Turkish Journal of Engineering

https://dergipark.org.tr/en/pub/tuje

e-ISSN 2587-1366

Digital twin of multi-model drone detection system on Airsim for RF and vision modalities

Yusuf Özben [1], Süleyman Emre Demir [1], Hüseyin Birkan Yılmaz [*1]

[1] Bogaziçi University, Department of Computer Engineering, Türkiye, yusufozben@gmail.com, SuleymanD41k9@gmail.com, birkan.yilmaz@bogazici.edu.tr

Cite this study: Özben, Y., Demir, S. E., & Yılmaz, H. B. (2024). Digital twin of multi-model drone detection system on Airsim for RF and vision modalities. Turkish Journal of Engineering, 8 (3), 572-582

https://doi.org/10.31127/tuje.1436757

Keywords
Drone detection
Signal processing
Image processing
Machine learning
Simulation

Research Article

Received: 13.02.2024
Revised: 11.03.2024
Accepted: 12.03.2024
Published: 15.07.2024

Abstract

Drones have become more prevalent in recent years and are used for both beneficial and malicious purposes. As a result, protecting restricted areas from unauthorized drone activities has become crucial. However, some researchers face challenges in developing drone detection systems due to the high costs of necessary equipment. This paper presents an innovative solution by creating an Airsim Graphical User Interface (GUI) tool compatible with the Unreal Engine. This tool enables the simulation of drone flights and creation of image and radio frequency (RF) datasets for drone detection in a simulation environment. Our approach involves modeling the measurement devices such as cameras to capture image data and software defined radio (SDR) receiver to capture RF signals as raw in-phase and quadrature (IQ) data. Moreover, users can manage automated route planning for drones, recording configurations, and different cameras and RF configurations. Researchers can now generate datasets with various images and RF configurations without the need for physical drones, cameras, or SDRs, enabling experimentation with different drone detection models. Furthermore, we proposed models for drone detection systems by using generated datasets from the proposed dataset generation system.

1. Introduction

Drone usage increased dramatically all over the world. As of 15 January 2024, there are 863,728 registered drones in the US, according to the Federal Aviation Administration (FAA). Moreover, in Turkey, the number of registered drones reached 68,426 in June 2023. Drones have many useful applications like 3D modelling of historical places [1-5] and communication technologies [6], but they can also be used for illegal activities such as transporting prohibited substances and carrying out attacks. Automated drone detection systems using technologies like image, radar, and AI algorithms are needed to prevent these activities in certain areas like airports and government buildings.

Automated drone detection systems can be built using different techniques. However, in the literature, there are four main methods: vision, RF, acoustic, and radar. These methods have several advantages and disadvantages, as shown in Table 1. Multi-model drone detection systems that use two or more detection methods are built to eliminate disadvantages.

There are several research for drone detection systems in the literature. For these drone detection systems, different types of equipment like cameras, drones, and radars are used. However, there are some researchers who are unable to access these pieces of equipment. They generally use publicly available drone detection datasets. However, these publicly available datasets are created for specific tasks like classifying drones and birds in images or classifying three drone brands by using RF signals. Also, created labels could be limited for researchers. For example, a researcher could need Cartesian coordinates of drones in image datasets, but in these datasets generally only bounding boxes are published. In addition, a researcher could need a dataset that uses multiple drone detection methods. For example, we could not find any combined image and RF dataset in the literature.

In this paper, we proposed a solution for creating datasets for drone detection systems in a simulation environment without any other drone detection hardware requirement. The proposed system supports only image and RF dataset generation for now, and we plan to extend the simulation capabilities by adding other types of measurement devices. We chose image and RF dataset generation because they are the most popular two drone detection methods in the literature. The

created solution uses Unreal Engine for the simulation environment and Airsim for drone simulation. In addition, the Airsim GUI is proposed for managing drone detection data generation steps on a single screen. From the Airsim GUI, drones, cameras, RF, recording, and autonomous drone path settings can be configured, and datasets can be generated based on settings.

The remainder of the paper is organized as follows, Section 2 provides the literature review about related topics, Section 3 details the created dataset generation simulation architecture for drone detection, Section 4 gives example of possible use cases and drone detection models for simulation datasets, and Section 5 concludes the article, identifies the contributions, and discuss the future directions.

Table 1. Drone detection methods and their advantages and disadvantages.

Method	Advantages	Disadvantages
Camera	Cost-effective, provide visual evidence, etc.	Different scale and shape of drones, similarities with birds, different weather and lighting conditions, requires line-of-sight, etc.
RF	Resistance to different weather and lighting conditions, long range detection, no need for line-of-sight, etc.	Hard to differentiate from other RF resources, cannot detect completely autonomous drones, etc.
Acoustic	Easy to deploy, cost-effective, no need for line-of-sight, etc.	A lot of background noise, short range, etc.
Radar	Long range, resistance to autonomous drones, localization, etc.	Hard to detect small drones, high cost, etc.

2. Literature review

In the literature, several publicly published datasets exist for drone detection systems and proposed drone detection models. We summarize the literature for image and RF based datasets and drone detection methods, and drone detection in simulation environments.

2.1. Image-based drone detection datasets

In the literature, the most accessible datasets are image-based drone detection datasets than the other methods. The main reason is creating an image dataset is easier than the other detection methods.

There are various workshops available that focus on image-based drone detection. One of the most popular workshops is the Anti-UAV Workshop & Challenge. This challenge happened three times and there are a publicly published datasets for these challenges. The first challenge was organized in conjunction with Computer Vision and Pattern Recognition Conference (CVPR) as the first International Workshop on Anti-UAV Challenge [7], and 160 RGB and infrared video sequences were published. The 2nd anti-UAV Workshop & Challenge [8] was organized in conjunction with International Conference on Computer Vision (ICCV) 2021, and they extended their dataset with 280 HQ videos which are captured with a thermal infrared camera and contain different scale multiple drones. Finally, the last challenge [9] was organized in conjunction with CVPR 2023 with an extended dataset.

The Drone vs. Bird Detection Challenge is another well-known challenge for drone detection. This challenge began during the International Workshop on Small-Drone Surveillance, Detection, and Counteraction Techniques (WOSDETC) [10] at Conference on Advanced Video and Signal-Based Surveillance (AVSS) 2017 and has occurred five times since [11-14]. During these workshops, image datasets were published. The main focus of these datasets was detecting small drones and differentiating them from birds.

Li et al. [15] published a public dataset for reconstructing 3D flight trajectories by using multiple cameras. The dataset includes five sub-datasets, each with variations in the number of cameras, locations, and synchronization configurations.

2.2. RF-based drone detection datasets

For RF-based drone detection, there are no such workshops as in image-based drone detection. However, in the literature, there are several public datasets.

Allahham et al. [16] has one of the most popular RF datasets for drone detection. It contains time series RF data recorded using universal software radio peripheral (USRP) in 2.4 GHz. Collected RF data contains Parrot Bebop, Parrot AR, and DJI Phantom 3 drones and only background RF activity. They also categorize the drone data into the drone modes as on, hovering action, flying, and video recording. The dataset serves 2 classes for drone detection, 3 classes for drone type classification, and 10 classes for drone mode classification.

Swinney et al. [17] and Glüge et al. [18] publicly shared a dataset that contains raw IQ data. Swinney et al. [17] made available the DroneDetect V2 dataset, which includes drone signals with other signal interference in the same frequency band as Bluetooth. They used a BladeRF software-defined radio (SDR) and three drone models. The created class types are similar to those in Allahham et al. [16], with 2 classes for drone detection, 4 classes for drone type classification, and 10 classes for drone mode classification. On the other hand, the dataset published in Glüge et al. [18] not only contains drone signals but also remote controller signals. After recording the signals, they applied Labnoise and Gaussian noise to the drone signals.

The dataset published in Medaiyese et al. [19] is known as the Cardinal RF (CardRF) dataset and frequently used in research. This dataset's primary contribution is that it includes labeled drone RF data from both line-of-sight (LoS) and non-line-of-sight (NLOS) scenarios. Additionally, the dataset contains various types of drones and 2.4 GHz transmitters.

2.3. Drone detection via RF dataset

Because creating RF dataset for drone detection from scratch is exhausting, mainly publicly available datasets are used in the literature for drone detection research.

The DroneRF dataset in Allahham et al. [16] has been used in various research studies [20-25]. Al-Sa'd et al. [20] outline the dataset and uses a deep neural network for analysis. For the 2-class problem, they reported the average accuracy as 99.7%, for the 4-class problem 84.5%, and for the 10-class problem 46.8%. Al-Emadi and Al-Senaid [21] and Allahham et al. [22] propose a solution using a 1D convolutional neural network, achieving 59.2% and 87.4% accuracy in the 10-class problem, respectively. In addition, Medaiyese et al. [23] uses XGBoost, a more resource-friendly machine learning system than a deep neural network, to achieve 70.1% accuracy for the 10-class classification. Nemer et al. [24] proposed a hierarchical model that improves the 10-class problem using concatenated classifiers. The first set of classifiers addresses a 2-class drone detection problem. Following that, if a drone is detected, they proceed to solve a 3-class drone-type detection problem. Finally, the last set of classifiers is employed to tackle the task of identifying drone modes. Furthermore, they used two different classification methods for their classifiers: XGBoost and K-Nearest Neighbors (KNN). Ensemble learning is used for decision. Their model achieved 99.2% accuracy for the 10-class problem.

Glüge et al. [18] not only shares the dataset but also compares the performance of drone detection using IQ data and 2D spectrogram. They utilized a very deep convolutional network (VGGNet) and showed that a 2D spectrogram is more reliable in detecting drones in lower SNR conditions.

Nguyen et al. [26] propose a drone detection system using two different software-defined radios and two different drones. Their objective was to locate the drone and its controller in 3D space by combining angle of arrivals (AoA) and distances. The proposed system achieved a 12.2-degree error in identifying the drone's direction, with a localization error of 12.71 meters. Moreover, the system obtained a 9.9-degree error in identifying the controller's direction, and an accuracy of 11.36 meters in locating the controller.

2.4. Drone detection via camera

In the literature, there are several drone detections via image exist. They use both publicly available datasets and custom created datasets.

The YOLO series is a popular choice for detecting objects, including drones. Unlu et al. [27] use two cameras for drone detection. One camera had a dynamic platform and lens for a low-angle view, while the other had a static platform and lens for a wide-angle view. They detected objects with the wide-angle camera and classified them using the low-angle camera with 35X zoom. This approach allowed them to capture drones as small as 6x6 pixels using YOLOv3. They achieved a 0% false alarm rate and a 91% true positive rate. Singha and Aydin [28] used YOLOv4 for drone detection and obtained a 74.36% mAP50 score using a dataset that contained both bird and drone images. Zhai et al. [29], the latest version of YOLO, YOLOv8s (YOLOv8 small model), was used. The researchers modified the YOLOv8 and achieved even higher performance metrics. They achieved an 85.2% mAP score using the original version of YOLOv8s (YOLOv8 small model), which was improved to 95.3% with their optimized version of YOLOv8.

Seidaliyeva et al. [30] proposed a solution for static cameras using the drone-vs-bird dataset. The researchers divided the drone detection problem into two phases: moving object detection and classification. They employed the background subtraction method to detect moving objects and used MobileNetV2 to classify drones, birds, and backgrounds. For their evaluation, they reported an F1-score of 0.742 for IoU at 0.3.

2.5. Drone detection in simulation environments

Game engines are useful for drone detection tasks due to their reality. However, game engines mainly used for drone detection via images in the literature.

Drone detection by utilizing another drone's camera is investigated as a problem in Carrio et al. [31]. To address this issue, the authors propose a solution that employs a depth map. They test this approach in real-world settings after producing synthetic data in Unreal Engine. The same team continues their research in Carrio et al. [32] and improves their approach by using the same synthetic data generation method.

During the Drone-vs-Bird challenge held at AVSS in 2021, the winning team utilized a simulation environment to improve their score [33]. They employed a 3D drone model and diverse 2D backgrounds, which were incorporated into the Unity game engine. Subsequently, they captured images from the game engine and merged them with the competition dataset. The team achieved better results for drone detection using YOLOv5 by only adding synthetic data.

3. Dataset generation system

3.1. The simulation component architecture

Our dataset generation system works based on Unreal Engine, Airsim, and pyphysim, as shown in Figure 1. As a simulation environment, Unreal Engine is used. It is one of the most popular game engines. It has high-quality vision, audio, and programmability features. Also, it has built-in Airsim compatibility. Airsim is developed by Microsoft and it is an open-source, cross-platform simulator for cars and drones. The dataset generation system mainly uses drone and camera functionalities of Airsim.

In addition to Unreal Engine, Airsim GUI is the main component of the proposed architecture. We developed Airsim GUI to manage the simulation environment and generate image and RF datasets for drone detection models. It contains Airsim client library to communicate with Airsim. This communication is required to send commands to Airsim and collect data from Airsim. Collected data is processed and written into a file system to create a dataset. Another component in the Airsim GUI is pyphysim library. It is a Python library that provides simulation for digital communication physical layer. It is used to generate RF datasets.

Figure 1. The simulation component architecture.

3.2. The simulation environment: Boğaziçi University north campus

Some parts of the Boğaziçi University north campus are selected as a simulation environment. The selected simulation environment covers the library, computer engineering building, basketball court, portal statue, the pyramid, and the Atatürk bust. This area covers approximately 10300 m². Figure 2 shows the selected area's map image.

The simulation environment was created through two main steps: creating individual buildings and creating the complete environment. In the first step, chosen buildings were created for the simulation environment. In the second step, these created buildings are put in the simulation environment.

In the process of creating individual buildings, the DJI Mavic Air 2 drone was used to capture aerial pictures of the selected buildings. All areas of the buildings were scanned by drone, combined in Blender, and created in a 1:1 scale for simulation. The buildings created in the first step were exported as material to import Unreal Engine. They were combined in Unreal Engine and the selected simulation environment was created with a 1:1 scale. Figure 3 shows images from the simulation environment.

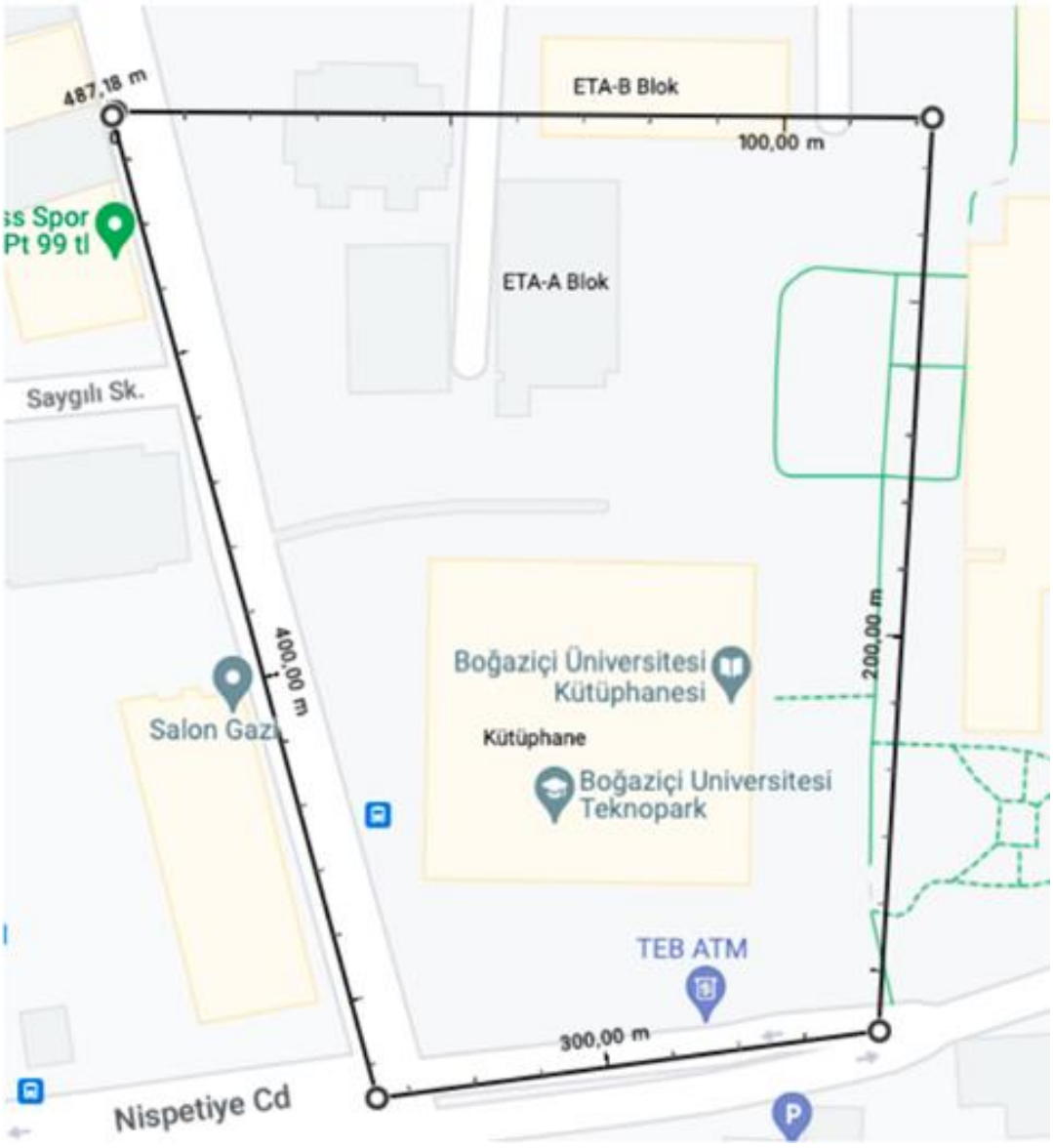

Figure 2. Map image of the selected simulation environment.

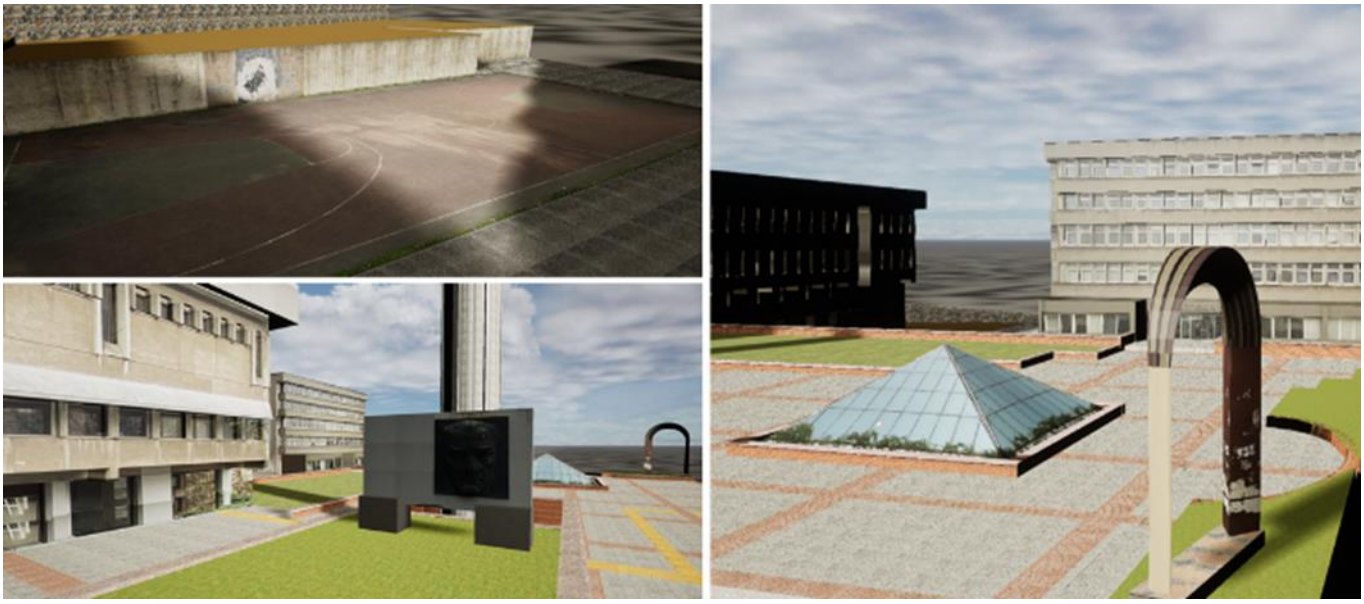

Figure 3. Screenshot images from the simulation environment. Top left image: The basketball court. Bottom left image: The Atatürk bust. Right image: The campus view from the Northeast.

3.3. Airsim GUI

For drone detection dataset generation, we created a GUI that communicates with the Airsim plugin in the Unreal Engine. We named this GUI as the Airsim GUI. This GUI was created with PyQt, a cross-platform GUI designer tool for Python. Communication between Airsim GUI and Airsim is performed by using the Airsim client library for Python. The Airsim client library has several APIs for several tasks.

There are several functions for managing the simulation environment and generating datasets in the Airsim GUI. Similar functions are put in the same tab to separate and group functionalities. Until now, six different tabs have been created: guide, drones, RF parameters, set drone path, recording, and offline data generation tabs.

- **Guide:** Directives are given in the guide tab. Also, it has an additional functionality to add a table component in a simulation environment. This table component provides locating multiple cameras with the exact relative location between them using only a single location and rotation information.
- **Drones:** The drones tab has spawning and retrieving drone functionalities. When spawning a drone, the drone's name and Cartesian coordinates for spawn location are taken as input. For retrieving a drone, the drone's name is selected from the list.
- **RF parameters:** RF parameters tab contains RF dataset-related inputs. If an RF dataset needs to be created, fields in this tab must be filled. These fields are mainly related to transmitter, channel, and SDRs. This tab will be described in Section 3.4 in detail.
- **Set drone path:** In the set drone path tab, drone paths can be defined. When creating a dataset, drones can follow these defined paths autonomously. There is a functionality to save and load created drone paths for generating another dataset with the same drone paths.
- **Recording:** The recording tab encompasses features for managing the recording process to generate datasets. Initially, a user chooses the cameras and their formats for recording. Then, the user can select or deselect the checkboxes for the recording functionalities. These checkboxes include "Save segmented images", "Follow path while recording", and "Generate RF after recording". By selecting the "Save segmented images" option, the drone segmentation can be recorded along with the camera images. These segmented images can later be used as a ground truth for drone detection models. Enabling the "Follow path while recording" ensures that the drones will follow the specified paths in the set drone path tab once the "Start Recording" button is clicked. If the user selects the "Generate RF after recording" option, the RF data generation process will start with the configured RF parameters after the simulation scenario finishes in the Unreal Engine. Once the image and RF datasets are generated, three main folders are created: "vehicle_logs", "camera_logs," and "rf_logs". The "vehicle_logs" folder contains individual files for each drone, named according to their drone names. These files contain logs of the drone's location, speed, and timestamps. The "camera_logs" folder includes separate files for each camera, named after their camera names. These files contain the image locations in the file system and timestamps of the captured images. The images themselves are stored in folders named after the respective cameras. The "rf_logs" folder stores the collected RF data from SDRs and the used RF parameters.
- **Offline data generation:** The final tab is the offline data generation. Since the RF generation step takes a long time, this process could be run later. Moreover, RF generation can be done multiple times with different parameters for the same scenario. This tab was created to generate RF datasets without running the same scenarios in Unreal Engine. Example images from the Airsim GUI are shown in Figure 4.

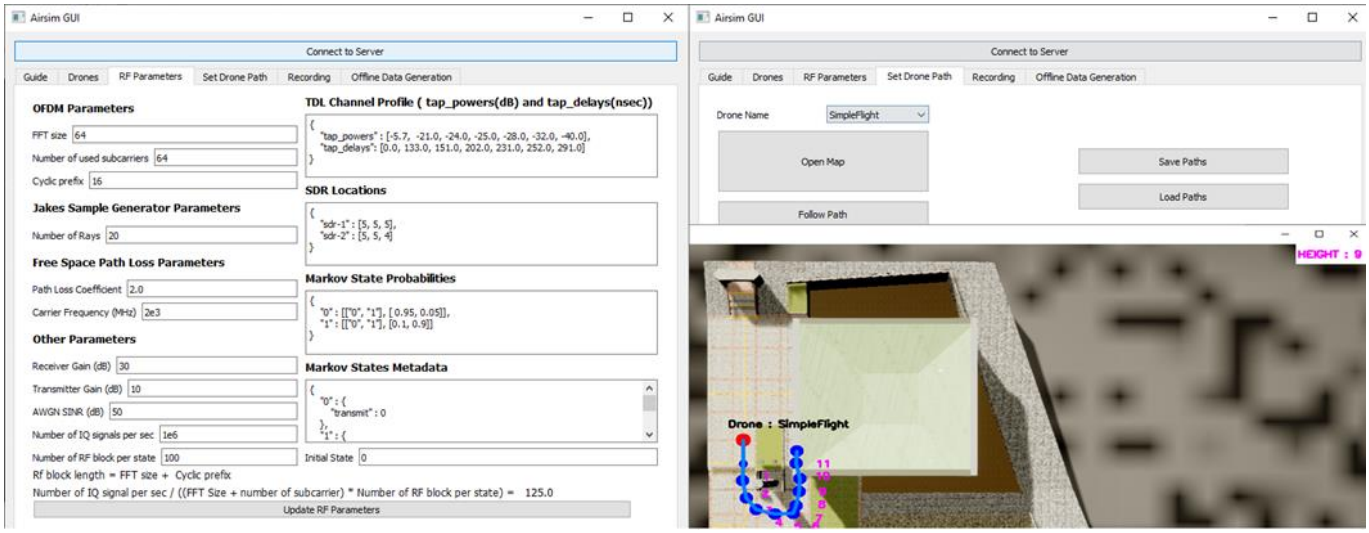

Figure 4. Example images from the Airsim GUI. (Left: RF parameters tab. Right:Set drone path tab.)

3.4. RF dataset generation

Figure 5 shows high-level RF dataset generation steps. RF dataset is generated with created files in the "vehicle_logs" folder. There is a file for every single drone which contains location and speed logs with timestamps. Until the AWGN step, RF data is generated separately for drones.

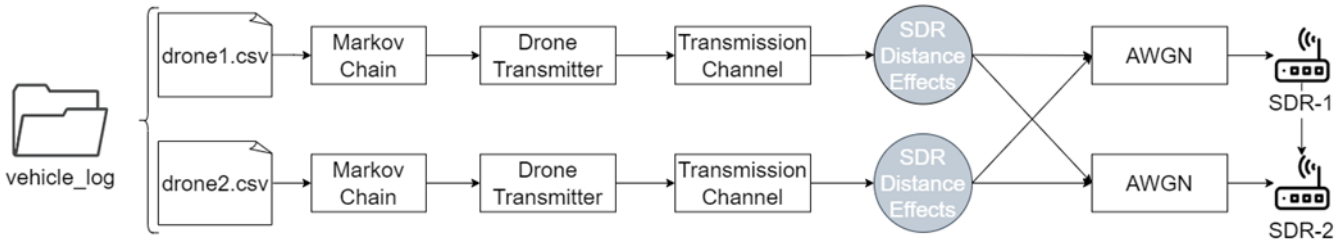

Figure 5. High-level RF dataset generation steps.

The pyphysim library was mainly used to generate RF dataset. The created RF datasets refer to collected IQ signals by SDRs in the simulation environment. SDR is the main component of RF signal-based drone detection systems. The RF dataset generation architecture supports multiple drones and SDRs. For this version of our data generation system, drone controllers are excluded since they are assumed to be out of coverage, as seen in Figure 6. Because of that, only drone-transmitted data is generated. SDR, drone transmitters, and other RF simulation configurations are set in the RF parameters tab of Airsim GUI. Then, the RF dataset can be created directly after recording or using previously created datasets with the offline data generation tab.

RF data generation steps start from reading drone log files, and at the end RF log files are created for every SDR. First of all, drone logs are read line by line. Between two lines, the time difference and the number of IQ signals that need to be generated between them is calculated. However, drones do not generate RF signals every time. To handle that Markov chain is used. Users can create a Markov chain to define idle (not transmitting) and busy (transmitting) states and probabilities of moving other states. Also, if the state represents busy, it contains transmitting power and one of the supported modulation techniques (Binary phase shift keying (BPSK) or quadrature phase shift keying (QPSK)). There is no limit to the created number of states. Figure 7 shows a Markov chain example containing data and probabilities of moving other states. A user defines the number of IQ signals in a single state from the Airsim GUI, and the number of generated states between two consecutive logs is calculated based on it. If the state is idle, there is no other process until the AWGN step.

If the state is busy, data transmission from a drone to SDR happens. The first block in this process is the drone transmitter block. It contains the processes that happen in the drone transmitter. First, the random number generator is run based on the modulation technique. If the modulation technique is BPSK, the generated numbers are zero and one. If the modulation technique is QPSK, the numbers range from zero to four. These generated numbers represent symbols. These represented symbols are transformed into complex signals in the IQ plane. Then, OFDM is applied to the signal with the FFT size, cyclic prefix size, and the number of subcarrier parameters taken from Airsim GUI. Finally, transmitting power and transmitter gain are applied to the signal transmitted to the transmission channel.

The transmission channel mainly contains two models: Jakes' channel model and tapped delay line model. In the transmission channel, only purely stochastic channel modeling is used. Jakes' model is used because it is famous for modeling mobile wireless communication systems realistic, contains effects of

scatters, and has a Doppler effect. Combining Jakes' model with the tapped delay line model provides a defining multipath tap delays and powers as an input.

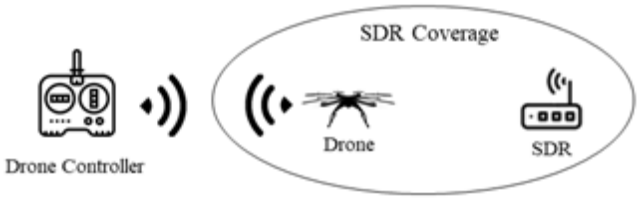

Figure 6. The SDR coverage in our simulation system shows that drone controllers are always out of coverage.

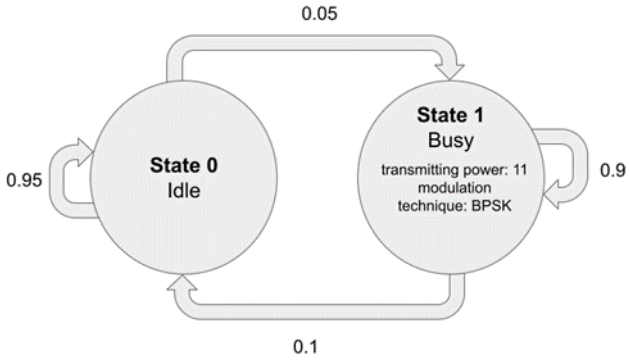

Figure 7. There are two states named idle and busy. In a busy state, transmission happens with 11 Watt and the BPSK modulation technique. Numbers on the arrows show the transition probabilities.

Because there is multiple SDR support, the same distance-related channel effects cannot be applied for every drone-SDR pair with a single calculation. So, for every SDR, these effects are applied separately for a drone transmission. These effects are path loss and phase shift. The free space path loss model is used as a path loss model. This model uses distance, carrier frequency, and path loss coefficient. Distance between a drone and SDR is calculated between two consecutive logs, and carrier frequency and path loss coefficient are taken from a user. Phase shift is applied based on the difference between the distances of drones and SDRs. After that, all drone signals generated simultaneously are summed together for SDR receivers.

At the end, for every SDR receiver, AWGN is added, and the receiver gain is applied. The generated signals are written as a byte stream to the "rf_logs" folder. Every SDR writes its data into a file with the name of the SDR. In addition to generated signals, the RF parameters used in dataset generation are also written in the "rf_parameters.pickle" file.

4. Example use cases

A dataset was created using the implemented simulation environment and the Airsim GUI to show example use cases. Different drone path scenarios, camera and SDR locations, and lighting conditions were chosen, and a dataset with image and RF was created with these configurations. In the simulation environment, two cameras, two SDRs, a drone, and ten birds were used. Between cameras and SDRs, there is 10 m. We proposed three different use cases: drone detection via image, drone detection via RF, and drone distance detection via RF.

The same hardware and software configurations were used in use cases as an 8GB Nvidia GTX 1070 graphic card, PyTorch 1.13.1 as the deep learning framework, Python version 3.8, CUDA version 11.0, and the operating system Windows 10.

4.1. Drone detection via image

Drone detection via image problem is a subset of object detection in computer vision. There are several object detection algorithms in the literature. YOLO is one of the most popular of them. It has different versions, and the state-of-the-art version is the YOLOv8 model. This model contains different sizes based on the number of parameters: nano(n), small(s), middle(m), large(l), and extra-large(x). Because the detection speed is crucial, achieving higher detection capability in smaller models is required. So, the nano version of YOLOv8 was chosen for drone detection.

Before training the model, the created dataset was prepared for the model. For ground truth, bounding boxes are required. However, Airsim GUI saves images with segmentation for ground truth. Because of that, segmented ground truths were transformed into bounding boxes. Then, the dataset was divided into training, validation, and test datasets as 60%-20%-20%, respectively.

The transfer learning was used for the training with a pre-trained model on the COCO dataset [34]. The model was trained to detect only drones. 1024 x 576 pixel image format was used as an input of the model. After 14 epochs, the training process was finished. The training metrics are given in Table 2.

Table 2. YOLOv8 nano model training process via image.

Epoch	Precision	Recall	mAP50	mAP50-95
1	0.76755	0.53322	0.55266	0.2614
2	0.61498	0.51329	0.53621	0.27792
3	0.74676	0.48494	0.54547	0.27878
4	0.80832	0.5604	0.61579	0.35433
5	0.68501	0.52492	0.5474	0.29289
6	0.80637	0.62261	0.67955	0.40068
7	0.85189	0.64014	0.68377	0.40726
8	0.82561	0.65275	0.72545	0.44725
9	0.84713	0.65615	0.70576	0.41265
10	0.91571	0.59552	0.71363	0.4251
11	0.87754	0.6711	0.73256	0.4228
12	0.85971	0.62292	0.707	0.41085
13	0.8656	0.63953	0.71531	0.44009
14	0.90008	0.65947	0.73839	0.45815

Used metrics in Table 2 are precision, recall, and mean average precision (mAP) scores. The mean average precision score is the primary performance evaluation metric for object detection tasks. The mAP50 is calculated by computing the average precision (AP) at a 50% intersection over union (IoU) threshold. The mAP50-95 scores are calculated by taking the average of precision values across a range of IoU thresholds starting from 0.5 and ending at 0.95, with a step size of 0.05. This calculation is performed as in Equation 1. The average precision is calculated based on the precision-recall

curve that corresponds to a specific IoU threshold. Equation 2 and 3 provide the formulas for precision and recall, in terms of False Negative (FN), True Positive (TP), and False Positive (FP). The IoU score is calculated by calculating the ratio of the intersection areas between the two bounding boxes and the area of their union, as shown in Equation 4.

$$mAP50-95 = \frac{1}{11}\sum_{t=0.5}^{0.95} mAPt \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union} \quad (4)$$

4.2. Drone detection via RF

As in Section 4.1, YOLOv8 was used for drone detection via RF. The RF data was transformed into images because YOLO was developed for vision tasks. First of all, SDR data is split into 0.25-second sequences. These sequences contain IQ signals. For every sequence, a short-time Fourier transform (STFT) is calculated. Because there were two different SDR in the simulation environment, both STFT results are concatenated vertically, as shown in Figure 8. In the created dataset, there are only cases that have drones. Because of that, cases that do not contain drones were generated using Gaussian noise, and the same processes were applied to these signals. All datasets are labeled with drone and background noise signals. Then, the dataset is split into training, validation, and test datasets as 60%-20%-20%, respectively.

The nano version of YOLOv8 is used for the classification task for this problem. Rather than an object detection task, the classification task classifies the whole image. Again, transfer learning was used for this model's training, and the pre-trained model with the ImageNet dataset [35] was used. After the first epoch, the accuracy metric remained relatively consistent, as indicated in Table 3. The third epoch provides the most accurate calculation; the corresponding accuracy is 0.92531. The accuracy indicates how often the network predicts the correct label with the highest probability.

Table 3. YOLOv8 nano model training process for drone detection via RF.

Epoch	Accuracy
1	0.92304
2	0.92485
3	0.92531
4	0.92531

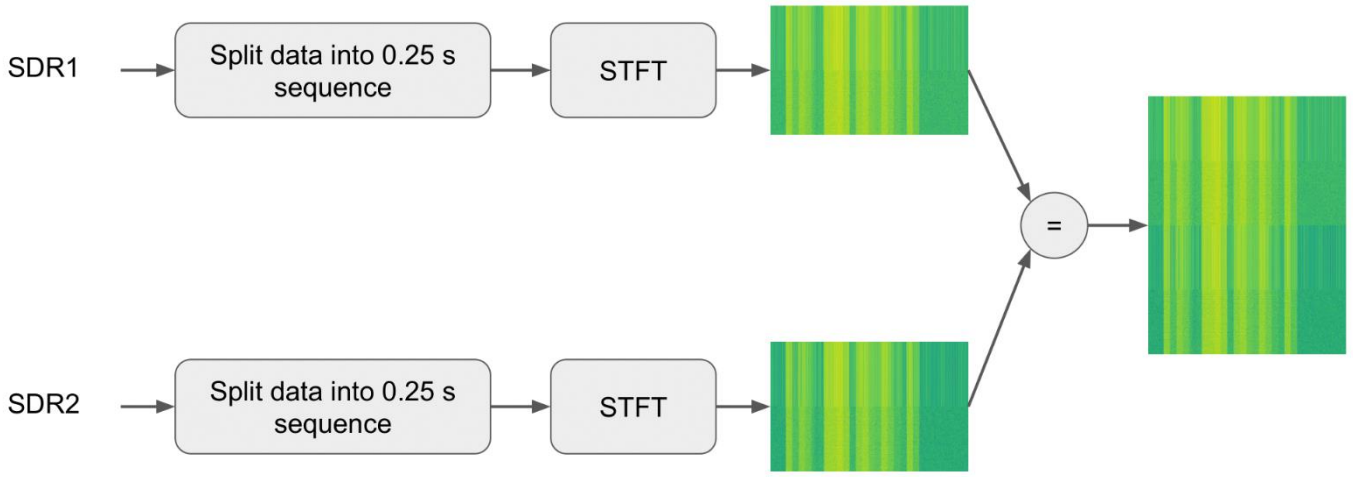

Figure 8. Data preprocess for RF data.

4.3. Drone distance detection via RF

In the final use case, drone distance detection via RF dataset problem was chosen. For this task, a custom convolutional neural network (CNN) was built to predict the distance between drones and SDRs. Because there are two SDRs in the simulation environment, the middle of the SDRs is selected as a SDR location. Two different sequence lengths, 0.25 seconds and 0.50 seconds, are used to compare performances.

For data preprocessing, the SDR signals were divided into 0.25-second and 0.50-second sequences and labeled with the average distance in the sequences. The sampling frequency was 10^6 in the dataset. This means that each 0.25-second sequence contains 250,000 time points, and a 0.50-second sequence contains 500,000 time points. I and Q components are split for both SDRs and merged vertically to provide input for the CNN model. 4x250000-shaped and 4x500000-shaped arrays were created as input for the custom CNN models. The datasets for different sequence lengths were divided into training, validation, and test datasets as 60%-20%-20%, respectively.

The proposed CNN models are shown in Figure 9. The proposed models contain three 2D convolution layers and two 2D average pooling layers. To prevent

overfitting, two dropout layers with a 20% probability are added. After the convolution layers, fully connected layers were added to the end. Because 0.50 second sequences contain more data, one more dense layer is added to its model.

Both models are trained with their datasets separately with the same hyperparameters. For the training, the batch size is chosen as 10. Rectified linear units are used as an activation layer. To converge the optimal solution, Adam optimizer is used, and the learning rate is chosen as 0.001. The best results for 0.25 second sequence were achieved in the 8th epoch and 0.50 second sequence were achieved in the 10th. The test result statistics are given in Table 4. In the given statistics, the errors in min 10% and max 10% were trimmed. The root mean square error for the test dataset is 7.29 meters for 0.25 second sequence and 6.07 meters for 0.50 second sequence. As a result, broader sequences provide better performance for the drone distance detection task.

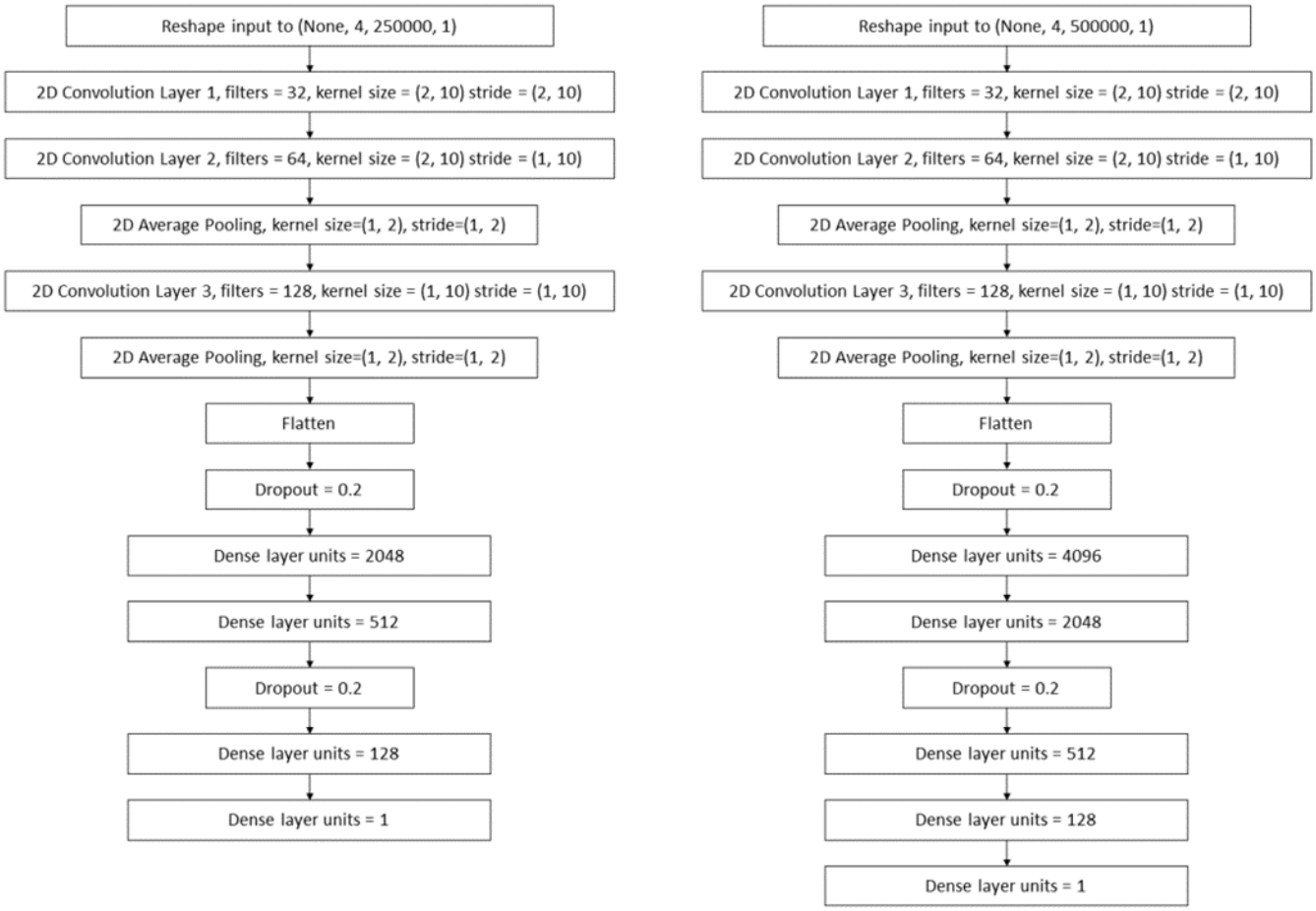

Figure 9. Custom CNN models to predict the distance. (Left is for 0.25 second and right is for 0.50 second sequences).

Table 4. Drone distance detection statistics for the absolute difference and percentage error between ground truth and estimations. For statistics, we trimmed rows that are in the min 10% and max 10% percentage error.

0.25 second sequences (RMSE=7.29) (Epoch 8)	Absolute difference (m)	mean	std	min	25%	50%	75%	max
		6.26	3.73	0.93	3.16	5.70	8.54	17.98
	Percentage error (%)	mean	std	min	25%	50%	75%	max
		14.82	7.73	3.33	8.20	14.10	20.22	34.23
0.50 second sequences (RMSE=6.07) (Epoch 10)	Absolute difference (m)	mean	std	min	25%	50%	75%	max
		5.32	2.93	0.64	2.94	4.65	7.16	14.96
	Percentage error (%)	mean	std	min	25%	50%	75%	max
		12.50	5.53	3.17	7.92	11.76	16.72	25.71

5. Conclusion

In this study, we have proposed a comprehensive solution for generating datasets for drone detection models in a simulation environment. We described the proposed architecture and created the Airsim GUI tool. Then, the implemented structure for creating RF dataset is explained. Moreover, three different drone detection models were tested on the generated dataset from the proposed architecture.

As a future work, we aim to enhance the RF dataset generation architecture in the Airsim GUI. We want to incorporate LOS and NLOS channel models. Also, we want to add dataset generation features for other modalities like acoustics and radar.

Acknowledgement

This work has been supported by Boğaziçi University BAP under the grant BAP-SUP-17862

Author contributions

Yusuf Özben: Methodology, Software, Tests, Writing-Original draft preparation. **Süleyman Emre Demir:** Methodology, Software, Tests. **Hüseyin Birkan Yılmaz:** Methodology, Visualization, Investigation, Writing-Reviewing and Editing.

Conflicts of interest

The authors declare no conflicts of interest.

References

1. Şasi, A., & Yakar, M. (2017). Photogrammetric modelling of Sakahane Masjid using an unmanned aerial vehicle. Turkish Journal of Engineering, 1(2), 82-87. https://doi.org/10.31127/tuje.316675
2. Karataş, L., Alptekin, A., Kanun, E., & Yakar, M. (2022). Tarihi kârgir yapılarda taş malzeme bozulmalarının İHA fotogrametrisi kullanarak tespiti ve belgelenmesi: Mersin Kanlıdivane ören yeri vaka çalışması. İçel Dergisi, 2(2), 41-49.
3. Karataş, L., Alptekin, A., Karabacak, A., Yakar, M. (2022). Detection and documentation of stone material deterioration in historical masonry buildings using UAV photogrammetry: A case study of Mersin Sarisih Inn. Mersin Photogrammetry Journal, 4(2), 53-61. https://doi.org/10.53093/mephoj.1198605
4. Şasi, A., & Yakar, M. (2018). Photogrammetric modelling of Hasbey Dar'ülhuffaz (Masjid) using an unmanned aerial vehicle. International Journal of Engineering and Geosciences, 3(1), 6-11. https://doi.org/10.26833/ijeg.328919
5. Çolak, A., Aktan, N., & Yılmaz, H. M. (2022). Modelling of its surroundings and Selime Cadhetral by UAV data. Advanced UAV, 2(1), 24-28.
6. Banafaa, M., Pepeoğlu, Ö., Shayea, I., Alhammadi, A., Shamsan, Z., Razaz, M. A., ... & Al-Sowayan, S. (2024). A comprehensive survey on 5G-and-beyond networks with UAVs: Applications, emerging technologies, regulatory aspects, research trends and challenges. IEEE Access, 12, 7786 – 7826. https://doi.org/10.1109/ACCESS.2023.3349208
7. Jiang, N., Wang, K., Peng, X., Yu, X., Wang, Q., Xing, J., ... & Han, Z. (2021). Anti-uav: a large-scale benchmark for vision-based uav tracking. IEEE Transactions on Multimedia, 25, 486-500. https://doi.org/10.1109/TMM.2021.3128047
8. Zhao, J., Wang, G., Li, J., Jin, L., Fan, N., Wang, M., ... & Guo, Y. (2021). The 2nd anti-uav workshop & challenge: Methods and results. Computer Vision and Pattern Recognition. https://doi.org/10.48550/arXiv.2108.09909
9. Zhao, J., Li, J., Jin, L., Chu, J., Zhang, Z., Wang, J., ... & Shengmei, J. S. (2023). The 3rd anti-uav workshop & challenge: Methods and results. Computer Vision and Pattern Recognition. https://doi.org/10.48550/arXiv.2305.07290
10. Coluccia, A., Ghenescu, M., Piatrik, T., De Cubber, G., Schumann, A., Sommer, L., Klatte, J., Schuchert, T., Beyerer, J., Farhadi, M., Amandi, R., Aker, C., Kalkan, S., Saqib, M., Sharma, N., Daud, S., Makkah, K., & Blumenstein, M. (2017). Drone-vs-bird detection challenge at IEEE AVSS2017. IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS) Drone-vs-Bird detection challenge, 1-6, Lecce, Italy.
11. Coluccia, A., Fascista, A., Schumann, A., Sommer, L., Dimou, A., Zarpalas, D., ... & Mantegh, I. (2017). Drone-vs-bird detection challenge at IEEE AVSS2017. 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 1-8. https://doi.org/ 10.1109/AVSS.2017.8078464
12. Coluccia, A., Fascista, A., Schumann, A., Sommer, L., Dimou, A., Zarpalas, D., ... & Rajashekar, S. (2021). Drone vs. bird detection: Deep learning algorithms and results from a grand challenge. Sensors, 21(8), 2824. https://doi.org/10.3390/s21082824
13. Coluccia, A., Fascista, A., Schumann, A., Sommer, L., Dimou, A., Zarpalas, D., ... & Mantegh, I. (2021). Drone-vs-bird detection challenge at IEEE AVSS2021. 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 1-8. https://doi.org/10.1109/AVSS52988.2021.9663844
14. Coluccia, A., Fascista, A., Schumann, A., Sommer, L., Dimou, A., Zarpalas, D., ... & Pavleski, D. (2022). Drone-vs-bird detection challenge at ICIAP 2021. In International Conference on Image Analysis and Processing, 410-421. https://doi.org/10.1007/978-3-031-13324-4_35
15. Li, J., Murray, J., Ismaili, D., Schindler, K., & Albl, C. (2020). Reconstruction of 3D flight trajectories from ad-hoc camera networks. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1621-1628. https://doi.org/10.1109/IROS45743.2020.9341479
16. Allahham, M. S., Al-Sa'd, M. F., Al-Ali, A., Mohamed, A., Khattab, T., & Erbad, A. (2019). DroneRF dataset: A dataset of drones for RF-based detection, classification and identification. Data in Brief, 26, 104313. https://doi.org/10.1016/j.dib.2019.104313
17. Swinney, C. J., & Woods, J. C. (2021). RF detection and classification of unmanned aerial vehicles in environments with wireless interference. International Conference on Unmanned Aircraft Systems (ICUAS), 1494-1498. https://doi.org/10.1109/ICUAS51884.2021.9476867
18. Glüge, S., Nyfeler, M., Ramagnano, N., Horn, C., & Schüpbach, C. (2023). Robust drone detection and classification from radio frequency signals using convolutional neural networks. 15th International Joint Conference on Computational Intelligence (IJCCI), 496-504. https://doi.org/10.5220/0012176800003595
19. Medaiyese, O., Ezuma, M., Lauf, A., & Adeniran, A. (2022). Cardinal RF (CardRF): An outdoor UAV/UAS/drone RF signals with Bluetooth and WiFi signals dataset.
20. Al-Sa'd, M. F., Al-Ali, A., Mohamed, A., Khattab, T., & Erbad, A. (2019). RF-based drone detection and

identification using deep learning approaches: An initiative towards a large open source drone database. Future Generation Computer Systems, 100, 86-97. https://doi.org/10.1016/j.future.2019.05.007
21. Al-Emadi, S., & Al-Senaid, F. (2020). Drone detection approach based on radio-frequency using convolutional neural network. 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), 29-34. 10.1109/ICIoT48696.2020.9089489
22. Allahham, M. S., Khattab, T., & Mohamed, A. (2020). Deep learning for RF-based drone detection and identification: A multi-channel 1-D convolutional neural networks approach. IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), 112-117. https://doi.org/10.1109/ICIoT48696.2020.9089657
23. Medaiyese, O. O., Syed, A., & Lauf, A. P. (2021). Machine learning framework for RF-based drone detection and identification system. 2nd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS), 58-64. https://doi.org/10.1109/ICON-SONICS53103.2021.9617168
24. Nemer, I., Sheltami, T., Ahmad, I., Yasar, A. U. H., & Abdeen, M. A. (2021). RF-based UAV detection and identification using hierarchical learning approach. Sensors, 21(6), 1947. https://doi.org/10.3390/s21061947
25. Zhao, Z., Du, Q., Yao, X., Lu, L., & Zhang, S. (2023). A Two-Dimensional Deep Network for RF-based Drone Detection and Identification Towards Secure Coverage Extension. In 2023 IEEE 98th Vehicular Technology Conference (VTC2023-Fall), 1-5. https://doi.org/10.1109/VTC2023-Fall60731.2023.10333485
26. Nguyen, P., Kim, T., Miao, J., Hesselius, D., Kenneally, E., Massey, D., ... & Vu, T. (2019). Towards RF-based localization of a drone and its controller. Proceedings of the 5th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications, 21-26. https://doi.org/10.1145/3325421.3329766
27. Unlu, E., Zenou, E., Riviere, N., & 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, 11, 1-13. https://doi.org/10.1186/s41074-019-0059-x
28. Singha, S., & Aydin, B. (2021). Automated drone detection using YOLOv4. Drones, 5(3), 95. https://doi.org/10.3390/drones5030095
29. Zhai, X., Huang, Z., Li, T., Liu, H., & Wang, S. (2023). YOLO-Drone: an optimized YOLOv8 network for tiny UAV object detection. Electronics, 12(17), 3664. https://doi.org/10.3390/electronics12173664
30. Seidaliyeva, U., Akhmetov, D., Ilipbayeva, L., & Matson, E. T. (2020). Real-time and accurate drone detection in a video with a static background. Sensors, 20(14), 3856. https://doi.org/10.3390/s20143856
31. Carrio, A., Vemprala, S., Ripoll, A., Saripalli, S., & Campoy, P. (2018). Drone detection using depth maps. In 2018 IEEE/RSJ international conference on intelligent robots and systems (IROS), 1034-1037. https://doi.org/10.1109/IROS.2018.8593405
32. Carrio, A., Tordesillas, J., Vemprala, S., Saripalli, S., Campoy, P., & How, J. P. (2020). Onboard detection and localization of drones using depth maps. IEEE Access, 8, 30480-30490. https://doi.org/10.1109/ACCESS.2020.2971938
33. Akyon, F. C., Eryuksel, O., Ozfuttu, K. A., & Altinuc, S. O. (2021). Track boosting and synthetic data aided drone detection. 17th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 1-5. https://doi.org/10.1109/AVSS52988.2021.9663759
34. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. Computer Vision–ECCV 2014: 13th European Conference, Zurich, 13, 740-755. https://doi.org/10.1007/978-3-319-10602-1_48
35. Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, 248-255. https://doi.org/10.1109/CVPR.2009.5206848