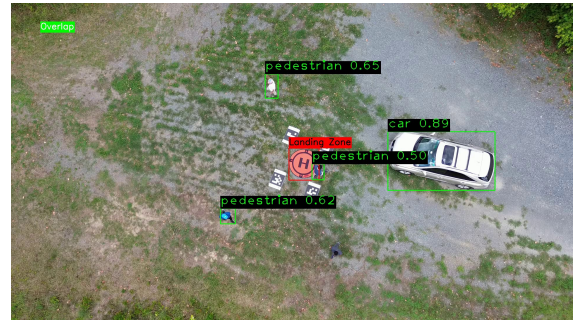
(c)



(d)



(e)



(f)



(g)



(h)

**Fig. 8** Example frames from a flight test video using the OLSAS for the detection of the Hoodman landing pad and surrounding objects. The left column contains frames that were processed through the OLSAS that uses an ArUco-based landing site detection algorithm in the Landing Pad. The right column contains frames where the OLSAS uses YOLOv8-based landing pad detector. When the object bounding boxes overlap with the landing site bounding box, a flag is signaled in the top-left corner of the frame.

## References

- [1] Federal Aviation Administration, “Package Delivery by Drone (Part 135),” , 2023. URL [https://www.faa.gov/uas/advanced\\_operations/package\\_delivery\\_drone](https://www.faa.gov/uas/advanced_operations/package_delivery_drone), accessed: 2024-10-19.
- [2] Federal Aviation Administration, “FAA Makes Drone History in Dallas Area,” , 2024. URL <https://www.faa.gov/newsroom/faa-makes-drone-history-dallas-area>, accessed: 2024-10-19.
- [3] Marquand, Y. L., “FAA authorises Zipline and Wing for BVLOS operations in Dallas,” , 2023. URL <https://www.revolution.aero/news/2024/07/30/faa-authorises-zipline-and-wing-for-bvlos-operations-in-dallas/>, accessed: 2024-10-19.
- [4] Singh, I., “Zipline, Wing enter new drone delivery era with historic FAA waiver,” , 2024. URL <https://dronedj.com/2024/07/30/zipline-wing-drone-delivery-dallas/>, accessed: 2024-10-19.
- [5] Steinberg, D., “Retail Delivery by Drone,” , October 2024. URL <https://time.com/7094858/wing/>, accessed: 2024-11-27.
- [6] Nate Milner, “1 year and 26 cities later: Celebrating our DFW growth,” , 2024. URL <https://blog.wing.com/2024/10/1-year-and-26-cities-later-celebrating.html>.
- [7] Zhao, Z., Lee, J., Li, Z., Park, C. H., and Wei, P., “Vision-based Perception with Safety Awareness for UAS Autonomous Landing,” *AIAA SCITECH 2023 Forum*, American Institute of Aeronautics and Astronautics, National Harbor, MD & Online, 2023. <https://doi.org/10.2514/6.2023-0126>, URL <https://arc.aiaa.org/doi/10.2514/6.2023-0126>.
- [8] Jocher, G., Chaurasia, A., and Qiu, J., “Ultralytics YOLOv8,” , 2023. URL <https://github.com/ultralytics/ultralytics>.
- [9] Rasool, G., Bouaynaya, N., Jalayer, M., and Specht, D., “Rotorcraft Landing Sites – An AI-Based Identification System,” Tech. Rep. CAIT-UTC-REG 32, Rowan University, College of Engineering, 02 2021. URL <https://rosap.ntl.bts.gov/view/dot/61870>.
- [10] Yang, S., Scherer, S., and Zell, A., “An Onboard Monocular Vision System for Autonomous Takeoff, Hovering and Landing of a Micro Aerial Vehicle,” *Journal of Intelligent Robotic Systems*, Vol. 69, 2013. <https://doi.org/10.1007/s10846-012-9749-7>.
- [11] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollar, P., “Focal Loss for Dense Object Detection,” *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [12] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., “You Only Look Once: Unified, Real-Time Object Detection,” *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788. <https://doi.org/10.1109/CVPR.2016.91>.
- [13] Redmon, J., and Farhadi, A., “YOLOv3: An Incremental Improvement,” , 2018. URL <https://arxiv.org/abs/1804.02767>.
- [14] Yang, X., Murphy, M., Brittain, M. W., and Wei, P., “Computer Vision for Small UAS Onboard Pedestrian Detection,” *AIAA AVIATION 2020 FORUM*, American Institute of Aeronautics and Astronautics, VIRTUAL EVENT, 2020. <https://doi.org/10.2514/6.2020-3270>, URL <https://arc.aiaa.org/doi/10.2514/6.2020-3270>.
- [15] imyhxy, “YOLOv5,” , Accessed on 11 2024. URL <https://github.com/ultralytics/yolov5>.
- [16] Wubben, J., Fabra, F., Calafate, C. T., Krzeszowski, T., Marquez-Barja, J. M., Cano, J.-C., and Manzoni, P., “Accurate Landing of Unmanned Aerial Vehicles Using Ground Pattern Recognition,” *Electronics*, Vol. 8, No. 12, 2019. <https://doi.org/10.3390/electronics8121532>, URL <https://www.mdpi.com/2079-9292/8/12/1532>.
- [17] Del Pizzo, S., Papa, U., Gaglione, S., Troisi, S., and Core, G., “A Vision-based navigation system for landing procedure,” *ACTA IMEKO*, Vol. 7, 2018, p. 102. [https://doi.org/10.21014/acta\\_imeko.v7i2.533](https://doi.org/10.21014/acta_imeko.v7i2.533).
- [18] Khalili, B., and Smyth, A. W., “SOD-YOLOv8—Enhancing YOLOv8 for Small Object Detection in Aerial Imagery and Traffic Scenes,” *Sensors*, Vol. 24, No. 19, 2024. <https://doi.org/10.3390/s24196209>, URL <https://www.mdpi.com/1424-8220/24/19/6209>.
- [19] Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J., “Path Aggregation Network for Instance Segmentation,” *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8759–8768. <https://doi.org/10.1109/CVPR.2018.00913>.
- [20] Ju, R.-Y., and Cai, W., “Fracture Detection in Pediatric Wrist Trauma X-ray Images Using YOLOv8 Algorithm,” *Scientific Reports*, Vol. 13, 2023. <https://doi.org/10.1038/s41598-023-47460-7>.
- [21] Zhu, P., Wen, L., Du, D., Bian, X., Fan, H., Hu, Q., and Ling, H., “Detection and Tracking Meet Drones Challenge,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, pp. 1–1. <https://doi.org/10.1109/TPAMI.2021.3119563>.

- [22] AISKYEYE, “Introduction - VISDRONE,” Online, 2025. URL <https://aiskyeye.com/home/>, accessed: May 5, 2025.
- [23] Dwyer, B., Nelson, J., Hansen, T., et al., “Roboflow (version 1.0) [software],” [Online]. Available: <https://roboflow.com>, 2024. Computer vision.
- [24] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L., “Microsoft COCO: Common Objects in Context,” *Computer Vision – ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Springer International Publishing, Cham, 2014, pp. 740–755.