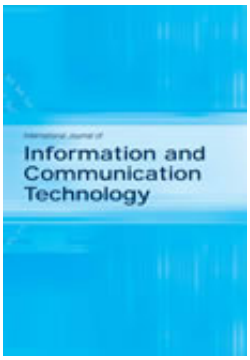


International Journal of Information and Communication Technology

ISSN online: 1741-8070 - ISSN print: 1466-6642
https://www.inderscience.com/ijict

UAV target detection and tracking technology based on deep learning algorithms

Yilu Song, Yue Liu, Yong Wu

DOI: 10.1504/IJICT.2025.10072785

Article History:

Received:	05 June 2025
Last revised:	26 June 2025
Accepted:	27 June 2025
Published online:	26 August 2025

UAV target detection and tracking technology based on deep learning algorithms

Yilu Song*, Yue Liu and Yong Wu

College of Mechanical and Electrical Engineering,
Sanya Aviation and Tourism College,
Sanya, 572000, China
Email: ylu15501976806@163.com
Email: yue123890@126.com
Email: yongzhi123888@yeah.net
*Corresponding author

Abstract: Nowadays, there is an increasing number of tasks on unmanned aerial vehicles (UAVs), which require accurate and real-time analysis ability. In this work, we propose a lightweight and efficient model for object detection and tracking in UAV videos. We use Tiny-YOLOv5 as the detector while DeepSORT is implemented as the tracking module. This combination provides high quality results with low computational cost. Meanwhile, we also introduce multi-modal inputs to improve the detection performance in challenging environments. These inputs are fused with a simple early operation before being passed to the detection model. We evaluate our method on several public UAV datasets. As is shown in results, our system achieves better detection accuracy and tracking stability than baseline models. Meanwhile, according to the evaluation, our model still remains a low computational cost suitable for real-time UAV applications.

Keywords: unmanned aerial vehicle; UAV; object tracking; object detection; computer vision; you only look once; YOLO.

Reference to this paper should be made as follows: Song, Y., Liu, Y. and Wu, Y. (2025) 'UAV target detection and tracking technology based on deep learning algorithms', *Int. J. Information and Communication Technology*, Vol. 26, No. 31, pp.72–87.

Biographical notes: Yilu Song received her Bachelor's degree from Jilin Business and Technology College in 2012. She is currently an Assistant Lecturer at the Sanya Aviation and Tourism College. Her research interests include UAV technology and information engineering.

Yue Liu received her Master's degree from Hebei Normal University in 2017. She is currently a Lecturer at the Sanya Aviation and Tourism College. Her research interests include applied psychology and UAV technology.

Yong Wu received his Bachelor's degree from Civil Aviation University of China in 2007. He is currently an Associate Professor at the Sanya Aviation and Tourism College. His research interests include electronic information engineering and UAV technology.

1 Introduction

Unmanned aerial vehicles (UAVs) are playing a growing role in a variety of fields, including overhead surveillance, traffic analysis, disaster response, and monitoring of environmental conditions (Mohsan et al., 2022). A critical component in many of these tasks is the ability to automatically detect and track objects of interest from UAV-captured imagery (Shakhatreh et al., 2019). Object detection involves identifying and localising instances of predefined categories (e.g., pedestrians and vehicles) in individual frames, while object tracking ensures consistent identification of these objects across sequential frames (MahmoudZadeh et al., 2024). The integration of detection and tracking enables UAVs to perceive dynamic environments, make autonomous decisions, and support real-time situational awareness (Nadeem et al., 2022; Gay et al., 2019). Given the constraints of UAV platforms, such as limited onboard computation and energy, developing lightweight yet accurate detection and tracking systems is essential (Elmokadem and Savkin, 2021).

UAV-based object detection and tracking face several unique challenges due to the nature of aerial imaging and platform constraints. Detection usually faces difficulties including small object sizes, large scale variations, camera motion, frequent occlusions, and complex backgrounds. Objects captured from UAVs often appear small and distorted due to altitude and angle, making accurate localisation difficult. When it comes to tracking, additional challenges arise. Motioning blur, targeting disappearance and appearance variations are hard to address (Mohsan et al., 2023). In the beginning, researchers used traditional machine learning methods to address these challenges, such as SVMs, Adaboost, and handcrafted feature-based models (e.g., HOG, Haar, and SURF) (Ghasemi et al., 2020; Liu et al., 2019; Martinez and Barczyk, 2019). However, they often lacked robustness to scale changes and struggled in cluttered or dynamic scenes. In contrast, deep learning approaches, especially convolutional neural networks (CNNs), have significantly advanced the field (Chen et al., 2023; Yundong et al., 2020; Zhu et al., 2024). These models automatically learn hierarchical features, enabling better handling of scale, shape, and background variations. Nevertheless, many deep models are too heavy and slow for UAVs. This has led to growing interest in building lightweight and fast models that still perform well in complex and changing environments.

In our approach, we adopt Tiny-YOLOv5 for object detection because of its balance between efficiency and accuracy. We integrate DeepSORT to handle the tracking task, which effectively maintains object identities across frames. To further improve performance, we include multi-modal inputs – RGB images, depth maps, and optical flow – which are fused before being passed into the detection model.

Our work integrates the following key contributions:

1 We leverage a compact version of the YOLO detector combined with DeepSORT tracking, ensuring real-time performance without sacrificing detection and tracking accuracy.

2 We introduce the implementation of multi-modal inputs (e.g., visual, depth, and motion data) to enhance detection in challenging environments, such as low visibility or dynamic scenes, improving robustness and reliability.

3 An improved loss function is proposed which optimises both detection and tracking. It achieves better performance by addressing the trade-offs between these two tasks during training.

4 A thorough evaluation is conducted based on multiple UAV-based datasets. The results showcases the balance between speed, accuracy, and computational cost. This demonstrates the applicability of our method for real-time, resource-constrained applications.

Our paper follows the given architecture: Section 2 reviews the related works on object detection and tracking, especially concentrating on deep learning methods. Section 3 describes the detailed methodology of our proposed model. Subsequently, Section 4 shows the evaluation results compared with other baseline models. In Section 5, we conclude this paper and talk about the future plans.

2 Relevant works

Nowadays, there are a large number of research and experiments on UAVs. These works usually concentrate on automatic detection and tracking of objects. Li et al. (2021) developed a lightweight approach called U2U-D&T to support autonomous 'see-and-avoid' functions between UAVs. According to the paper, there are one target detector and one target tracker included in their algorithm. The detection component leverages background subtraction and feature-based classification to identify UAVs based on both motion and appearance cues. Differently, the target tracker uses optical flow and Kalman filtering to track detected targets across frames. Experimental results show improvements over other methods in precision, recall, and F-score. Traditional object detection methods rely on pre-trained networks. In order to solve this challenge, Micheal et al. (2021) introduce an approach which uses a deeply supervised object detector (DSOD) for object detection and long-short-term memory (LSTM) for object tracking. Initially, they trained the DSOD model directly on UAV-specific imagery, which helped it outperform detectors that rely on pre-trained networks. For tracking, they employed an LSTM-based module, which captures temporal information from previous frames and predicts the future positions of objects.

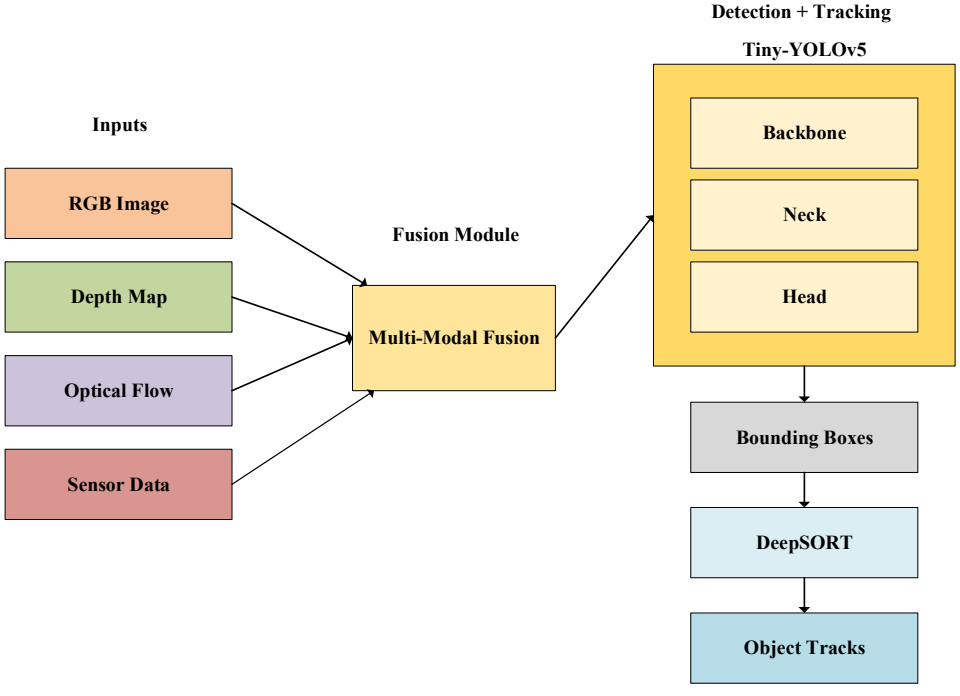
Unlu et al. (2019) introduced a simplified version of the YOLOv3 model designed specifically for detecting drones. Their method combines images from a narrow-angle lens with those from a wide-angle lens, allowing detection to occur in both perspectives at once. This setup lowers GPU memory demand and supports smooth, uninterrupted tracking. In contrast, Nemer et al. (2021) proposed a machine learning-based system that detects and classifies UAVs using radio frequency (RF) data. Their approach uses four classifiers arranged in a step-by-step structure to first detect UAV presence, then recognise the model type, and finally determine its flight status. This layered classification method performed well in a ten-class detection task, indicating strong potential for real-time UAV identification in environments with dense RF signals.

3 Methodology

3.1 Architecture of Tiny-YOLOv5

You only look once (YOLO) is a real-time object detection framework introduced to address the limitations of traditional region-based methods. Unlike earlier approaches such as R-CNN and its variants, which rely on generating region proposals followed by classification, YOLO reframes object detection as a single regression problem. It simultaneously predicts bounding boxes and class probabilities directly from full images in one evaluation, allowing for significant speed improvements. Generally, YOLO has three components: the backbone for feature extraction, the neck for combining features across scales, and the head for producing the final detection outputs. Figure 1 shows the general architecture of our proposed method.

Figure 1 The architecture of our proposed model (see online version for colours)



The backbone serves as the primary component for extracting features, transforming input images into rich, descriptive feature representations. In YOLOv5, this component is built using CSPDarknet, which utilises cross stage partial connections along with residual structures to improve training efficiency and lower unnecessary computation. Once the backbone produces the feature maps, they are sent to the neck module that merges information from different scales. After that, the head uses these processed features to determine object locations and their categories.

The main operation in the backbone is convolution, where an input feature map X is convolved with a filter or kernel K to produce an output feature map Y. This can be represented mathematically as:

$$Y(i,j) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X(i+m,\ j+n) \cdot K(m,n) \tag{1}$$

where M and N denote the dimensions of the convolutional kernel, and (i, j) represents the location of a pixel in the resulting feature map. This process is used to capture local patterns from the input image. After convolution, batch normalisation (BN) is applied to the output feature map to standardise its values. This helps make the training process more stable and faster by minimising shifts in the data distribution within the network. The BN equation is:

$$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}; \quad y = \gamma \hat{x} + \beta \tag{2}$$

In this formula, μ and σ represent the batch's mean and variance, while γ and β are parameters that the model learns during training. The small constant ϵ is added to avoid division by zero.

After applying BN, the network uses a non-linear activation function such as Leaky ReLU to add non-linearity. This function helps the model learn patterns from both positive and negative inputs. It is described mathematically as:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{if } x \le 0 \end{cases} \tag{3}$$

where the activation coefficient α is set to 0.1.

Finally, YOLO employs Residual Connections within its backbone to improve training efficiency and to allow the network to learn more effectively. The residual block can be represented as:

$$F(x) = f(x) + x \tag{4}$$

where $f(x)$ is the result from the convolutional layers and x is fed into the block. Adding x, which is the residual connection, helps retain key features from earlier layers and allows deeper layers to learn more complex representations.

The neck acts as a bridge between the feature extraction module (backbone) and the detection head. It merges feature maps from multiple hierarchical levels to enhance the model's ability to recognise objects at different scales. Commonly, architectures such as the feature pyramid network (FPN) and path aggregation network (PANet) are employed to strengthen multi-scale feature integration. The FPN combines deeper, low-resolution features that carry semantic meaning with shallower, high-resolution features. This improves the quality of feature maps used for detecting objects. PANet further improves this by adding bottom-up paths that allow low-level spatial features to be passed forward more effectively.

The neck includes several key operations. One is upsampling, which increases the resolution of a feature map. This can be written as:

$$X_{up} = Upsample(X) \tag{5}$$

where the width and height of X are doubled using interpolation. The next operation is concatenation, which merges feature maps from different levels of the backbone:

$$F = Concat(X_{up}, Y) \tag{6}$$

where X_{up} is the upsampled feature map and Y refers to another feature map obtained from a different layer of the backbone network. The two feature maps are joined together by stacking them along the channel axis to merge details from multiple scales. Then, a convolution layer is used to blend the features and decrease the number of channels. This is expressed as:

$$F' = Conv(F) \tag{7}$$

where F stands for the merged feature map, and $Conv$ is the convolution filter. These steps in the neck allow the model to mix spatial and meaning-related features from different sizes, helping it better detect small and large objects quickly.

The detection head in our system handles the feature maps from the neck at different scales. It is responsible for producing object detection outputs at three distinct scales, corresponding to small, medium, and large objects. For each spatial location (grid cell) in the feature maps, the head predicts multiple bounding boxes. Each box includes a set of parameters: the centre location (x, y), width w, height h, a score showing whether an object is present, and the probabilities for each class.

We use standard YOLO-style prediction formulas for the bounding box coordinates:

$$Box = \left(\sigma\left(t_x\right) + c_x, \sigma\left(t_y\right) + c_y, e^{t_w} \cdot a_w, e^{t_h} \cdot a_h\right) \tag{8}$$

where t_x, t_y, t_w, t_h are the raw outputs generated by the detection head. The coordinates (c_x, c_y) represent the location of the grid cell. a_w, a_h refer to the anchor box's width and height. The sigmoid function σ is used to keep the predicted centre points inside the limits of the grid cell. The objectness score is computed as:

$$P_{obj} = \sigma\left(t_o\right) \tag{9}$$

where t_o is the predicted objectness logit. This value estimates the confidence that an object exists in the predicted box.

For classification, we use sigmoid activation (multi-label scenario) or softmax (single-label scenario) to get the probability $P(c_i)$ for each class i:

$$P\left(c_i\right) = \sigma\left(t_{c_i}\right) \tag{10}$$

$$P\left(c_i\right) = \frac{e^{t_{ci}}}{\sum_{j=1}^{n} e^{t_{cj}}} \tag{11}$$

In the context of UAV tracking, the head's output is also used to initialise and update object tracks over time. The detected bounding boxes serve as inputs for the tracking algorithms, which associates detections across frames to maintain consistent object identities.

3.2 Architecture of DeepSORT

In our system, the tracking module is based on DeepSORT, which extends SORT by integrating both motion and appearance information to associate objects across video frames. The motion of each object is estimated using a Kalman filter, where the object state vector is defined as:

$$\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}, \dot{r}]^T \tag{12}$$

Here, u and v indicate the bounding box's centre position, while s stands for its size (or area), r represents the ratio between width and height, and the other components describe the corresponding motion speeds. The prediction step of the Kalman filter is expressed as:

$$\hat{\mathbf{x}}_k = \mathbf{F}\mathbf{x}_{k-1} \tag{13}$$

$$\hat{\mathbf{P}}_k = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q} \tag{14}$$

where $\mathbf{F}$ denotes the matrix that governs how the state evolves over time, and $\mathbf{Q}$ represents the covariance of the noise associated with the system's dynamics. Upon receiving a detection, the update step corrects the predicted state:

$$\mathbf{K}_k = \hat{\mathbf{P}}_k\mathbf{H}^T\left(\mathbf{H}\hat{\mathbf{P}}_k\mathbf{H}^T + \mathbf{R}\right)^{-1} \tag{15}$$

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + \mathbf{K}_k\left(\mathbf{z}_k - \mathbf{H}\hat{\mathbf{x}}_k\right) \tag{16}$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}\right)\hat{\mathbf{P}}_k \tag{17}$$

where $\mathbf{H}$ is the observation model, $\mathbf{R}$ is the observation noise covariance, and $\mathbf{z}_k$ is the measurement from the current detection.

To associate detections with existing tracks, DeepSORT formulates a cost matrix using both motion and appearance features. The Mahalanobis distance measures motion-based similarity:

$$d_{motion}(i, j) = \left(\mathbf{z}_j - \hat{\mathbf{z}}_i\right)^T \mathbf{S}_i^{-1}\left(\mathbf{z}_j - \hat{\mathbf{z}}_i\right) \tag{18}$$

where $\hat{\mathbf{z}}_i$ and S_i are the predicted mean and covariance of the i^{th} track, and z_j is the j^{th} detection. For appearance matching, DeepSORT uses a CNN to extract a feature vector f from the object's image region. The cosine distance is used to compute appearance similarity:

$$d_{appearance}(i, j) = 1 - \frac{\mathbf{f}_i \cdot \mathbf{f}_j}{\|\mathbf{f}_i\|\|\mathbf{f}_j\|} \tag{19}$$

The total cost used for the Hungarian algorithm is a weighted combination of both distances:

$$TotalCost(i, j) = \lambda \cdot d_{motion}(i, j) + (1 - \lambda) \cdot d_{appearance}(i, j) \tag{20}$$

3.3 Data fusion

Data fusion with multi-modal inputs involves combining information from RGB cameras and thermal (infrared) cameras, which improves the performance of object detection and tracking tasks. In UAV applications, RGB images provide rich detail in good lighting, while thermal images capture heat signatures and are useful in low-light or night conditions. By fusing these inputs, the system gains a more complete understanding of the scene, allowing it to detect and track objects more reliably across diverse environments. Fusion can be done at different stages of the pipeline: early fusion merges raw RGB and thermal images at the input level, while late fusion combines features extracted separately by each modality.

3.4 Loss function

In this research, the detection model is refined using an integrated loss function that consists of three main components: localisation through bounding box regression, objectness score estimation, and classification across multiple categories. For the regression component, the complete intersection over union (CIoU) loss is employed, which builds upon conventional IoU by incorporating the centre-point distance between predicted and ground truth boxes. The CIoU loss is formulated as:

$$\mathcal{L}_{CIoU} = 1 - \mathrm{IoU} + \frac{\rho^2\left(\mathbf{b}, \mathbf{b}^{gt}\right)}{c^2} + \alpha v \tag{21}$$

where $\rho^2(\mathbf{b}, \mathbf{b}^{gt})$ represents the Euclidean distance between the centres of the $\mathbf{b}$ and the ground truth box $\mathbf{b}^{gt}$. c refers to the diagonal length of the smallest box that encloses both $\mathbf{b}$ and $\mathbf{b}^{gt}$. v evaluates the aspect ratio alignment, while α a positive parameter that balances the overall trade-off..

To guide the model in distinguishing foreground objects from background, we use binary cross-entropy (BCE) loss for the objectness score. It is defined as:

$$\mathcal{L}_{obj} = -\left[y \cdot \log\left(\hat{p}\right) + (1-y) \cdot \log\left(1-\hat{p}\right)\right] \tag{22}$$

where $y \in \{0, 1\}$ is the ground truth indicator of object presence and $\hat{p}$ is the predicted confidence score.

For multi-class object classification, we use the categorical cross-entropy loss across all predicted classes. This loss is given by:

$$\mathcal{L}_{cls} = -\sum_{i=1}^{C} y_i \cdot \log\left(\hat{p}_i\right) \tag{23}$$

where C is t the overall count of categories, y_i the actual label for the i^{th} category, and $\hat{p}_i$ indicate the estimated probability corresponding to class i.

The final detection loss $\mathcal{L}_{\text{det}}$ combines the three terms as a weighted sum:

$$\mathcal{L}_{\text{det}} = \lambda_1 \mathcal{L}_{CIoU} + \lambda_2 \mathcal{L}_{obj} + \lambda_3 \mathcal{L}_{cls} \tag{24}$$

where λ_1, λ_2, λ_3 serve as tuning parameters that regulate the influence of each individual loss component. In this approach, fusion of RGB and thermal images is performed either

at the initial input stage or within the feature extraction layers. No additional loss terms are required for handling the multi-modal data. The model learns to extract features from both modalities during training, and the standard YOLOv5 losses are used to optimise the network for object detection.

4 Evaluation

4.1 Datasets

In this study, we use four well-known open datasets to test how well our detection and tracking system works: UAVDT (Li et al., 2021), VisDrone (Muzammul et al., 2024), DOTA (Terven et al., 2023), and UA-DETRAC (Wen et al., 2020). UAVDT offers 100 video clips, giving about 80,000 frames. Each frame has a resolution of 1,080 × 540. It contains labelled types of vehicles like cars, buses, and trucks in different weather and road scenes. VisDrone provides 263 video files and more than 10,000 labelled pictures. The image size is 1,920 × 1,080. It shows people, vehicles, and bikes in crowded city areas. DOTA is a large dataset for finding objects in aerial photos. It holds over 2,800 high-resolution images and includes more than 180,000 labelled targets across 15 categories. The image sizes vary, and labels are drawn with rotated boxes. UA-DETRAC focuses on vehicle tracking and detection. It has 100 videos and 1.2 million labelled frames with a resolution of 960 × 540.

4.2 Metrics for evaluation

To measure the effectiveness of the object detection and tracking framework that integrates Tiny-YOLOv5 with DeepSORT using multi-modal input data, a set of widely accepted evaluation metrics is applied.

Among these, mean average precision (mAP) stands out as a fundamental metric in object detection. It is computed by taking the average of precision scores across multiple recall levels for each class. The mAP is calculated as follows:

$$mAP = \frac{1}{n}\sum_{i=1}^{n} \mathrm{AP}(i) \tag{25}$$

In this equation, n indicates the number of object classes, while $AP(i)$ refers to the average precision for the i^{th} class. Average precision is calculated as the area under the precision-recall curve for each category. Precision values are obtained at different levels of recall and are defined as follows:

$$Precision = \frac{TP}{TP + FP} \tag{26}$$

In this context, TP refers to true positives, while FP denotes false positives. Recall, by contrast, evaluates the proportion of actual objects that were correctly identified, and it is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{27}$$

where *FN* stands for false negatives. Another important measurement is intersection over union (IoU), which quantifies the degree of overlap between the predicted bounding box and the true bounding box. It is computed using the formula:

$$IoU = \frac{Area\ of\ intersection}{Area\ of\ union} \tag{28}$$

This measure assesses if the predicted bounding box overlaps enough with the actual object to qualify as an accurate detection.

When assessing multi-object tracking performance, the multiple object tracking accuracy (*MOTA*) metric is commonly employed. It takes into consideration false positives (*FP*), missed detections (*FN*), and identity mismatches (*IDSW*), and is given by:

$$MOTA = 1 - \frac{FP + FN + IDSW}{GT} \tag{29}$$

where *GT* represents the total count of true objects present in the dataset. A greater MOTA value signifies improved tracking accuracy. Meanwhile, multiple object tracking precision (*MOTP*) evaluates the accuracy of the spatial alignment between the tracked objects and the actual ground truth:

$$MOTP = \frac{\sum_i \mathrm{IoU}_i}{N_{TP}} \tag{30}$$

where IoU_i represents the intersection-over-union score for the i^{th} tracked object and N_{TP} is the number of correctly identified instances. Fragmentation (*FM*) quantifies the frequency at which tracked objects are temporarily lost and subsequently reacquired throughout the video sequence, defined as:

$$Fragmentation = \sum_{i=1}^{n} \frac{F_i}{N_i} \tag{31}$$

where *F* represents the count of fragmentation occurrences for an object, while *N* indicates the total frames during which the object appears.

4.3 Experimental setup

The system was implemented using Python in combination with the PyTorch library. Training took place on a workstation equipped with an NVIDIA RTX 3090 graphics card, which has 24 GB of dedicated memory. Input images were resized to 640 × 640 pixels prior to feeding them into the model. The training process began with an initial learning rate set at 0.001 and a batch size of 16. Model optimisation utilised stochastic gradient descent (SGD) with a momentum coefficient of 0.937 and a weight decay value of 0.0005. The network underwent training for 100 complete iterations (epochs). To improve robustness and prevent overfitting, a variety of data augmentation techniques were applied, such as random horizontal flipping, colour distortion, and mosaic-style augmentation.

4.4 Detection evaluation on multiple UAV datasets

In this experiment, object detection models are tested on four UAV datasets. The models are Faster R-CNN, SSD, and RetinaNet. Each model uses RGB images as input. No tracking method is used. The goal is to test how well each detector works on different types of UAV data. The results are shown as given in Figures 2–5.

Figure 2 Detection results on UAVDT (see online version for colours)

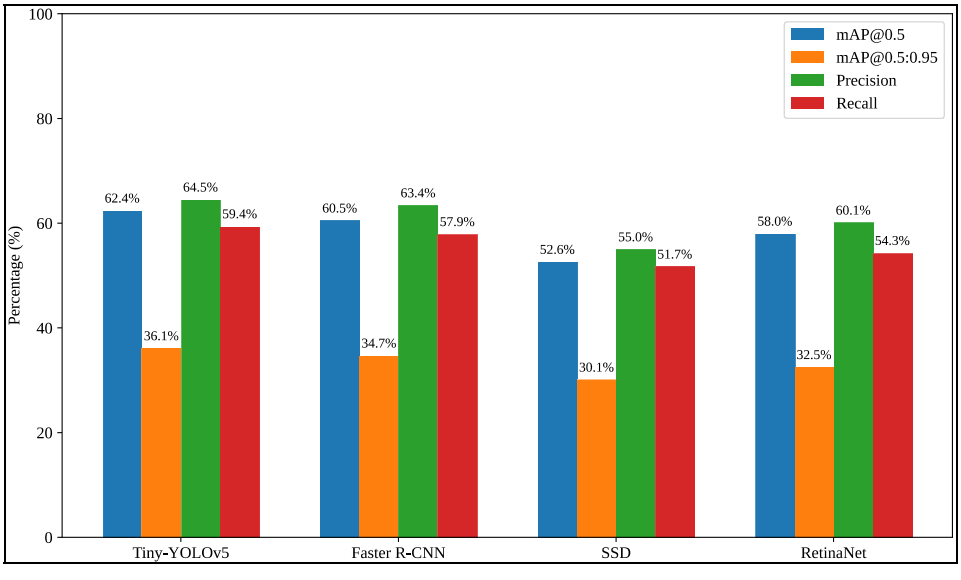
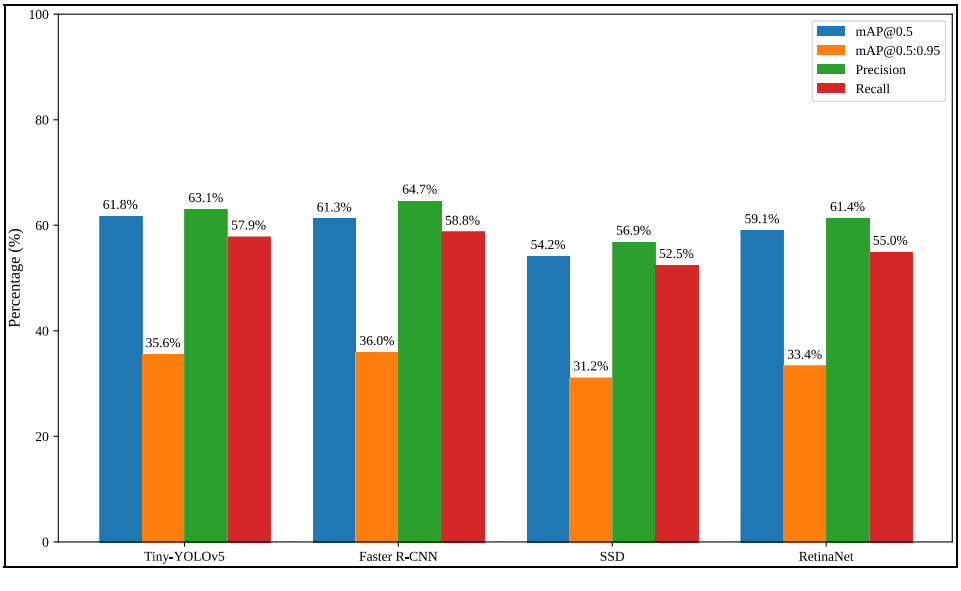


Figure 3 Detection results on VisDrone (see online version for colours)



Across the four datasets (UAVDT, VisDrone, DOTA, and UA-DETRAC), Tiny-YOLOv5 consistently demonstrates superior overall performance, achieving the highest or near-highest scores in both mAP@0.5 and mAP@0.5:0.95 across all benchmarks. This suggests it is particularly well-suited for UAV-based object detection tasks where speed and accuracy are both critical.

Figure 4 Detection results on DOTA (see online version for colours)

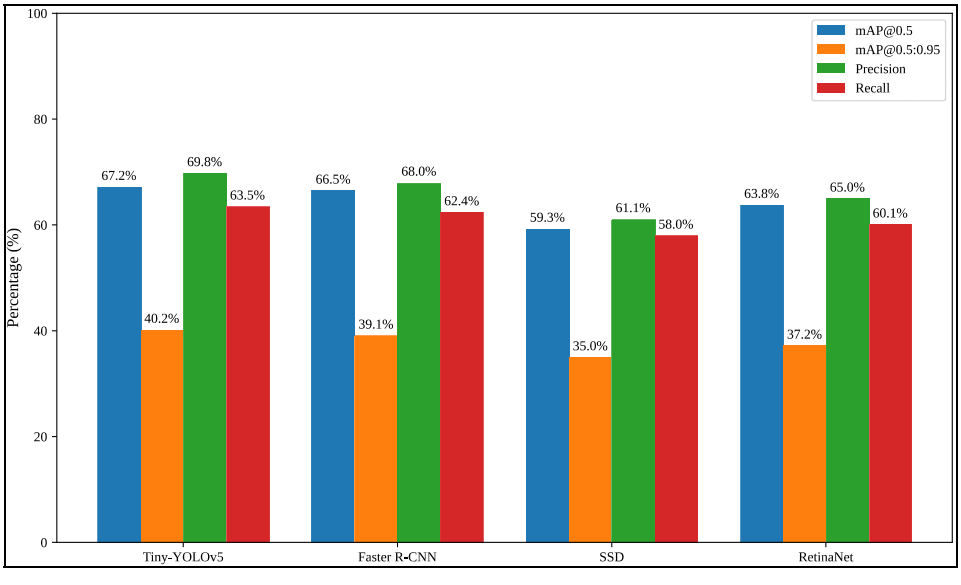
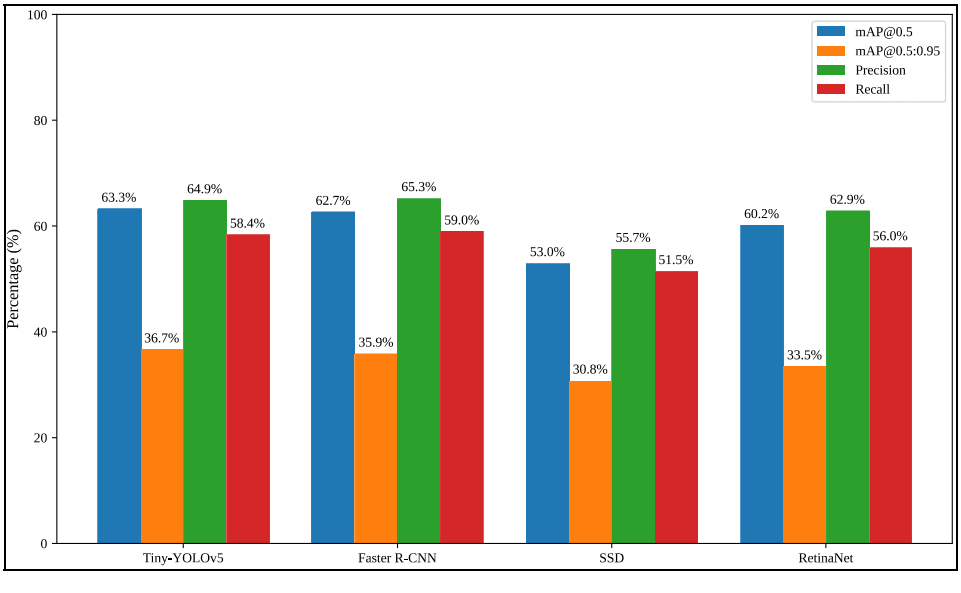


Figure 5 Detection results on UA-DETRAC (see online version for colours)



Faster R-CNN shows strong results in terms of precision and recall, often leading in recall, indicating its robustness in identifying true positives, though it slightly trails Tiny-YOLOv5 in mAP scores due to its heavier computational cost. RetinaNet performs moderately across all metrics, offering a balance between accuracy and complexity but not outperforming the top models. SSD consistently ranks lowest in all datasets, highlighting its limitations in handling the challenges posed by aerial imagery, such as small object sizes and complex backgrounds. The advantages of Tiny-YOLOv5 lie in its lightweight architecture and efficient inference speed, which make it especially suited for real-time detection on resource-constrained UAV platforms. Its ability to balance detection accuracy and computational efficiency allows it to maintain high precision even in complex and variable aerial scenes, making it an ideal choice for practical deployment.

4.5 Tracking evaluation on multiple UAV datasets

In this section, we test five tracking models: DeepSORT, SORT, ByteTrack, OC-SORT, and FairMOT. All models use detection results from Tiny-YOLOv5 on the VisDrone and UA-DETRAC datasets. Each tracker runs at 30 FPS. All experiments use the same hardware and same detector output to ensure fair comparison.

Table 1 Tracking comparison on VisDrone and UA-DETRAC (with Tiny-YOLOv5 detection)

Tracker	*Dataset*	*MOTA*	*MOTP*	*IDSW*	*MT (%)*
DeepSORT	VisDrone	49.7%	75.4%	31	44.2
SORT	VisDrone	42.5%	72.0%	69	35.0
ByteTrack	VisDrone	53.9%	76.1%	22	47.8
OC-SORT	VisDrone	51.2%	75.7%	29	45.3
FairMOT	VisDrone	56.3%	77.5%	15	50.5
DeepSORT	UA-DETRAC	54.5%	77.0%	18	50.3
SORT	UA-DETRAC	46.0%	73.4%	40	42.1
ByteTrack	UA-DETRAC	58.2%	78.1%	11	54.7
OC-SORT	UA-DETRAC	56.0%	77.3%	16	52.0
FairMOT	UA-DETRAC	59.4%	78.9%	9	55.5

The results in Table 1 show that ByteTrack and FairMOT give better accuracy and fewer ID switches. But they are slower. SORT runs faster but has more ID switches. DeepSORT gives a good balance between speed and ID accuracy. OC-SORT works well in fast object motion. This experiment shows the strengths and weaknesses of each model.

4.6 Ablation study

In this experiment, we test different parts of our model to see how each part helps. We start with only Tiny-YOLOv5 for detection. Then we add DeepSORT for tracking. After that, we add multi-modal inputs. Last, we apply the improved loss function. We test all four versions on the UAVDT dataset. The same settings are used for all tests. Each setup runs on the same machine with the same FPS.

We use two detection metrics: mAP@0.5 and mAP@0.5:0.95. We also use two tracking metrics: MOTA and IDSW. Results show that each part improves the performance. Adding DeepSORT helps track objects better. Multi-modal inputs give better detection in difficult scenes. The new loss function improves both detection and tracking accuracy. The evaluation results are shown in Table 2.

Table 2 Ablation study on UAVDT dataset

Method	*mAP@0.5*	*mAP@0.5:0.95*	*MOTA*	*IDSW*
Tiny-YOLOv5 only	62.1%	39.2%	-	-
+ DeepSORT	62.1%	39.2%	52.3%	25
+ multi-modal inputs	65.4%	42.8%	55.1%	20
+ improved loss function	67.3%	44.9%	58.6%	16

4.7 Computational cost analysis

In this section, we evaluate the computational cost and efficiency of each model version. We include five baseline models and our proposed model. The baselines are YOLOv5 + DeepSORT, YOLOv3 + SORT, Faster R-CNN + DeepSORT, FairMOT, and CenterTrack. Our model uses a lightweight detector, multi-modal inputs, DeepSORT tracking, and a custom loss function. All inputs are resized to 640×640. We record frames per second (FPS), GPU memory (MB), and total parameters (M).

Table 3 Full model computational cost comparison

Model	*Detector*	*Tracker*	*FPS*	*GPU memory (MB)*	*Parameters (M)*
YOLOv5 + DeepSORT	YOLOv5s	DeepSORT	65.2	890	7.9
YOLOv3 + SORT	YOLOv3-tiny	SORT	82.4	730	8.2
Faster R-CNN + DeepSORT	Faster R-CNN	DeepSORT	28.9	1,260	42.3
FairMOT	Joint (HRNet18)	-	35.5	1,450	34.0
CenterTrack	Joint (DLA-34)	-	39.7	1,380	30.2
Ours	Light CNN + MM	DeepSORT	48.6	1,105	8.5

Table 3 shows that our model stays light and fast while adding more inputs and better training. It runs faster than FairMOT and CenterTrack and uses less memory than Faster R-CNN.

5 Conclusions

This study presents a compact and efficient approach that combines Tiny-YOLOv5 with DeepSORT to perform object detection and tracking in aerial footage captured by UAVs. We also combine multi-modal inputs and a tailored loss function to improve accuracy

and robustness. Extensive experiments on public UAV datasets demonstrate that our model achieves competitive performance while maintaining real-time speed and low computational cost. The suggested system achieves an effective compromise between speed and precision, rendering it ideal for surveillance and monitoring applications using UAVs, especially in settings with limited computational resources.

Although our method shows promising results for UAV-based object detection and tracking, several avenues for improvement and exploration remain. A key direction for further research involves enhancing the model's efficiency to enable deployment on edge platforms, including drones that have constrained processing capabilities. Furthermore, extending the tracking capabilities for long-term missions, with better handling of ID switches and tracking for extended periods, would be valuable. Adapting the framework for other UAV tasks could also open up additional opportunities for practical applications.

Acknowledgements

This work is supported by the school level project of Sanya Aviation and Tourism College (No. SATC2024JC-07) and the Philosophy and Social Science Planning Project of Sanya City (No. SYSK2023-05).

Declarations

All authors declare that they have no conflicts of interest.

References

Chen, C., Zheng, Z., Xu, T., Guo, S., Feng, S., Yao, W. and Lan, Y. (2023) 'YOLO-based UAV technology: a review of the research and its applications', *Drones*, Vol. 7, No. 3, p.190.

Elmokadem, T. and Savkin, A.V. (2021) 'Towards fully autonomous UAVs: a survey', *Sensors*, Vol. 21, No. 18, p.6223.

Gay, C., Horowitz, B., Elshaw, J.J., Bobko, P. and Kim, I. (2019) 'Operator suspicion and human-machine team performance under mission scenarios of unmanned ground vehicle operation', *IEEE Access*, Vol. 7, pp.36371–36379.

Ghasemi, M., Varshosaz, M. and Pirasteh, S. (2020) 'Evaluating sector ring histogram of oriented gradients filter in locating humans within UAV images', *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. 43, pp.23–27.

Li, J., Ye, D.H., Kolsch, M., Wachs, J.P. and Bouman, C.A. (2021) 'Fast and robust UAV to UAV detection and tracking from video', *IEEE Transactions on Emerging Topics in Computing*, Vol. 10, No. 3, pp.1519–1531.

Liu, X., Guo, X., Zhao, D., Cao, H., Tang, J., Wang, C., Shen, C. and Liu, J. (2019) 'Integrated velocity measurement algorithm based on optical flow and scale-invariant feature transform', *IEEE Access*, Vol. 7, pp.153338–153348.

MahmoudZadeh, S., Yazdani, A., Kalantari, Y., Ciftler, B., Aidarus, F. and Al Kadri, M.O. (2024) 'Holistic review of UAV-centric situational awareness: applications, limitations, and algorithmic challenges', *Robotics*, Vol. 13, No. 8, p.117.

Martinez, P. and Barczyk, M. (2019) 'Implementation and optimization of the cascade classifier algorithm for UAV detection and tracking', *Journal of Unmanned Vehicle Systems*, Vol. 7, No. 4, pp.296–311.

Micheal, A.A., Vani, K., Sanjeevi, S. and Lin, C-H. (2021) 'Object detection and tracking with UAV data using deep learning', *Journal of the Indian Society of Remote Sensing*, Vol. 49, pp.463–469.

Mohsan, S.A.H., Khan, M.A., Noor, F., Ullah, I. and Alsharif, M.H. (2022) 'Towards the unmanned aerial vehicles (UAVs): a comprehensive review', *Drones*, Vol. 6, No. 6, p.147.

Mohsan, S.A.H., Othman, N.Q.H., Li, Y., Alsharif, M.H. and Khan, M.A. (2023) 'Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends', *Intelligent Service Robotics*, Vol. 16, No. 1, pp.109–137.

Muzammul, M., Algarni, A., Ghadi, Y.Y. and Assam, M. (2024) 'Enhancing UAV aerial image analysis: Integrating advanced SAHI techniques with real-time detection models on the VisDrone dataset', *IEEE Access*, Vol. 12, pp.21621–21633.

Nadeem, A., Ashraf, M., Qadeer, N., Rizwan, K., Mehmood, A., AlZahrani, A., Noor, F. and Abbasi, Q.H. (2022) 'Tracking missing person in large crowd gathering using intelligent video surveillance', *Sensors*, Vol. 22, No. 14, p.5270.

Nemer, I., Sheltami, T., Ahmad, I., Yasar, A.U-H. and Abdeen, M.A. (2021) 'RF-based UAV detection and identification using hierarchical learning approach', *Sensors*, Vol. 21, No. 6, p.1947.

Shakhatreh, H., Sawalmeh, A.H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N.S., Khreishah, A. and Guizani, M. (2019) 'Unmanned aerial vehicles (UAVs): a survey on civil applications and key research challenges', *IEEE Access*, Vol. 7, pp.48572–48634.

Terven, J., Córdova-Esparza, D-M. and Romero-González, J-A. (2023) 'A comprehensive review of YOLO architectures in computer vision: from YOLOv1 to YOLOv8 and YOLO-NAS', *Machine Learning and Knowledge Extraction*, Vol. 5, No. 4, pp.1680–1716.

Unlu, E., Zenou, E., Riviere, N. and Dupouy, P-E. (2019) 'Deep learning-based strategies for the detection and tracking of drones using several cameras', *IPSJ Transactions on Computer Vision and Applications*, Vol. 11, pp.1–13.

Wen, L., Du, D., Cai, Z., Lei, Z., Chang, M.-C., Qi, H., Lim, J., Yang, M-H. and Lyu, S. (2020) 'UA-DETRAC: a new benchmark and protocol for multi-object detection and tracking', *Computer Vision and Image Understanding*, Vol. 193, p.102907.

Yundong, L., Han, D., Hongguang, L., Zhang, X., Zhang, B. and Zhifeng, X. (2020) 'Multi-block SSD based on small object detection for UAV railway scene surveillance', *Chinese Journal of Aeronautics*, Vol. 33, No. 6, pp.1747–1755.

Zhu, J., Ma, C., Rong, J. and Cao, Y. (2024) 'Bird and UAVs recognition detection and tracking based on improved YOLOv9-DeepSORT', *IEEE Access*.