A simulated dataset in aerial images using Simulink for object detection and recognition

Mittal, Payal ; Sharma, Akashdeep ; Singh, Raman

Published in:
International Journal of Cognitive Computing in Engineering

DOI:
10.1016/j.ijcce.2022.07.001

Published: 22/07/2022

Document Version
Publisher's PDF, also known as Version of record

International Journal of Cognitive Computing in Engineering 3 (2022) 144–151

Contents lists available at ScienceDirect

International Journal of Cognitive Computing in Engineering

journal homepage: https://www.keaipublishing.com/en/journals/international-journal-of-cognitive-computing-in-engineering/

A Simulated Dataset in Aerial Images using Simulink for Object Detection and Recognition

Payal Mittal [a,1], Akashdeep Sharma [a,2,*], Raman Singh [b,3]

[a] University Institute of Engineering and Technology, Panjab University, Chandigarh, India
[b] University of the West of Scotland, United Kingdom

ARTICLE INFO

Keywords:
UAV Simulator
Object detection
Simulated dataset
Computer vision
Deep learning

ABSTRACT

The understanding and implementation of object detection and classification algorithms help in deploying diverse applications of UAVs. There is a need for a simulated UAV dataset to incorporate a pipeline for various algorithms. To reduce human efforts, multiple simulators have been utilized to mimic the real-time behavior of drones. Our work inspired simulators and can be considered by engineering students to create a dataset in a simulated environment. In this paper, we focused on the study of MATLAB-based Simulink through multiple environment settings. The core objective of the paper is to create a simulated dataset from the utilized quadcopter-based flight control model in MATLAB-based Simulink. In this customized model, few modifications have been made to obtain drone videos to detect object categories such as pedestrians, other drones and obstacles while navigating in a simulated environment. Additionally, these simulated images are annotated for aerial image interpretation with multiple object categories. The dataset is annotated and is freely downloadable from: https://bit.ly/38jlAsh. In this research study, we mainly focus on the process of drone simulation in the MATLAB-based Simulink model. Further, the captured dataset is verified on state-of-the-art object detectors such as YoloV3, TinyYolov3 etc. by evaluating the authenticity of the dataset.

1. Introduction

Unmanned Aerial Vehicles (UAVs) commonly known as drones are autonomous and meant to be hovering around a working area to gather information for the deployment of computer vision-related applications (McNeal, 2014). The applications of UAVs include human crowd detection using Convolutional Neural Networks (CNN) (Tzelepi & Tefas, 2017); crop classifications (George, Tiwari, Yadav, Peters & Sadana, 2013); wildlife conservation (Bondi, Dey, Kapoor, Piavis, Shah, Fang & Tambe, 2018); traffic monitoring through detection of vehicles (Zhang, Cao & Mao, 2017); drone surveillance system for violent human actions identification (Singh, Patil & Omkar, 2018); breach detection and mitigation (Shijith, Poornachandran, Sujadevi & Dharmana, 2017); pedestrian detections (Ma, Wu, Yu, Xu & Wang, 2016); and identification of mosquito breeding areas (Amarasinghe, Suduwella, Elvitigala, Niroshan, Amaraweera, Gunawardana & Keppetiyagama 2017). The real-time hardware cost of UAVs is expensive and accessing them requires high skilled training so it is not advisable to work on real-time UAV data in initial scenarios. The utilization of aerial vehicles by naïve users' for gathering data is potentially unsafe. As a promising solution, efficient UAV-based simulators reduce the safety risks of maneuvering UAVs while monitoring important situations such as inspecting critical infrastructure of disaster-affected spaces. The simulator capabilities facilitate the fast acquisition of high-quality aerial data to promote valuable insights to industries and promote research for developing computing paradigms for disaster emergency response stricken areas. The presented world-class simulators in this research work offer a great benefit to research society through enabling real-time environments for testing purposes. The applications of UAV-based simulators are diverse in nature by providing navigation services (Chen, Zhou, Yang, Chen, Li & Wen, 2022), traffic surveillance (Bashir, Boudjit & Zeadally, 2022) and risk assessment (Trepekli, Balstrøm, Friborg, Fog, Allotey, Kofie & Møller-Jensen, 2022; Utlu, ÖZTÜRK & Şimşek, 2021). In this research study, we explored Simulink, an add-on product in MATLAB which provides a visual editor, customizable libraries, and solvers for simulating dynamic models. It is developed by MathWorks, which performs various functions such as matrix manipulations, implementation of algorithms, and creation of user interfaces. The aerospace blockset of MATLAB enables us to incorporate vision-based algorithms into models and export results for further computations (Dabney & Harman, 2004). A project named Quadcopter in Simulink is utilized for capturing the simulated data which mimics the behavior of a real drone. The simulated dataset

* Corresponding Author: Telephone Number: +91-9814925790
E-mail addresses: payalmittal6792@gmail.com (P. Mittal), akashdeep@pu.ac.in (A. Sharma), raman.singh@uws.ac.uk (R. Singh).
[1] Research Scholar
[2] Assistant Professor
[3] Lecturer in Cyber Security

https://doi.org/10.1016/j.ijcce.2022.07.001
Received 20 August 2021; Received in revised form 1 July 2022; Accepted 14 July 2022
Available online 16 July 2022

with single and multiple instances of UAV objects is captured from this simulated model.

1.1. Motivation for study

Our study is focused on the need to analyze several UAV simulators and provide a simulated UAV dataset for young researchers to perform computer vision-related tasks. The efficient UAV-based simulators cannot harm human privacy and disturb airspace while reducing the safety risks of maneuvering UAVs. The proposed study aims to provide a simulated UAV dataset for further identification of aerial objects such as pedestrians, obstacles and other drones while navigating in the airspace. Further, verified annotations for captured simulated UAV dataset are provided and validation through applying arbitrary classification and detection algorithms is also presented. The main objectives of this paper include:

1. A simulated dataset of annotated tested images is generated for further utilization for object detection purposes.
2. A comparison of UAV-based network simulators for flight navigation.
3. Exploring MATLAB-based Simulink for capturing dataset.
4. Customized Simulink environmental settings for acquiring simulated data belonging to multiple categories such as another drone, pedestrians and obstacles.

The paper's organization is as follows: Section 2 discusses the related work for the process of simulation and a brief history of available simulators is given. The methodology process for achieving a simulated dataset from Simulink for UAV objects is proposed in Section 3. Further, the sub-sections describe the different experiments through which customized settings can be done in the Simulink environment for the proposed dataset. The simulated UAV dataset details are listed in Section 4. Moreover, the comparison and validation of the proposed dataset is mentioned in Section 4. The final section summarizes the performance of the proposed simulated dataset on the standard object detection algorithms.

2. Related work

The generic process of the simulation consists of designing models related to theoretical, mathematical or physical systems computed on a computer machine. As a consequence, the development of a simulated model provides a robust technical contribution to the UAVs community's multitude of related applications. For example, the civilian air force of different countries is interested in developing a simulated model that emulates UAV behaviour during flight navigation, using a powerful simulator. More examples of simulators-based studies are presented in (Álvares, Silva & Magaia, 2021; Xiao, Ma, Tan, Cong & Wang, 2022; Ning, Dong, Wen, Wang, Guo, Kwok & Poor, 2021; Yaoming, Yu, Anhuan & Lingyu, 2021; Causa & Fasano, 2021; Lyu, Cao, Yuan & Xie, 2021; Choi, Choi, Kim, Lee, Seo & Jun, 2021; Udroiu, Deaconu & Nanau, 2021; Matlekovic, Juric & Schneider-Kamp, 2022). The process of simulation helps users to initially access the working environment of drones and evaluates the performance of complex algorithms. There exist some powerful UAV simulatorwhich are described below:

Gazebo- Gazebo, a three-dimensional dynamic the simulator offers a high degree of fidelity to users and accurately simulates robotics in indoor and outdoor complex environments. The typical applications of Gazebo include regression testing with the real environment, designing and testing robotics algorithms. It is an open-source simulator, customized for its own algorithms. The system requirements of Gazebo rely on the configuration version but are currently best used with Ubuntu, a version of Linux. This simulator majorly supports plugins, a software component that provides additional functionality to existing software, can analyze Robot Operating System (ROS) messages and calls for sensor-related input and output. The multiple plugins available in Gazebo are camera, Inertial Measurement Unit (IMU) sensor, video, planar move and template for users who want to write their own plugin (Koenig & Howard, 2004).

AirSim- Aerial Informatics and Robotics Simulation (AirSim) simulator used innovative deep learning knowledge to extract knowledge to evaluate the performance of different autonomous vehicles [4]. In UAVs, only quadcopter models are implemented in AirSim but users can add sensors and robot models for their use. There exist three different views to the observer such as view of real-time depth, drone's and object segmentation with the viewpoint of the ground observer, fly with me and First Person View (FPV) mode. The majority code of AirSim is stored in AirSim Library (AirLib), a self-contained library that can be compiled with a C++ compiler and composed of a physics engine, vehicle model, control library and sensor model. AirSim uses the directions of +X i.e. North, +Y i.e. East and +Z i.e. Down which is the Ned Coordinate System and the initial point for the vehicle is (0,0,0) in the North, East and Down (NED) system. The objective of the AirSim simulator is to provide an artificial intelligence-based research paradigm for simulating the behavior of autonomous vehicles. Further, a few popular simulators such as YSFlight, AirSim and Gazebo are displayed in Fig. 1.

YSFlight - The simulator YSFlight is platform-independent, customized environment and has user-friendly interface. The user can choose their own choice of aircrafts, and military hummers for flying purposes. It is completely free, easy to use, has aircraft choices, has a vibrant community with many virtual groups, and events, and has been in development since 1999. YSFlight has had over 1,000,000 downloads in this time. The system requirements include OS which is XP/7/8, or Linux, or Mac OS X, Operating Speed 2.0 GHz or faster, equal or higher than NVIDIA GeForce 520m / ATI Radeon 8000 GPU, 1GB or larger RAM, 5GB or larger hard disk capacity. YSFlight is a freeware flight simulation game for Windows, OS X and Linux-based operating systems such as Ubuntu (Nisansala, Weerasinghe, Dias, Sandaruwan, Keppitiyagama, Kodikara & Samarasinghe, 2015).

Flight Gear - This multiplayer UAV simulator provides information on flight and air traffic control simulation. It has multi-GPU, and multi-screen support and can add new dynamic flight models in its interface. Using Flight Gear v.2.0, several trajectories and different runways were set up for computing tests on the models which provides an opportunity to validate robustness, especially in cluttered scenes with multiple pathways (Perry, 2004). The captured images are rendered in Flight Gear version 2.0.0 with the rendering options such as the resolution of 1024×768, 55° field-of-view, anti-aliasing based 8X full-screen and 16X anisotropic filtering.

Avionics - The cockpit is included in aircraft design for providing warning related to weather monitoring and anti-collision of UAVs in the Avionics simulator. There are several major vendors of flight avionics: Panasonic Avionics Corporation, Honeywell, and Universal Avionics System Corporation. This project aims at advancing software designs for interfaces, sensors and cockpit systems (Kayton & Fried, 1997).

X-Plane – X-Plane is a flight simulator developed by laminar research and can be used professionally or personally with macOS, Windows, and Linux, while mobile versions are also available for Android and iOS. X-Plane comes with almost the latest updateimagery of earth through which real-time performance can be evaluated. X-Plane stands out from existing simulators through an aerodynamic model which is based on the blade element theory. The effective flight simulators emulate the real-world performance by computing the empirical data in predefined lookup tables to calculate lift or drag-based aerodynamic forces, which vary with heterogeneous flight conditions (Garcia & Barnes, 2009). In the next section, a detailed description of Simulink for UAV data is provided followed by the methodology for capturing the simulated dataset. The effective flight simulators emulate the real-world performance by computing the empirical data in pre-defined lookup tables to calculate lift or drag-based aerodynamic forces, which

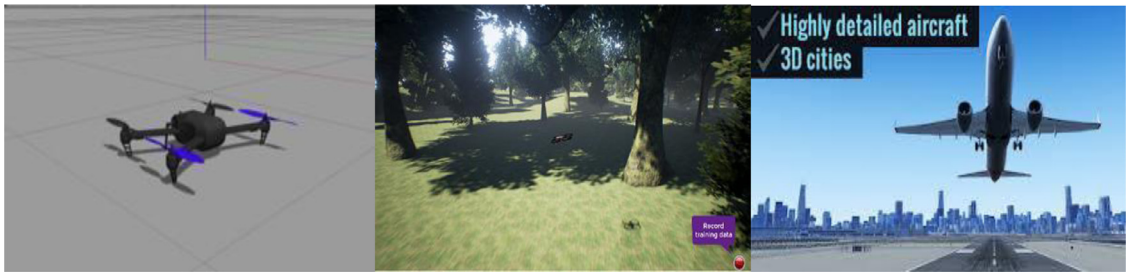


Fig. 1. Snapshots of available flight simulators a) Gazebo b) AirSim c) X-plane

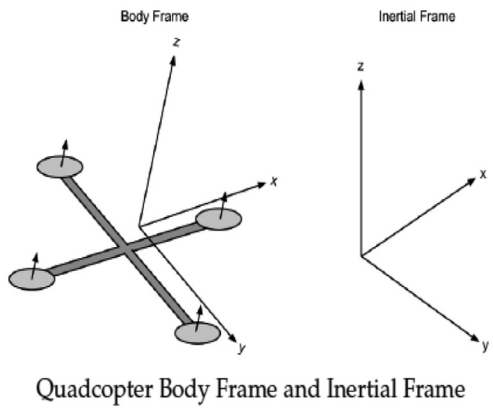
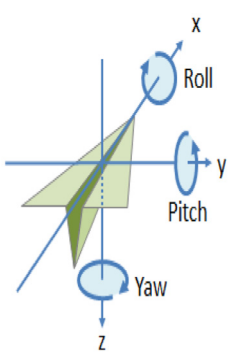


Fig. 2. Quadcopter body and inertial frame (Luukkonen, 2011)



Axis of Rotation	Euler Angle Name	Euler Angle Symbol
x	Roll	u
y	Pitch	v
z	Yaw	w

Fig. 3. Description of Euler angles in modelling of Quadcopter (Nemati & Kumar, 2014)

vary with heterogeneous flight conditions (Garcia & Barnes, 2009). In the next section, a detailed description of Simulink for UAV data is provided followed by the methodology for capturing the simulated dataset.

2.1. Simulink

A quadcopter is a helicopter as presented in Fig. 2 has four equally spaced rotors fixed at the corners of a square body and the propellers move clockwise and anticlockwise. The inertial frame is defined by the ground, with gravity pointing in the - z-direction. The orientation of the quadcopter with the rotor axes directing in the + z-direction and the arms pointing in the x and y directions are defined by the body frame. The Euler angles which consist of roll, pitch and yaw are displayed in Fig. 3. refers to the coordinate system which can describe the properties such as roll, pitch and yaw of a quadcopter. The roll and pitch angles give the tilt of rotors with respect to the vertical gravitational axis whereas the yaw angle gives the amount of rotation around the vertical axis. The

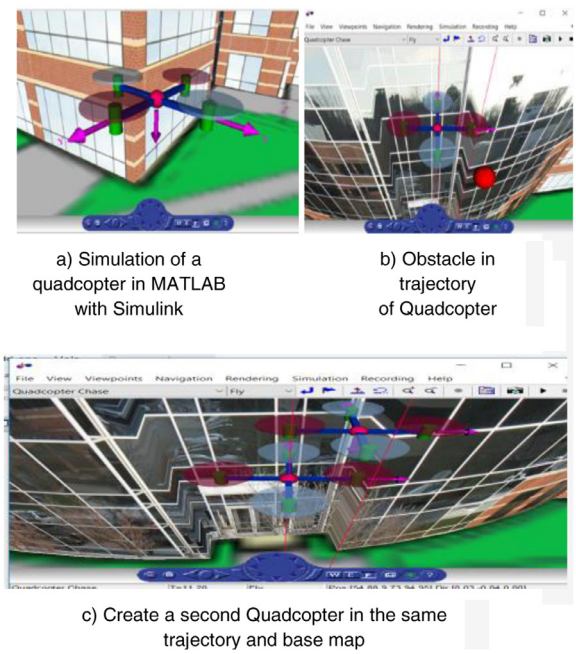


Fig. 4. Snapshots of Quadcopter Project in MATLAB-based Simulink in different scenarios

The quadcopter model has four wings and is widely used for a variety of computer vision applications due to its small size and high stability. The design of remote-controlled quadcopter includes Proportional Integral Derivative (PID) controller which is implemented with ardupilot mega board. The system consists of IMU which comprises accelerometer and gyro sensors to determine the system orientation and speed control of four motors to enable the quadcopter to fly in all directions. The Quadcopter Project example in MATLAB shows the use of Simulink to model a quadcopter based on the PARROT® series of mini-drones as defined in Fig. 4(a). The desired images in the Simulink quadcopter are done by replacing the inbuilt environment images with our images in the texture folder and capturing the view from the quadcopter by changing the viewpoints or by adding the camera sensor node. The Simulink design verifier acknowledges to identify of design errors and causes test case scenarios for model verification. In the MATLAB-based Simulink simulator, a virtual reality model is added by inserting an obstacle of different shapes like a sphere at particular coordinates as represented in Fig. 4(b). The predefined viewpoints and camera angles can also be modified to obtain required data from different viewpoints. In the MATLAB-based quadcopter model, multiple customizations such as modifying the surrounding environment of simulation by performing certain operations such as changing the ground map or surrounding building images or creating a second quadcopter in the same trajectory and base map as shown in Fig. 4(c). Further, the next section contains a methodology of data collection from the simulator followed by providing annotation information for the captured simulated dataset.

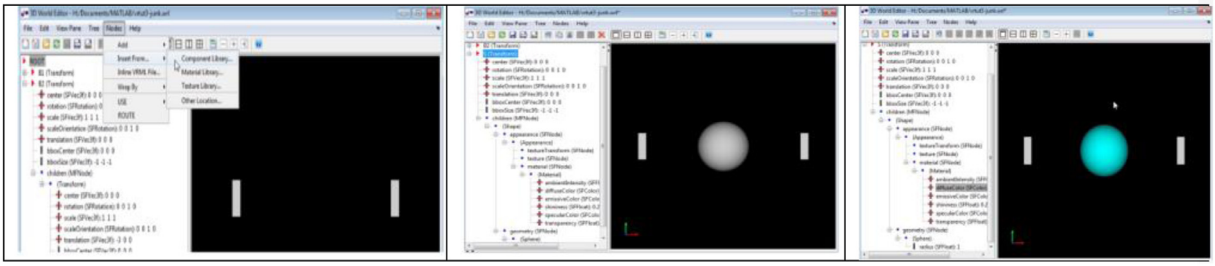


Fig. 5. Block Parameters of the Simulation in MATLAB-based Simulink

3. Methodology of captured simulated dataset

The MATLAB-based Simulink is compatible while dealing with control theory and model-based design. Simulink sessions can be started in two ways: by entering the command Simulink in the MATLAB command window and alternatively, clicking on the Simulink icon in the MATLAB toolbar. There is the utilization of Simulink for the creation of a simulated dataset of multiple categories which can further be deployed in object detection algorithms. The requirements of a custom dataset must fulfil the following constraints:

a should be free of copyright or at least freely usable i.e. open-source within the vision community, which is a strong criterion
b the number of different targets, as well as object types, should be diverse enough to represent the needs of the object recognition applications
c the ground truth, included in the annotations distributed with the dataset, should be complete enough to acknowledge the estimation of target object detection algorithms.

The proposed simulated dataset completely adheres to the above-discussed constraints. The simulated environment details are discussed in the next section.

3.1. Process of simulation

The steps for creating a Quadcopter model is to simply run the asbQuadcopterStart command in the command window or in the Simulink model library click on 'Examples' then in the model examples search for Aerospace Blocksetin which click for Quadcopter model. After performing these steps Simulink will initialize your Quadcopter model. The backend files can be seen in the current folder menu on the left side and the current variable can be seen in the workspace menu. Now, the model can be modified by making changes in the 3D world editor or by modifying the block diagram. The modifications can also be done by editing code in the backend files in the editor. The effects of modifications can be seen by running the simulation in the simulation window. The Quadcopter project example is available in the aerospace blockset of Simulink. The programming of drones with Simulink includes three steps:

1 Designing- Simulink can be used to design physical systems using block modelling. A comprehensive library of predefined blocks helps you to build models.
2 Simulation- The blocks from the dynamic library can be extended to the model using Simulink Editor and further, the blocks can be connected with each other to establish mathematical relationships between multiple components.
3 Deployment- Then, the simulated model is deployed with the help of suitable hardware and additional requirements.

Several simulation experiments were explored such as importing a customized map to the Simulink environment, building a virtual reality world using Simulink 3D animation (deformation of a sphere); editing the Quadcopter environment (changing the buildings around the Quadcopter environment); adding obstacles in the trajectory of the Quadcopter; inserting horizontal and vertical motion; inserting another Quadcopter to the trajectory of the existing one and adding touch sensor to the Quadcopter. We have performed experiments for the process of simulation in order to perform customized settings for obtaining the simulated dataset as described in Table I. The building of a virtual reality world using Simulink 3D animation while adding obstacles at particular coordinates in the trajectory of the Quadcopter is depicted in Fig. 6. The insertion of horizontal motion in the Quadcopter for editing the block models to modify the default vertical motion into horizontal motion of the Quadcopter. In addition, to import the desired images in our Simulink quadcopter, we simply replaced the inbuilt environment images with campus building images in the texture folder but kept the names of the replaced images the same as that of the inbuilt images so that the code of the project can access replaced images at run time. For capturing the view of the environment, there is a need to adjust the dial which changes these (i.e. in this case velocity is increased) after every 10 sec. These velocities are integrated to calculate acceleration. Finally, the calculated acceleration is expanded using VR signal Expander. The expanded signal is passed to Quadcopter translation in the VR sink.

All of the above-discussed experiments in a quadcopter setting play an important role in setting up an ideal environment for collecting the data from the simulator. Multiple settings have been done such as change of actors and camera position, controlling the speed of quadcopter, background images etc. to obtain a simulated dataset in a customized environment. This section provides a theoretical aspect of understanding the simulator with respect to collecting datasets from the simulator. The speed of the quadcopter can be controlled by using a digital clock to give time as input. The inputs from the digital clock are mapped with a velocity which is inside Matlab function blocks using switch cases.

4. Simulated UAV dataset

The simulation of the quadcopter is done through which a dataset of 6 simulated videos is retrieved. The simulated videos have different scenarios such as campus buildings, the presence of multiple obstacles, pedestrians etc. In MATLAB-based Simulink quadcopter model, different instances of navigation videos can be obtained either by modifying the surrounding environment of simulation by changing ground map, by changing surrounding building images and by adding stationary or moving obstacles. The additional viewpoints or camera angles can be introduced, also the predefined viewpoints and images and by adding stationary or moving obstacles. The additional viewpoints or camera angles can be introduced, also the predefined viewpoints and camera angles can be modified to obtain navigationnavigation video from different viewpoints. The simulation data inspector helps in the visualization of different data types that can be generated throughout the designing process as highlighted in Fig. 7(a) and (c). The comparisons of runs are made

Table I
Diverse configuration settings of MATLAB-based Simulator for capturing simulated dataset

Building a virtual reality world using Simulink 3D animation	Adding the boxes	Add a transform node. Select children and add shapes. Under shape, nodes select appearance and add appearance. Under shape, node select material and add material Add box node under geometry node Add another box using the same steps
	Adding the sphere	Select root. Click on the node and then click on a component library Select root. Click on the node and then click on a component library Add sphere.wrl Adjust all the sizes accordingly
Editing the Quadcopter Environment	Changing the buildings around the Quadcopter environment.	opening the absQuadcopterAH1-4.wrl file or absQuadcopterAH3.wrl file changes are made in the image URL option present inside the appearance parameter of the transform node of the editor any image URL can be inserted in the place of the original image URL and the changes are reflected in the quadcopter environment
Adding Obstacle in The Trajectory of the Quadcopter	Adding obstacles at particular coordinates in the trajectory of the Quadcopter	To add an obstacle in the path of the quadcopter, a Shape node is created in the ROOT The position of the obstacle is changed by changing the value of the transform and center parameters of the Shape node.
Inserting Horizontal Motion in The Quadcopter	Editing the block models to modify the default vertical motion into horizontal motion of the Quadcopter	Select the transform and rotation nodes of the quadcopter in the Block Parameters dialog box as shown in Fig. 5. The transform and rotation nodes reflect in the VR Sink of the block diagram of the quadcopter In the block diagram table of the quadcopter open the Axes to VR Axes block Attach the output line of x-axis to the given MUX to make the quadcopter move in the horizontal motion Connect the output lines of the block table to the quadcopter nodes of the VR Sink in the block diagram
Inserting Horizontal Motion in The Obstacle	Editing the block parameters to add horizontal motion to the obstacle present in the path of the Quadcopter	Add Shape node to the ROOT Select the transform parameter of the shape node in the Block Parameters dialog box. The transform and rotation parameters reflect in the VR Sink of the block diagram of the quadcopter Open the Axes to VR Axes block Attach the output line of x-axis to the given MUX to make the obstacle move in the horizontal motion Connect the output lines of the block table to the shape nodes of the VR Sink in the block diagram

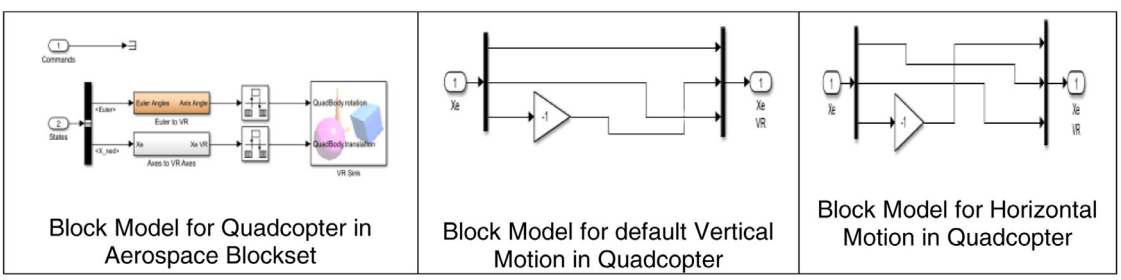


Fig. 6. Block Models for changing motions in MATLAB based Simulink

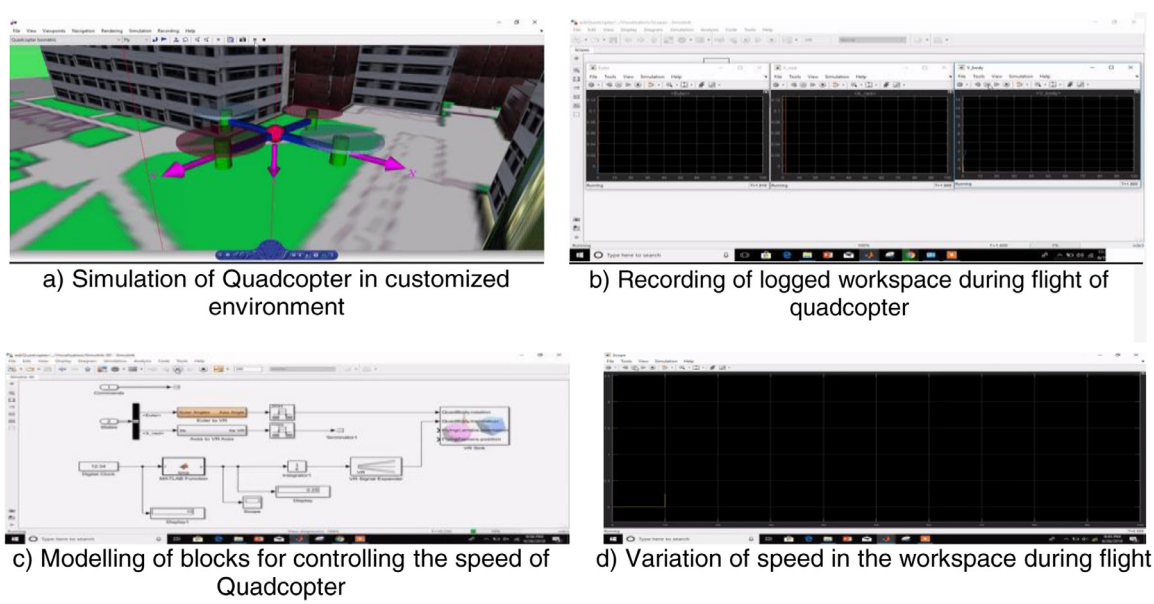


Fig. 7. Modelling of Quadcopter Project in MATLAB-based Simulink

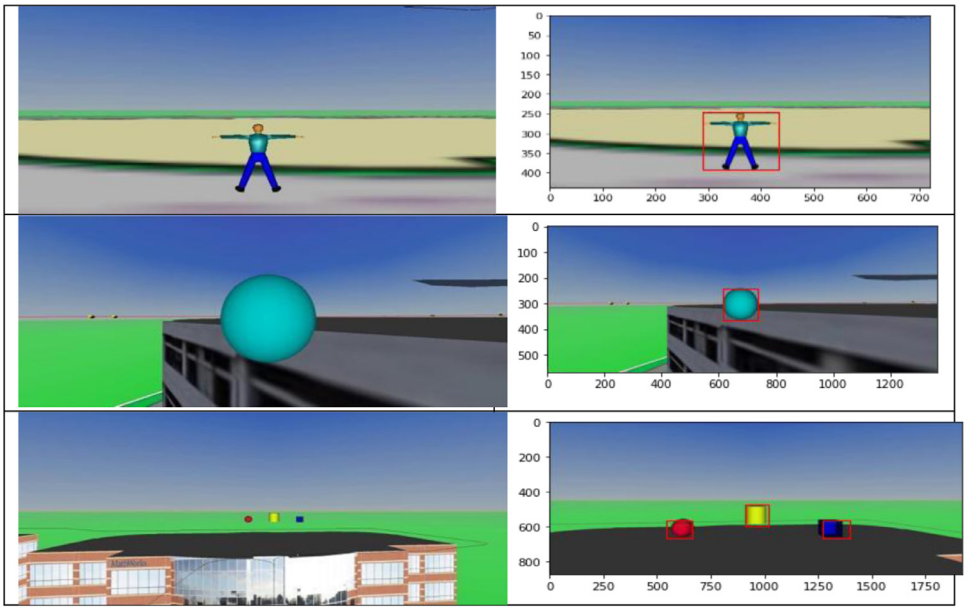


Fig. 8. Snapshots of dataset bounding boxes annotations from MATLAB-based Simulink

by recording logged workspace signals in the simulation data inspector by changing the visualization block, described in Fig. 7(b). The Quadcopter's trajectory can be controlled easily by introducing a user-defined trajectory function to the block diagram. Similarly, the Quadcopter's speed can be controlled by introducing a user-defined velocity function as represented in Fig. 7(d). Further, the modelling of blocks is done for controlling speed and variation in a workspace. The logging of data to the simulation data inspector from a Simulink model is performed by importing data from the base workspace. The visualization of data with the simulation data inspector supports the design, debugging, and verification workflows. The camera view of a quadcopter consists of two object views i.e. front-view and top-view mean the camera shooting along with the object and on the top of objects, respectively. A total of 2024 images have been obtained with multiple categories of objects such as obstacles 1, 2 and 3 in one scene, the presence of pedestrians and drones. We use labelling (Yu, Chen, Lee, Chen & Hsiao, 2019), a graphical image annotation tool to manually annotate the videos at 10 FPS. It is written in Python language and utilizes Qt for the graphical user interface. Then annotations will be saved in different formats (saved XML files in PASCAL VOC format, vastly utilized by ImageNet). Additionally, this annotation tool also supports the You Only Look Once (YOLO) object detector format. The collected dataset from the simulator can be further utilized for diverse object detection applications based on aerial datasets. The frames are retrieved from captured videos and perform annotation in the form of bounding boxes, displayed in Fig. 8. There are mainly three categories of objects with one or multiple instances such as pedestrian, drone, multiple obstacles etc. This overall study concludes with a description of a MATLAB-based simulator for animated aerial data. Further, a number of simulated videos have been generated from the simulator to check quadcopter settings in a customized environment. These videos are short in length because of the limited number of actors and can be extended by incorporating a number of classes as discussed in the above sections. The videos are then converted into frames for processing the recent object detection and recognition algorithms. The obtained dataset is satisfactorily in nature and can be further utilized by recent object recognition advances in the computer vision domain. The emerging deep learning in artificial intelligence makes a remarkable growth in image processing. As a future use, deep learning-based complex object detection and classification algorithms can be tested on our simulated UAV dataset.

4.1. Performance comparison of the simulated dataset using object detection algorithms

Recent years have seen noteworthy developments in object recognition and detection problems because of the advent of deep learning-based artificial intelligence algorithms in computer vision. The obtained simulated dataset is annotated with bounding box coordinates (Xmin, Xmax, Ymin and Ymax). For deployment of the generated simulated dataset in object detection techniques, we have to first check the feasibility of the dataset such that it can really be useful for detection. Some traditional object detection approaches such as person detectors, optical flow and edge detectors applied on simulator captured videos in different environments to test the annotated dataset (Pathak, Pandey & Rautaray, 2018). The optical flow studies the apparent velocities-based distribution of objects and by assessing the optical flow between video frames, the velocities of objects in the video are measured (Meier, Brockers, Matthies, Siegwart & Weiss, 2015). Edge detection is a popular vision technique, used for a variety of domains such as image segmentation (Huguet, De Andrade, Carceroni & Araújo, 2004) and object detection (Ding & Zhao 2018) problems. The person detector algorithmdetected a simulated pedestrian and a bounding box is inserted around it as shown in Fig. 9(b). The coordinates of the bounding box then matched with ground truth values which resulted in correct detection of an object from the simulated dataset. The Canny edge detector applied on pedestrians and the corresponding output is highlighted in Fig. 9(c). The captured videos are in motion so we pass respective frames to the optical flow-based detector and motion is detected in pedestrian movements as shown in Fig. 9(d). The higher efficiency attribute of one-stage detectors (Redmon & Farhadi, 2017; Liu, Anguelov, Erhan, Szegedy, Reed, Fu & Berg, 2016). over two-stage detectors (Ren, He, Girshick & Sun, 2015; He, Gkioxari, Dollár & Girshick, 2017) makes them deployable in object detection scenarios. The researchers eventually shifted towards one-stage detectors due to adaptability towards meeting challenges like providing high speed and fewer memory requirements. We passed our simulated dataset into some single-stage algorithms such as YOLOv3 (Redmon & Farhadi, 2018),

TinyYolov3 (Yi, Yongliang & Jun, 2019) and RetinaNet (Lin, Goyal, Girshick, He & Dollár, 2017) to check their feasibility. The key idea of YOLOv3 and TinyYolov3 is to look at an image to predict the number of objects and identify the location of objects. These approaches trained

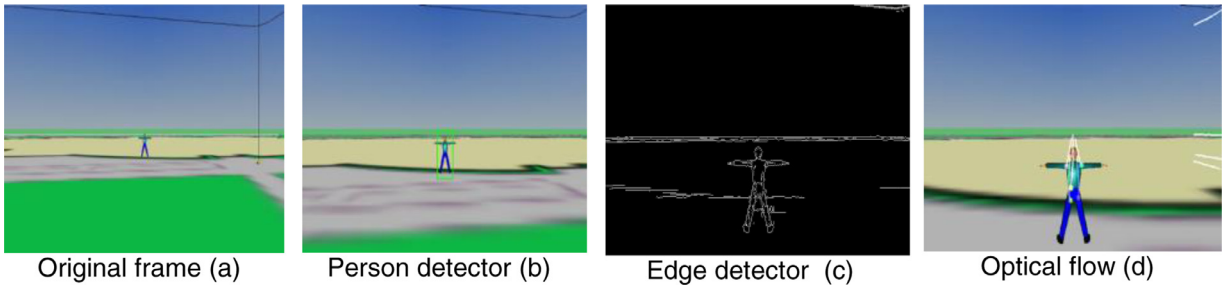


Fig. 9. Detection results achieved on Simulink captured frames

Table II
Performance comparisons of object detectors on UAV simulated dataset

Object Detection Approach	Accuracy	Time Required (sec)
Yolov3	98.20	3.62
TinyYolov3	87.26	**2.49**

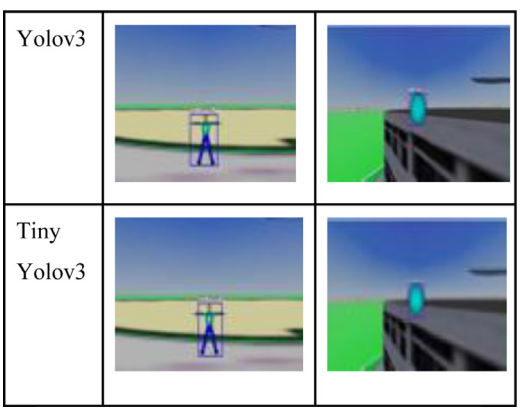


Fig. 10. Detection results achieved on Simulink captured frames

on complete images and directly boosted detection performance. These detectors are not trained from scratch so the categories of objects acquired from the originally trained pedestrian object frame into Yolov3, predicted value is the same predicted as traffic light dataset i.e. MS-COCO (Lin, Maire, Belongie, Hays, Perona, Ramanan & Zitnick, 2014). When we pass predicted as a traffic light and 52.63 accuracy with a processing time of 7.21 seconds. This might require further fine-tuning with the dataset which researchers can explore working in the object detection domain. Similarly, when we pass the pedestrian object frame into TinyYOLOv3, the predicted value is the same as ground and the 87.26 accuracy obtained with a processing time of 2.49 seconds whereas after passing round shape obstacle into the algorithm, the obstacle predicted as a sports ball and 72.25 accuracy with a processing time of 6.09 seconds. We also passed the frame which is carrying three obstacles, only one object is identified by TinyYolov3 and predicted as tv with a 57.83 accuracy value and 3.27 processing time.

The overall summary of comparable results over the simulated dataset is highlighted in Table II and further outputs from different object detection algorithms are presented in Fig. 10. By testing the dataset into recent object detection and recognition algorithms, it can be concluded that this simulated dataset of multiple objects is deployable in object detection scenarios. But if we only focus on deep learning-based approaches, the size of the acquired dataset is not enough so one can add more instances of objects or perform augmentation on the given frames.

5. Conclusion

The MATLAB-based Simulink provides a suitable platform for researchers working in UAV flight domains. This paper helps in simulating the behavior of UAVs by MATLAB-based Simulink by generating a dataset for object detection and recognition processes. A validated simulation model is developed for the computing paradigm encompassing the effects of flight navigation spaces for UAVs. The experiments for building a credible UAV simulation model are created by enforcing several settings such as altering speed, sensors, number of objects in-flight navigation etc. The customized hardware settings of the selected quadcopter model have a significant impact on the UAV simulation model, and a realistic representation of the simulated model is built. This simulation helps in generating a verified simulated dataset of different objects that mimics the working environment of UAVs during flight. The produced simulated UAV dataset has various object classes such as obstacles, another drone etc. and is tested in various situations. The dataset is well annotated and tested against state-of-the-art deep learning-based object detection algorithms. The one-stage Yolov3 object detector achieved 98.20 mAP when compared with the Tiny-Yolov3 detector and the corresponding visualization of detection results is also highlighted.

In the future, we aim to create more instances of objects in the simulated dataset. This paradigm helps researchers to get hands-on training with the flying conditions of UAVs. The proposed research is justified by the use of recent object detection and classification techniques by performing some detections on captured frames by state-of-the-art detection algorithms. The study is a good platform for engineering graduates to learn simulation of drones, the creation of simulated datasets for evaluating various objection detection algorithms based on deep learning-based methods.

Declaration of Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Álvares, P., Silva, L., & Magaia, N. (2021, March). Blockchain-Based Solutions for UAV-Assisted Connected Vehicle Networks in Smart Cities: A Review, Open Issues, and Future Perspectives. In *Telecom* (pp. 108–140). Multidisciplinary Digital Publishing Institute. (Vol. 2, No. 1.

Amarasinghe, A., Suduwella, C., Elvitigala, C., Niroshan, L., Amaraweera, R. J., Gunawardana, K., & Keppetiyagama, C. (2017, November). A machine learning approach for identifying mosquito breeding sites via drone images. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems* (pp. 1–2).

Bashir, N., Boudjit, S., & Zeadally, S. (2022). A closed-loop control architecture of UAV and WSN for traffic surveillance on highways. *Computer Communications*.

Bondi, E., Dey, D., Kapoor, A., Piavis, J., Shah, S., Fang, F., & Tambe, M. (2018, June). Airsim-w: A simulation environment for wildlife conservation with uavs. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies* (pp. 1–12).

Causa, F., & Fasano, G. (2021). Multiple UAVs trajectory generation and waypoint assignment in urban environment based on DOP maps. *Aerospace Science and Technology, 110*, Article 106507.

Chen, S., Zhou, W., Yang, A. S., Chen, H., Li, B., & Wen, C. Y. (2022). An End-to-End UAV Simulation Platform for Visual SLAM and Navigation. *Aerospace, 9*(2), 48.

Choi, J. H., Choi, B. J., Kim, N. G., Lee, C. W., Seo, J. P., & Jun, B. H. (2021). Estimation of Potential Risk and Numerical Simulations of Landslide Disaster based on UAV Photogrammetry.

Dabney, J. B., & Harman, T. L. (2004). *Mastering simulink*. Upper Saddle River: Pearson/Prentice Hall (Vol. 230).

Ding, S., & Zhao, K. (2018, March). Research on daily objects detection based on deep neural network. *IOP Conference Series: Materials Science and Engineering*. IOP Publishing (Vol. 322, No. 6.

Garcia, R., & Barnes, L. (2009). Multi-UAV simulator utilizing X-Plane. In *Selected papers from the 2nd International Symposium on UAVs, Reno, Nevada, USA June 8–10, 2009* (pp. 393–406). Dordrecht: Springer.

George, E. A., Tiwari, G., Yadav, R. N., Peters, E., & Sadana, S. (2013, August). UAV systems for parameter identification in agriculture. In *2013 IEEE Global Humanitarian Technology Conference: South Asia Satellite (GHTC-SAS)* (pp. 270–273). IEEE.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961–2969).

Huguet, A. B., De Andrade, M. C., Carceroni, R. L., & Araújo, A. D. A. (2004, October). Color-based watershed segmentation of low-altitude aerial images. In *Proceedings. 17th Brazilian Symposium on Computer Graphics and Image Processing* (pp. 138–145). IEEE.

Kayton, M., & Fried, W. R. (1997). *Avionics navigation systems*. John Wiley & Sons.

Koenig, N., & Howard, A. (2004, September). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)* (pp. 2149–2154). IEEE. (Vol. 3.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740–755). Cham: Springer.

Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (pp. 2980–2988).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21–37). Cham: Springer.

Luukkonen, T. (2011). Modelling and control of quadcopter. Independent research project in applied mathematics. *Espoo, 22*, 22.

Lyu, Y., Cao, M., Yuan, S., & Xie, L. (2021). Vision Based Autonomous UAV Plane Estimation and Following for Building Inspection. arXiv preprint arXiv:2102.01423.

Ma, Y., Wu, X., Yu, G., Xu, Y., & Wang, Y. (2016). Pedestrian detection and tracking from low-resolution unmanned aerial vehicle thermal imagery. *Sensors, 16*(4), 446.

Matlekovic, L., Juric, F., & Schneider-Kamp, P. (2022). Microservices for autonomous UAV inspection with UAV simulation as a service. *Simulation Modelling Practice and Theory*, Article 102548.

McNeal, G. S. (2014). Drones and aerial surveillance: Considerations for legislators. *Brookings Institution: The Robots Are Coming: The Project on Civilian Robotics*.

Meier, D., Brockers, R., Matthies, L., Siegwart, R., & Weiss, S. (2015, September). Detection and characterization of moving objects with aerial vehicles using inertial-optical flow. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2473–2480). IEEE.

Nemati, A., & Kumar, M. (2014, June). Modeling and control of a single axis tilting quadcopter. In *2014 American Control Conference* (pp. 3077–3082). IEEE.

Ning, Z., Dong, P., Wen, M., Wang, X., Guo, L., Kwok, R. Y., & Poor, H. V. (2021). 5G-enabled UAV-to-community offloading: joint trajectory design and task scheduling. *IEEE Journal on Selected Areas in Communications, 39*(11), 3306–3320.

Nisansala, A., Weerasinghe, M., Dias, G. K. A., Sandaruwan, D., Keppitiyagama, C., Kodikara, N., & Samarasinghe, P. (2015). Flight Simulator for Serious Gaming. In *Information Science and Applications* (pp. 267–277). Berlin, Heidelberg: Springer.

Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of deep learning for object detection. *Procedia computer science, 132*, 1706–1717.

Perry, A. R. (2004, June). The flightgear flight simulator. In *Proceedings of the USENIX Annual Technical Conference* (Vol. 686).

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7263–7271).

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems, 28*, 91–99.

Shijith, N., Poornachandran, P., Sujadevi, V. G., & Dharmana, M. M (2017, October). Breach detection and mitigation of UAVs using deep neural network. In *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)* (pp. 360–365). IEEE.

Singh, A., Patil, D., & Omkar, S. N. (2018). Eye in the sky: Real-time Drone Surveillance System (DSS) for violent individuals identification using ScatterNet Hybrid Deep Learning network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 1629–1637).

Trepekli, K., Balstrøm, T., Friborg, T., Fog, B., Allotey, A. N., Kofie, R. Y., & Møller-Jensen, L. (2022). UAV-borne, LiDAR-based elevation modelling: a method for improving local-scale urban flood risk assessment. *Natural Hazards*, 1–29.

Tzelepi, M., & Tefas, A. (2017, August). Human crowd detection for drone flight safety using convolutional neural networks. In *2017 25th European Signal Processing Conference (EUSIPCO)* (pp. 743–747). IEEE.

Udroiu, R., Deaconu, A. M., & Nanau, C. Ş. (2021). Data Delivery in a Disaster or Quarantined Area Divided into Triangles Using DTN-Based Algorithms for Unmanned Aerial Vehicles. *Sensors, 21*(11), 3572.

Utlu, M., ÖZTÜRK, M. Z., & Şimşek, M. (2021). Evaluation of Rockfall Hazard Based On UAV Technology And 3D Rockfall Simulations.

Xiao, K., Ma, L., Tan, S., Cong, Y., & Wang, X. (2022). Implementation of uav coordination based on a hierarchical multi-uav simulation platform. In *Advances in Guidance, Navigation and Control* (pp. 5131–5143). Singapore: Springer.

Yaoming, Z., Yu, S. U., Anhuan, X. I. E., & Lingyu, K. O. N. G. (2021). A newly bio-inspired path planning algorithm for autonomous obstacle avoidance of UAV. *Chinese Journal of Aeronautics, 34*(9), 199–209.

Yi, Z., Yongliang, S., & Jun, Z. (2019). An improved tiny-yolov3 pedestrian detection algorithm. *Optik, 183*, 17–23.

Zhang, J. S., Cao, J., & Mao, B. (2017, July). Application of deep learning and unmanned aerial vehicle technology in traffic flow monitoring. In *2017 International Conference on Machine Learning and Cybernetics (ICMLC): 1* (pp. 189–194). IEEE.